DIGITAL FORENSICS

Fall 2025

Montgomery College

PORTFOLIO

A Collection of Labs & Projects

Submitted by:

Marqell Broxton

Email:

marqellbroxton@gmail.com

LinkedIn:

linkedin.com/in/marqellbroxton

Course: NWIT 263 - Digital Forensics

Lab Number: 02

Lab Title: Kali Linux Installation

Date: 9/11/2025

1. Introduction

This lab focuses on installing a Kali Linux virtual machine

2. Lab Objectives

By completing this lab, I was able to:

Install Kali Linux in a virtual environment and verify the installation with basic commands. By the end of this exercise, I will have a Kali Linux system working inside VirtualBox (or VMware).

3. Tools and Requirements

Download the 64-bit Installer ISO - Visit: https://www.kali.org/get-kali/
Install VMware Fusion - https://www.vmware.com/products/desktop-hypervisor/workstation-and-fusion

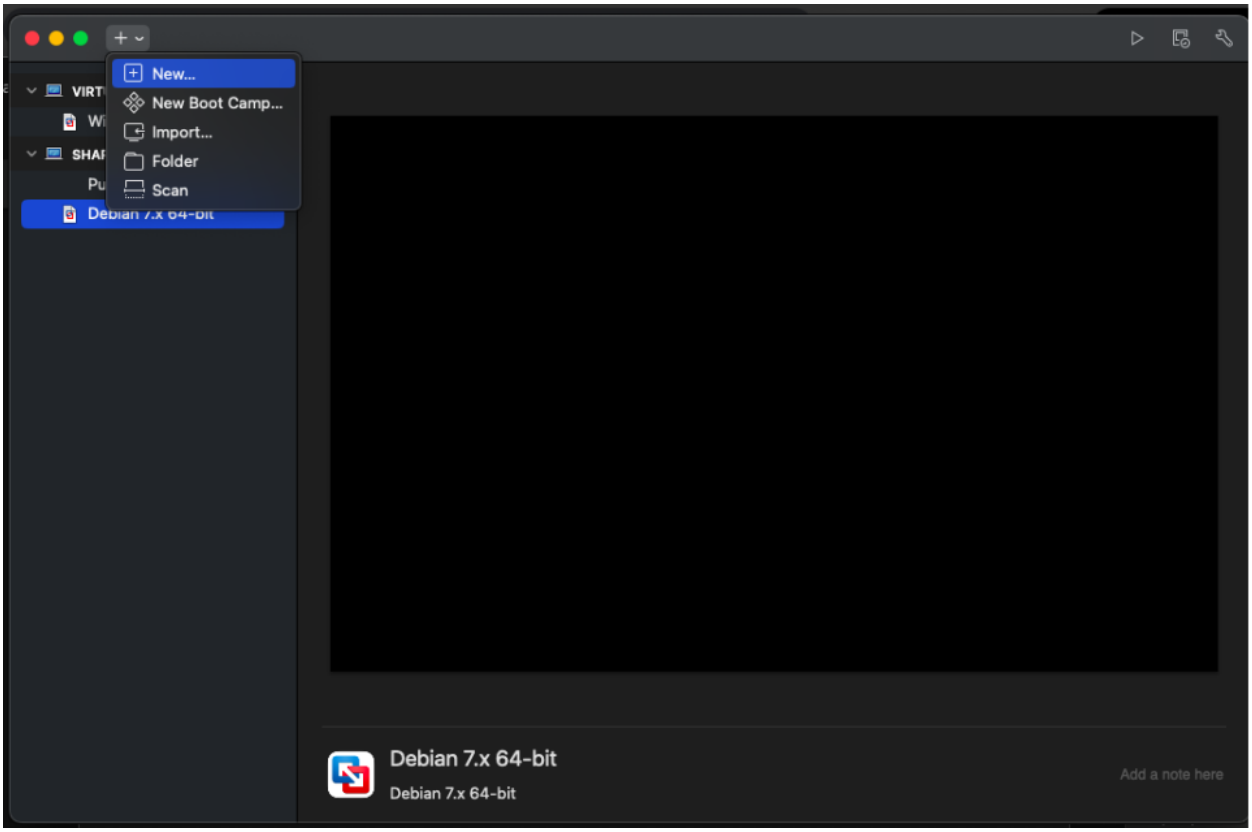
4. Lab Steps and Execution

Step 1: Downloaded the 64-bit Installer ISO and Installed VMware Fusion

Step 2: Created a Virtual Machine

- Opened VMware Fusion → Clicked New.
- Configured the VM as follows:
- Name: Kali Linux

- Type: Linux
- Version: Debian (64-bit)
- Memory (RAM): 4 GB
- Hard Disk: 20 GB



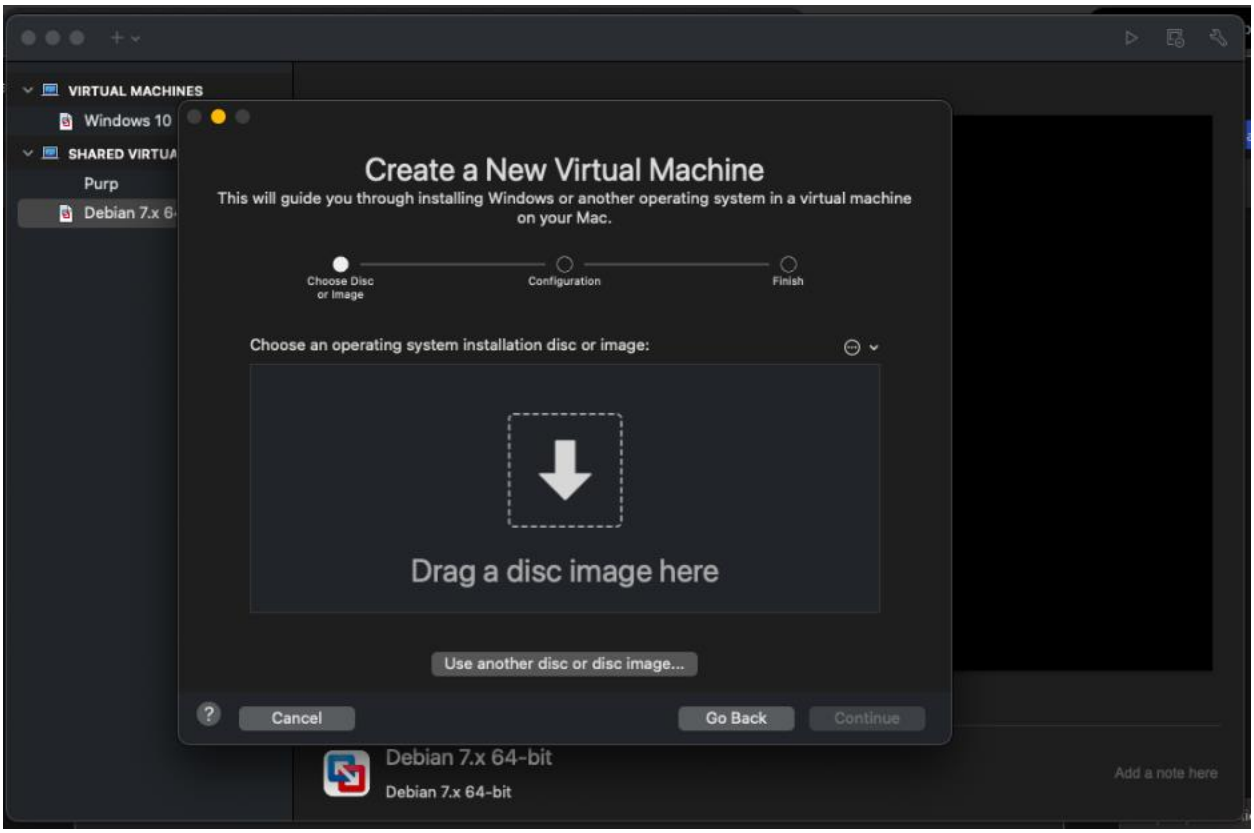
Step 3: Install Kali Linux

Started the VM and selected the downloaded ISO file as startup disk.

Choose Graphical Install.

Followed the on-screen instructions:

- Selected language, location, and keyboard.
- Created username and password.
- Partitioned disk → Choose Guided – Use Entire Disk.
- Waited for the installation to finish.

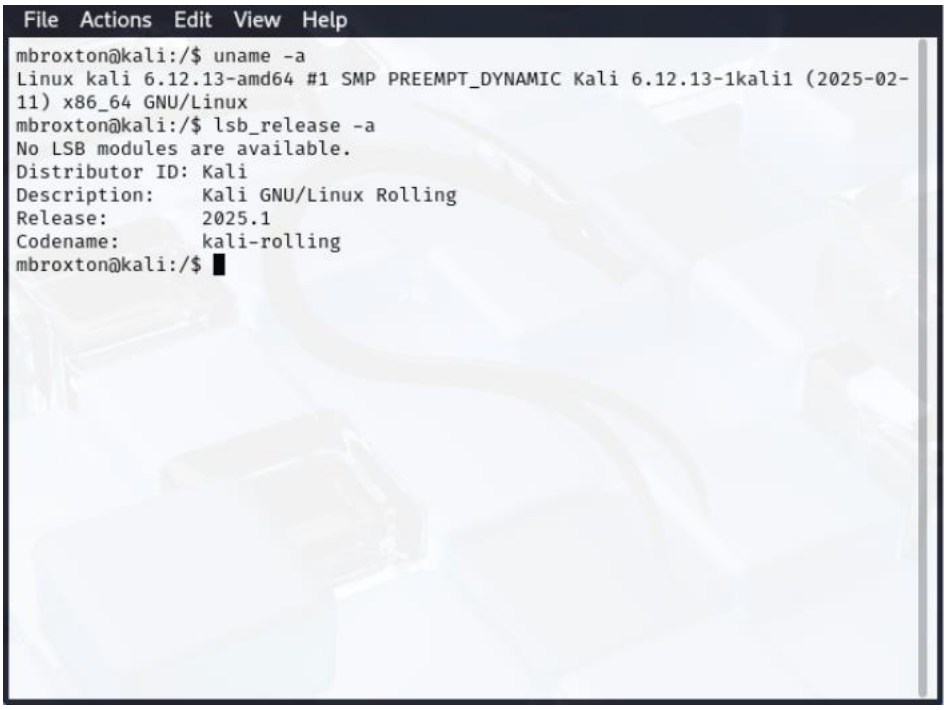


Step 4: Verify Installation

Logged in with my username and password.

Opened the terminal and typed:

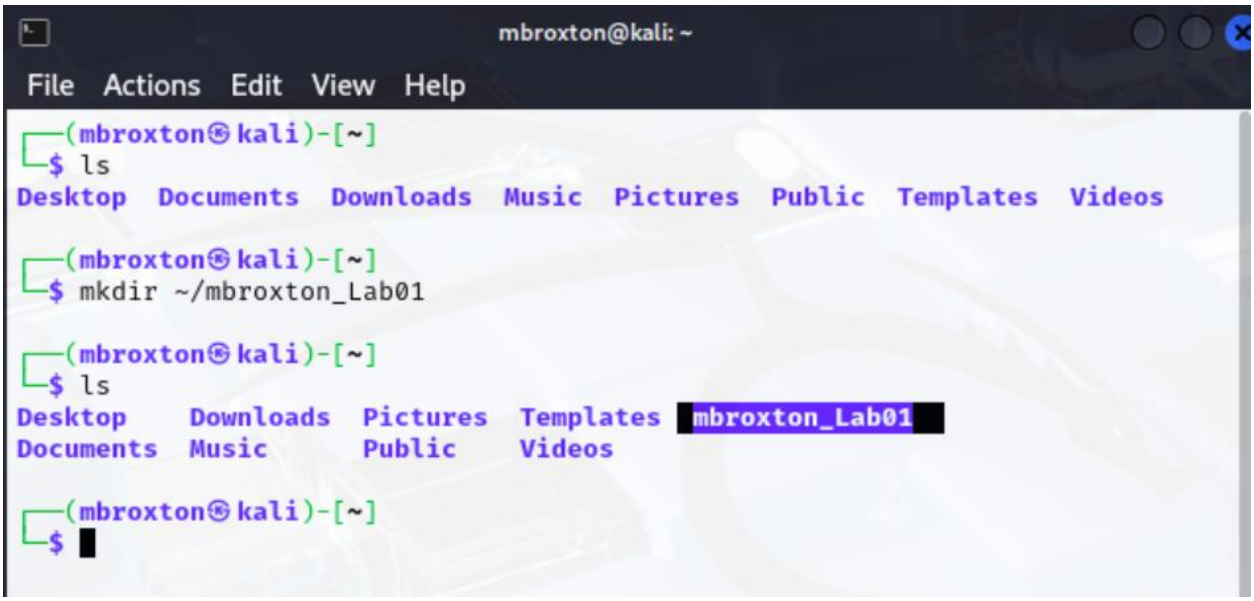
- uname –a
- lsb_release –a



Step 5: Personalize Your System

In the terminal, created a folder with my name:

- mkdir ~/mbroxton_Lab01



5. Conclusion

This lab successfully demonstrated the process of installing Kali Linux in a virtual environment using VMware Fusion. I was able to download the installer ISO, create a virtual machine, and complete the installation with the recommended system configurations. Next, I verified the installation with terminal commands and personalized the system. I confirmed that the Kali Linux environment was working properly. This lab was a foundational step in preparing a workspace for digital forensics.

6. Challenges and Observations

- Running uname -a and lsb_release -a confirmed the system was installed correctly. It was a good observation to see the system information
- Partitioning the disk and completing the graphical installation required careful attention
- Adjusting VM resources such as RAM and disk space was important

7. References

Kali ISO - Visit: https://www.kali.org/get-kali/

VMware Fusion - https://www.vmware.com/products/desktop-hypervisor/workstation-and-fusion

Lab Number: 03

Lab Title: Hashing Lab – Windows & Linux

Date: 9/19/2025

1. Introduction

This lab focuses on hashing files within Windows and Linux. The purpose is to understand how hashing ensures file integrity, recognize the differences between hash algorithms, and see how they are applied in digital forensics to confirm authenticity of evidence.

2. Lab Objectives

- Generate hashes of files in both Windows and Linux.
- Compare different types of hash functions (MD5, SHA-1, SHA-256).
- Verify that identical files produce the same hash and altered files do not.
- Document findings for forensic reporting.

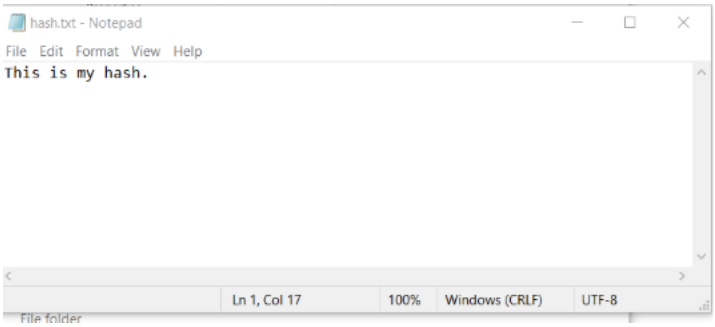
3. Tools and Requirements

- **Operating Systems:** Windows 10, Linux (Kali)
- **Windows Tool:** CertUtil (built-in utility)
- **Linux Tool:** sha256sum, md5sum, sha1sum

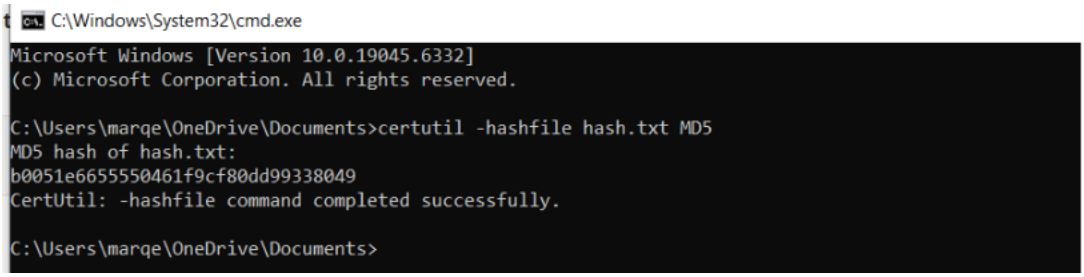
4. Lab Steps and Execution

Part 1 – Windows (Using CertUtil):

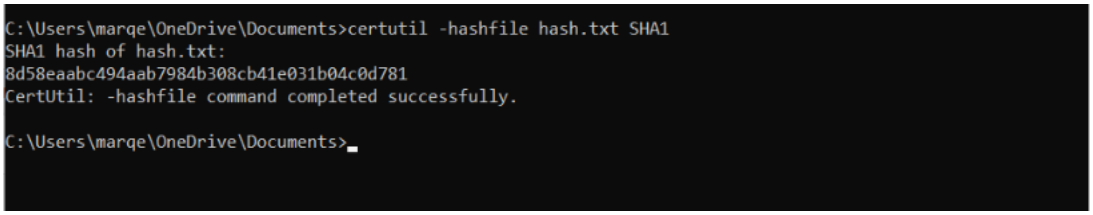
1. Created and saved a test file named hash.txt in Documents.



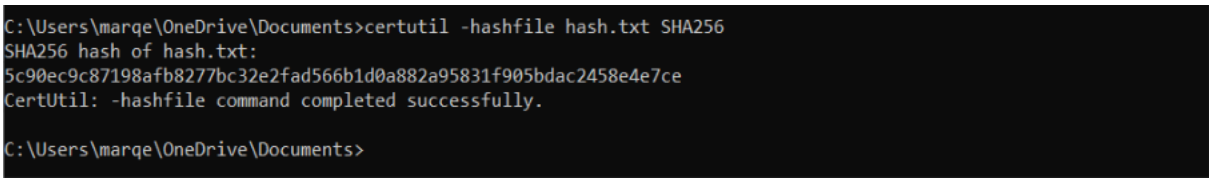
2. Opened Command Prompt and navigated to the Documents directory.
3. Ran the following commands:
   - certutil -hashfile hash.txt MD5



   - certutil -hashfile hash.txt SHA1



   - certutil -hashfile hash.txt SHA256



- certutil -hashfile hash.txt SHA512

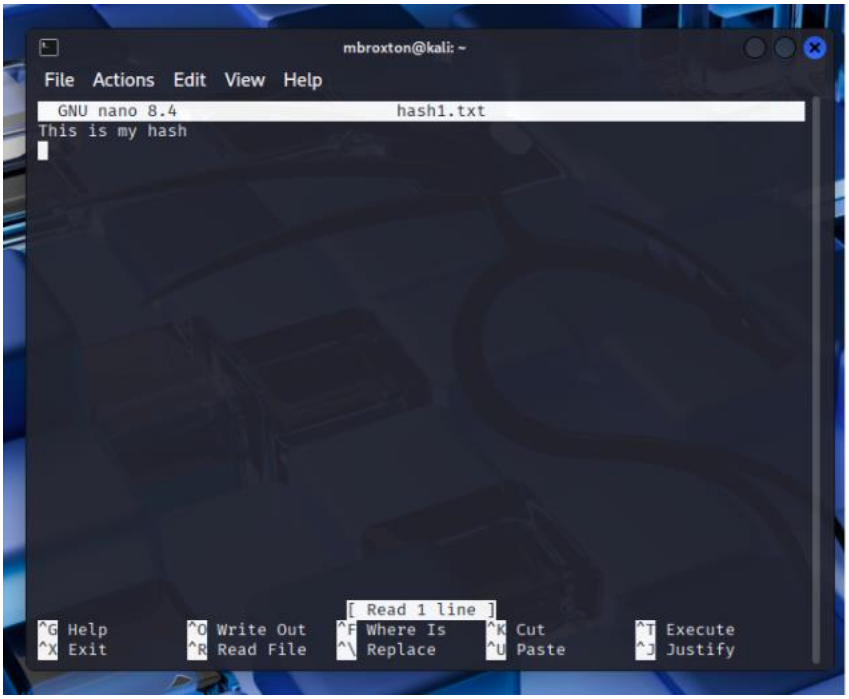
```
C:\Users\marqe\OneDrive\Documents>certutil -hashfile hash.txt SHA512
SHA512 hash of hash.txt:
8eca285136f51cbbbc084da954b50a7e06a7f8a085c89c6a340606010d5ff2eb016e3e1a96e90af0380edc26d50c6d660cc8ac351035159e727c06d4
53a8324f
CertUtil: -hashfile command completed successfully.

C:\Users\marqe\OneDrive\Documents>
```

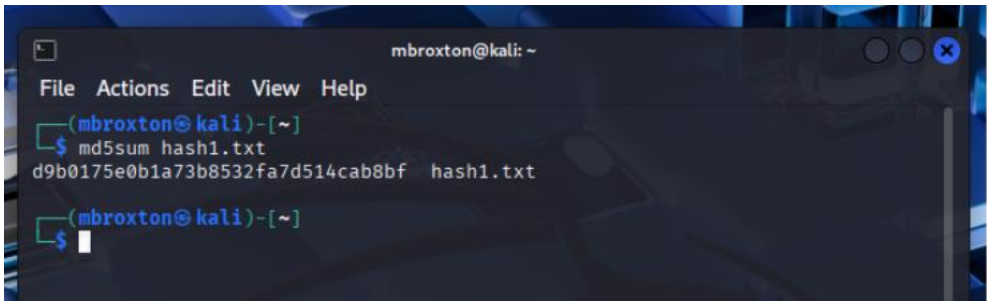
4. Observed that each command generated a unique hash value for the file depending on the algorithm.

Step 2: Linux (Using Built-in Hash Commands)

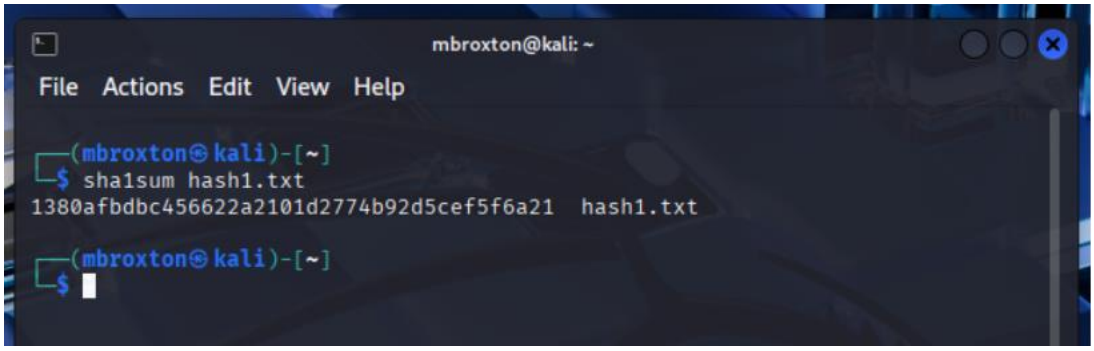
1. Created a file named hash1.txt in the Linux home directory



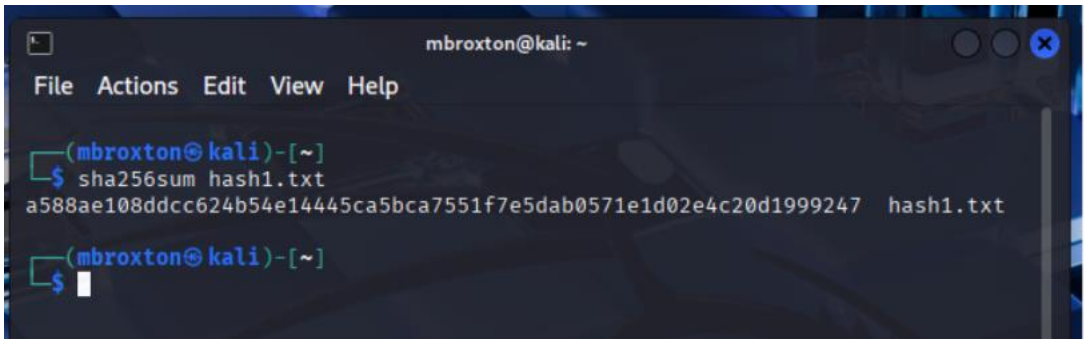
2. Opened terminal and ran:
   - md5sum hash1.txt



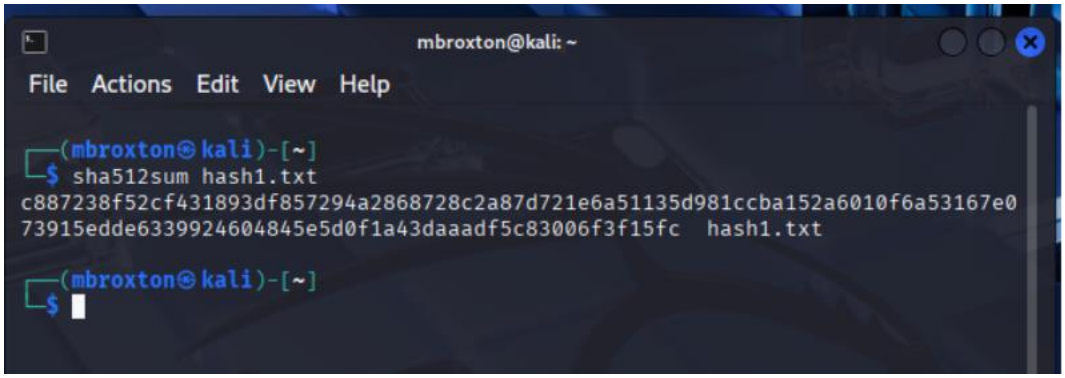
- sha1sum hash.txt



- sha256sum hash.txt



- sha512sum hash.txt



3. Edited the file slightly (removed period at the end). The new hash values were completely different, demonstrating the sensitivity of hashing to file changes.

4. Verified the integrity of the original file.

5. Conclusion

This lab demonstrated how hashing is used to ensure data integrity across Windows and Linux. Identical files consistently produced the same hash, while even minor alterations resulted in entirely different hashes. In digital forensics, hashing is crucial for validating evidence, ensuring files have not been altered during investigations or transfers.

6. Challenges and Observations

Remembering the exact syntax for CertUtil required careful attention.
Linux commands were straightforward but required checking file path accuracy.
Even a small edit (adding one character) drastically changed the hash, proving the reliability of hashing for tamper detection.
Using multiple algorithms reinforced the idea that while MD5 and SHA-1 are fast, SHA-256 is more secure.

7. References

Microsoft Docs – CertUtil: https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/certutil
Linux man pages (man sha256sum, man md5sum)

Lab Number: 04

Lab Title: Using FTK Imager to Create, Mount, and Dismount a Forensic Image

Date: 9/19/2025

1. Introduction

This lab focuses on using FTK Imager to create a forensic image of a storage device, mount the image for analysis, and properly dismount it afterward. Imaging ensures that investigators work with a verified copy of evidence rather than altering the original. Mounting allows forensic analysts to browse and analyze files in a controlled environment.

2. Lab Objectives

By completing this lab, I was able to:

- Create a forensic image of a USB drive using FTK Imager.
- Save the image in E01 (Expert Witness Format) with proper fragment size.
- Generate MD5/SHA1 hashes to verify image integrity.
- Mount the forensic image in read-only mode.
- Browse and analyze the mounted image.
- Properly dismount the image to maintain forensic soundness.

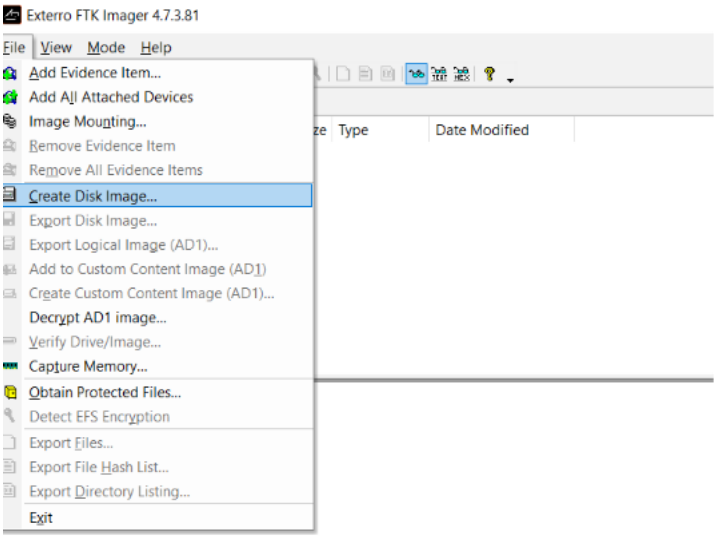
3. Tools and Requirements

- **Operating System:** Windows 10
- **Software:** FTK Imager (AccessData)
- **Storage Device:** USB drive (used as source)
- **Destination Folder:** B:\Forensic_Images\

4. Lab Steps and Execution

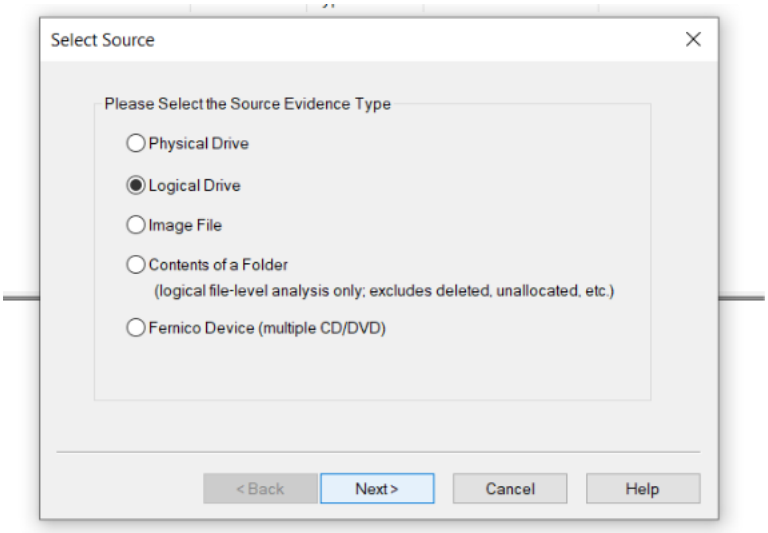
Step 1: Launch FTK Imager

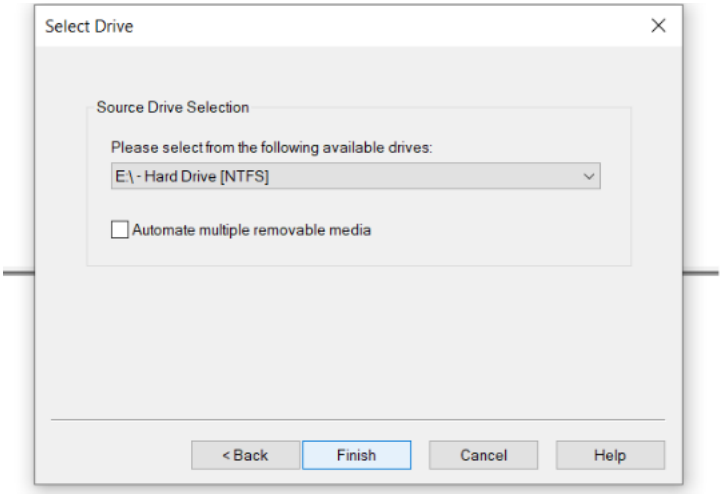
- Opened FTK Imager as Administrator.
- Selected File → Create Disk Image.



Step 2: Select Source Drive

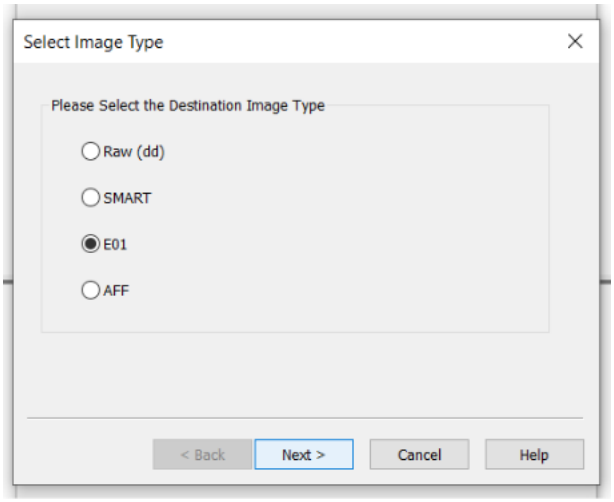
- Chose Logical Drive and selected my USB drive.
- Clicked Finish to confirm the source.



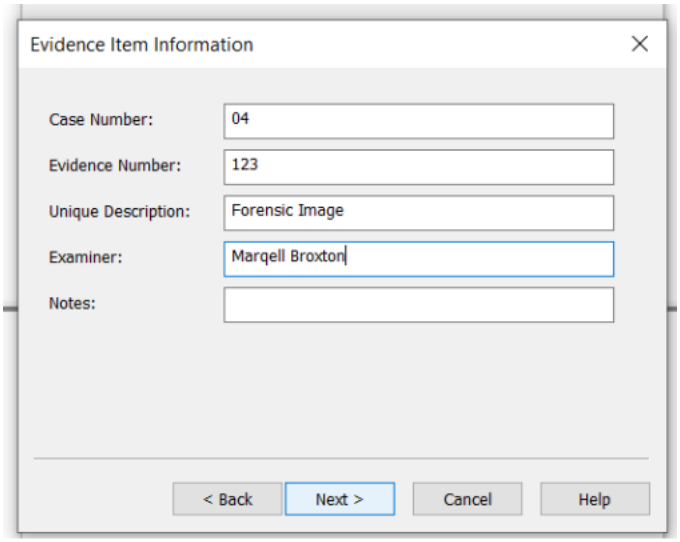


Step 3: Select Image Type and Destination

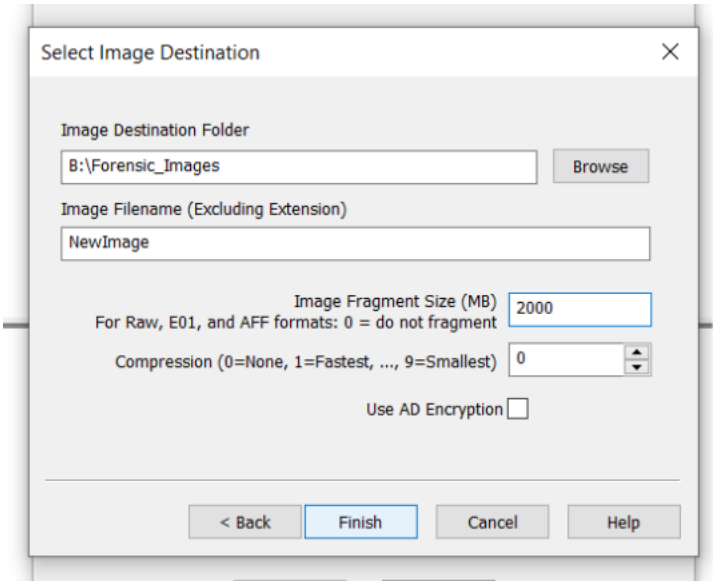
- Selected E01 (EWF) format.



- Entered case information (Case Number: 04, Examiner: Marqell Broxton).

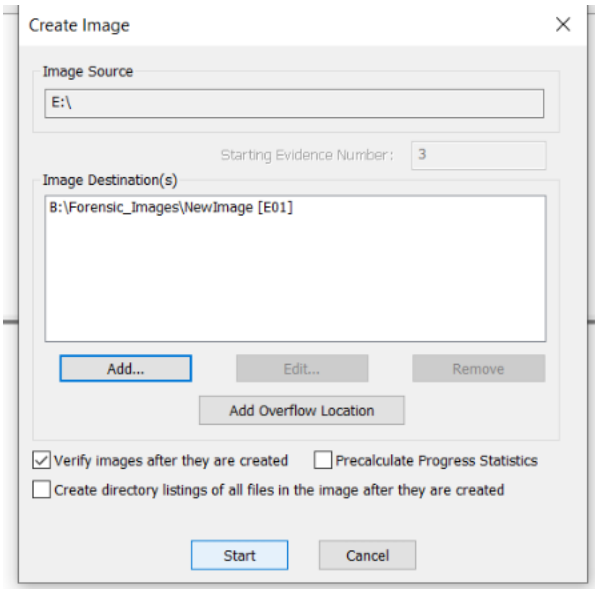


- Saved the image as NewImage.E01 in B:\Forensic_Images.
- Set fragment size to 2000 MB.
- Enabled MD5 and SHA1 hashing for verification.

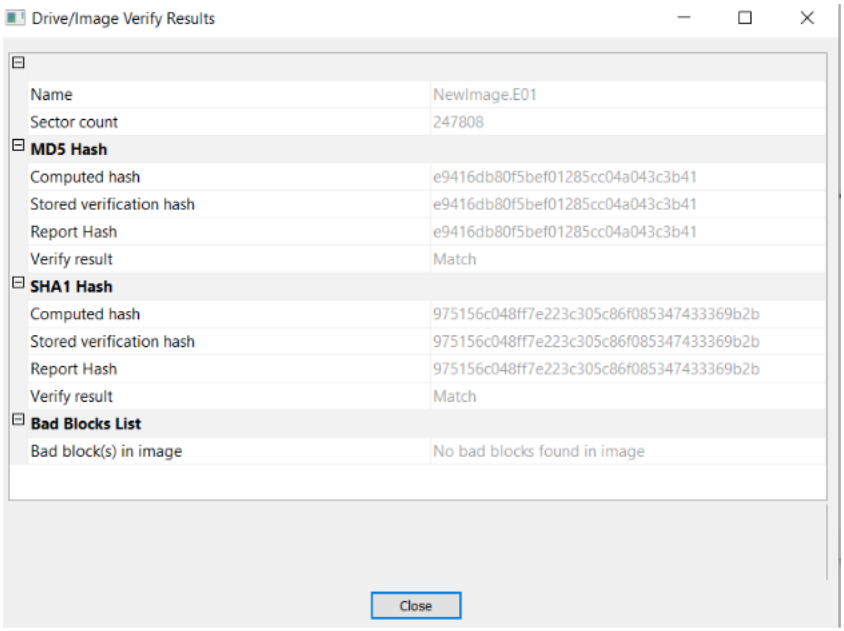


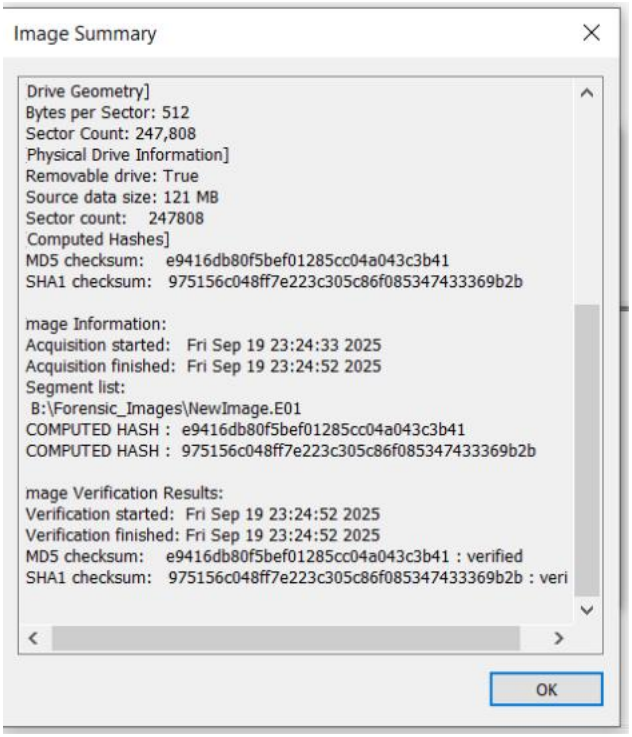
Step 4: Start the Imaging Process

- Clicked Start. FTK imaged the USB drive and generated MD5/SHA1 hashes.



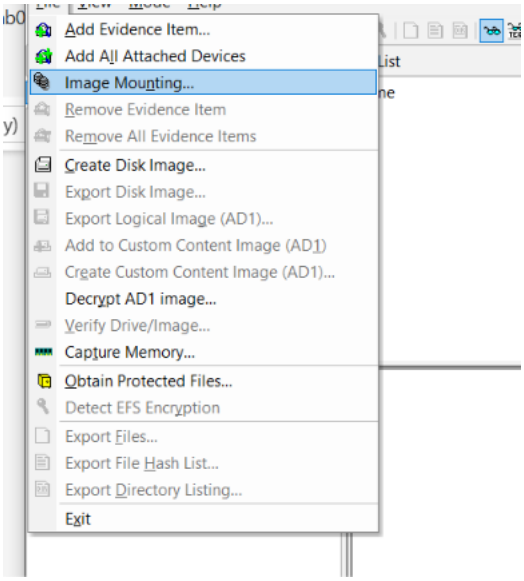
- Verified that the source hash and image hash matched.



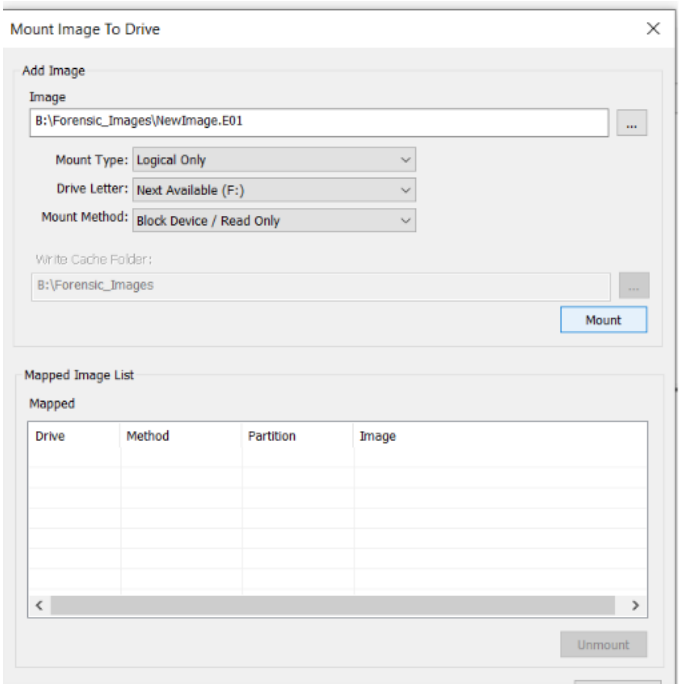


Step 5: Mount the Image

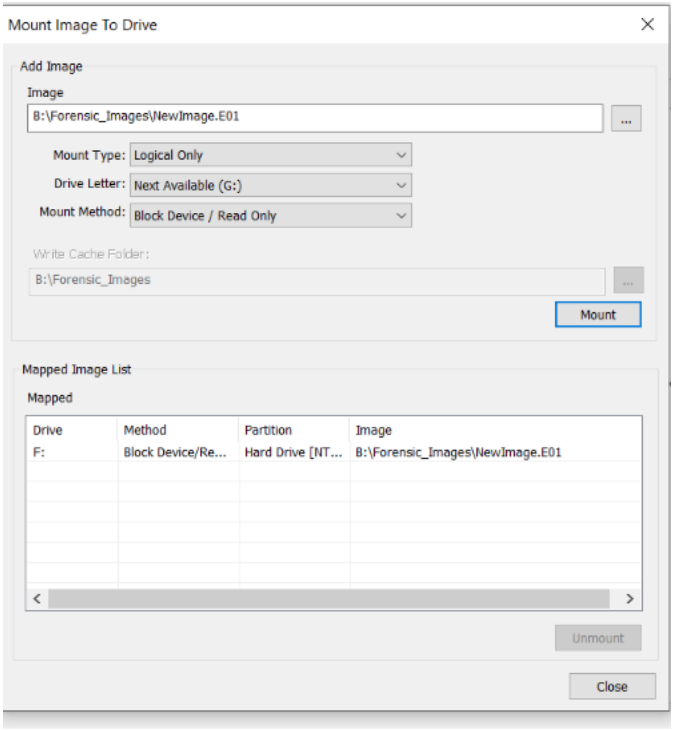
- Selected File → Image Mounting.



- Chose the newly created forensic_image.e01.



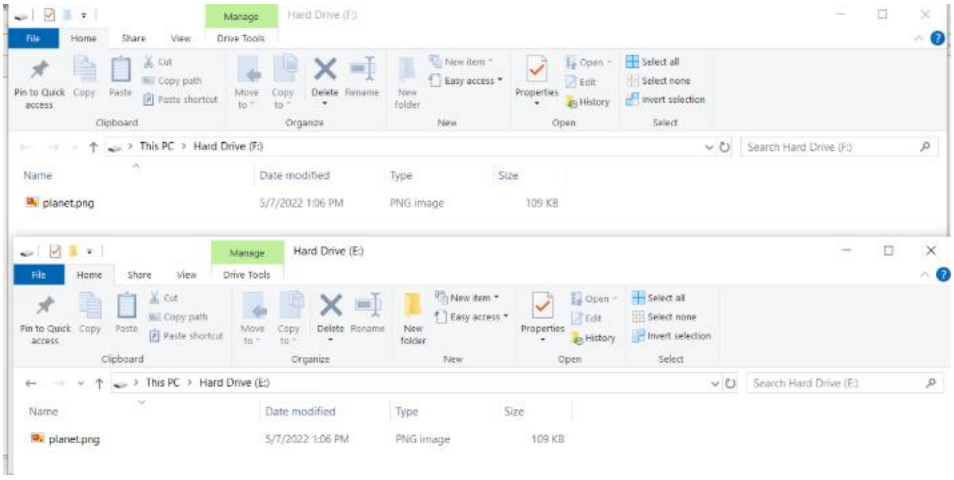
- Mounted the image as Read-Only with the drive letter F:.



- Opened File Explorer and confirmed the mounted drive was visible.

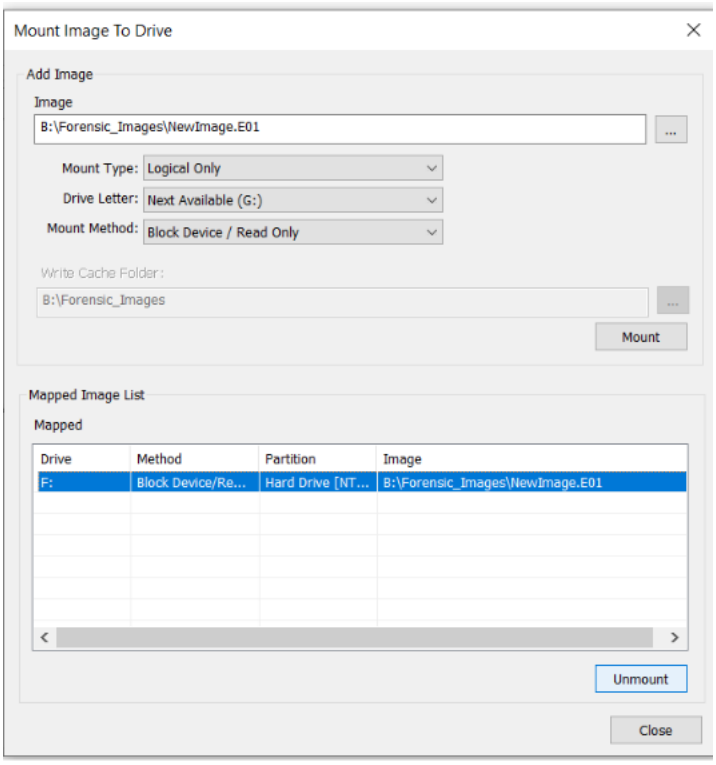
Step 6: Access the Mounted Image

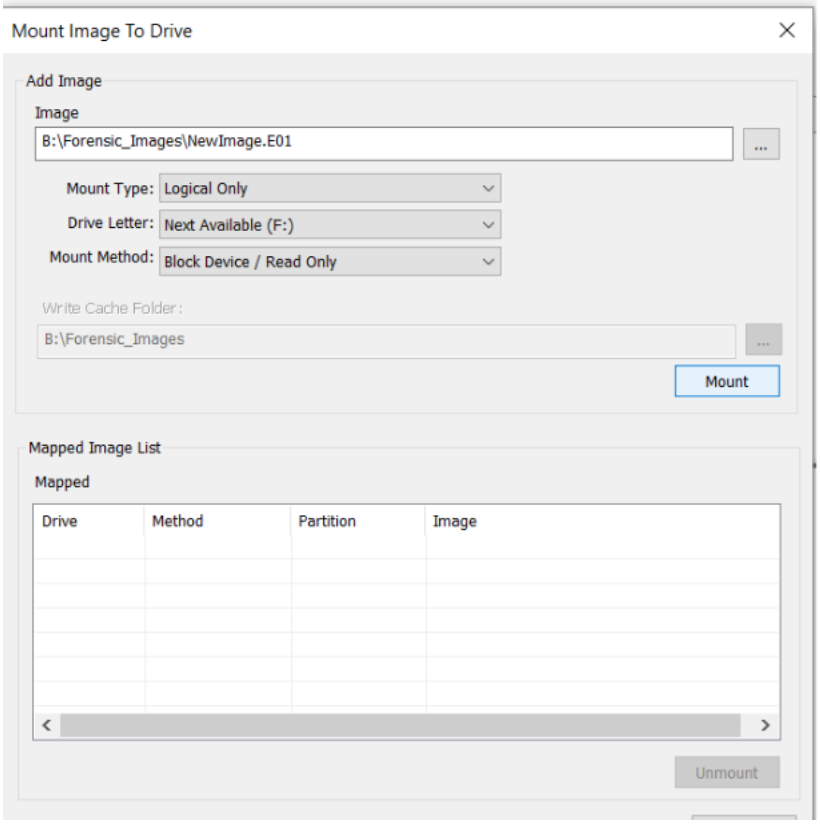
- Browsed the contents of the mounted drive through File Explorer.
- Verified files matched the original USB contents.



Step 7: Dismount the Image

- Returned to File → Image Mounting in FTK.
- Selected the mounted image and clicked Dismount.
- Confirmed the dismount; the drive letter disappeared from File Explorer.


Mount Image To Drive
Add Image
Image
B:\Forensic_Images\NewImage.E01
...
Mount Type: Logical Only
Drive Letter: Next Available (G:)
Mount Method: Block Device / Read Only
Write Cache Folder:
B:\Forensic_Images
...
Mount
Mapped Image List
Mapped
Drive
Method
Partition
Image
F:
Block Device/Re...
Hard Drive [NT...
B:\Forensic_Images\NewImage.E01
Unmount
Close


Mount Image To Drive
Add Image
Image
B:\Forensic_Images\NewImage.E01
...
Mount Type: Logical Only
Drive Letter: Next Available (F:)
Mount Method: Block Device / Read Only
Write Cache Folder:
B:\Forensic_Images
...
Mount
Mapped Image List
Mapped
Drive
Method
Partition
Image
Unmount

5. Conclusion

This lab demonstrated the use of FTK Imager to create, mount, and dismount forensic images. The process confirmed that FTK produces verified forensic copies using hashing to maintain evidence integrity. Mounting the image allowed access in read-only mode, ensuring the original evidence was not altered. Dismounting completed the process safely, simulating real-world forensic handling of digital evidence.

6. Challenges and Observations

- The progress bar sometimes did not update immediately; this was resolved by waiting for FTK to initialize.
- Remembering to choose a fragment size under 4 GB was important for compatibility.
- Running FTK as Administrator was required to access the drive fully.
- Logical imaging of small files (3 GB) was extremely fast compared to physical imaging of an entire disk.

8. References

- AccessData FTK Imager User Guide
- NIST Computer Forensics Tool Testing Program
- Microsoft Docs: File Systems and Hashing

Q&A (based on lab)

1. **What image format did you use?**
   E01 (Expert Witness Format).
2. **What hash algorithms were used to verify the image?**
   MD5 and SHA1.

3. **What drive letter was assigned when mounting the image?**
   F: (Read-Only).
4. **Why is FTK Imager important in digital forensics? Provide an example.**
   FTK Imager is important because it allows forensic investigators to create verified, tamper-proof copies of digital evidence. For example, if investigators seize a suspect's USB drive, they can create an E01 image with FTK, verify it with hashes, and analyze the mounted copy while leaving the original untouched for court evidence.

Lab Number: 05

Lab Title: Enabling and Disabling USB Write Protection in Windows

Date: 9/26/2025

1. Introduction

This lab focused on enabling and disabling USB write protection in Windows using the Registry Editor. The objective was to simulate the functionality of a write blocker, preventing data from being altered on a USB drive. Write blocking is an essential forensic practice that ensures evidence cannot be tampered with while being accessed.

2. Lab Objectives

- By completing this lab, I was able to:
- Enable write protection on a USB drive via the Windows Registry.
- Confirm that write attempts were blocked.
- Disable write protection to restore normal write access.

3. Tools and Requirements

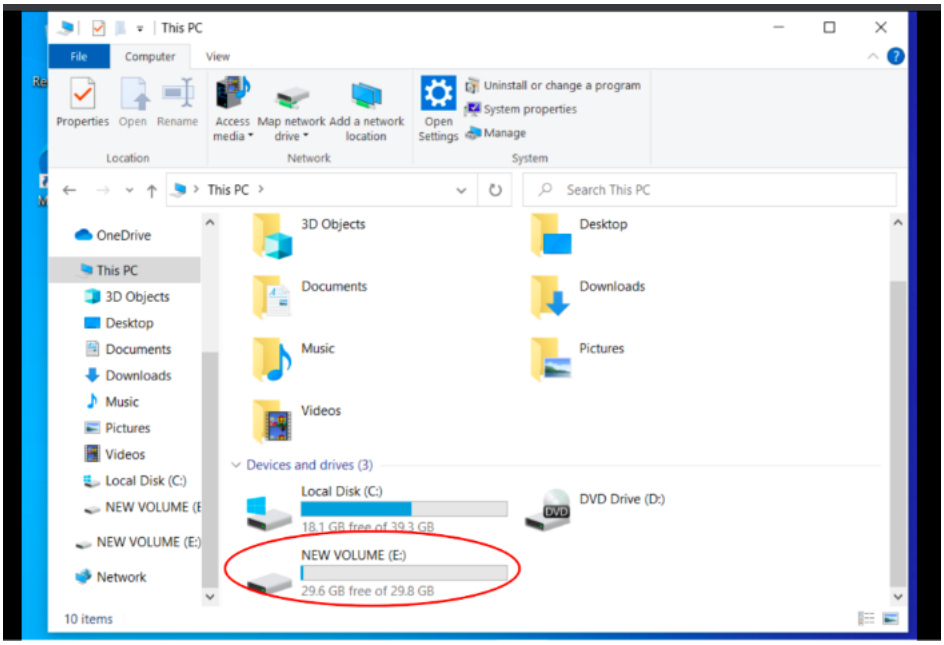
- **Operating System:** Windows 10
- **Hardware:** USB flash drive

- **Software:** Windows Registry Editor (regedit)

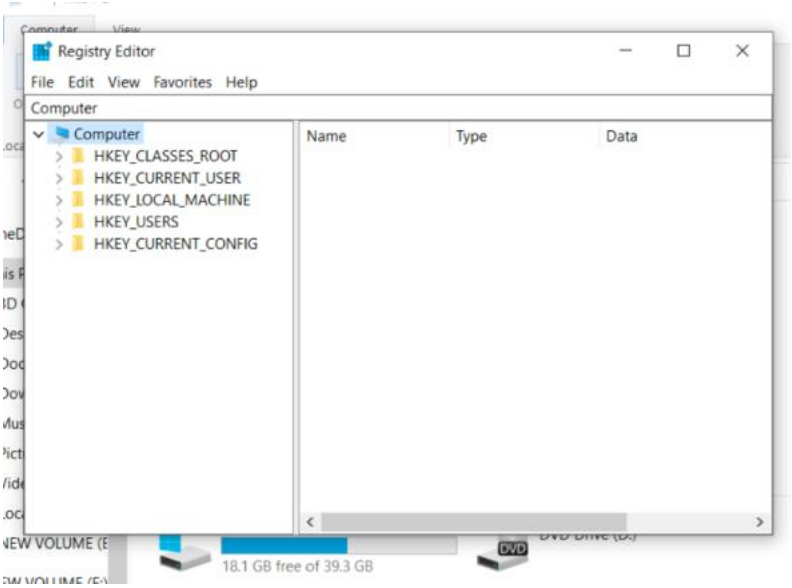
4. Lab Steps and Execution

Part 1 – Enable USB Write Protection

1. Inserted USB flash drive.

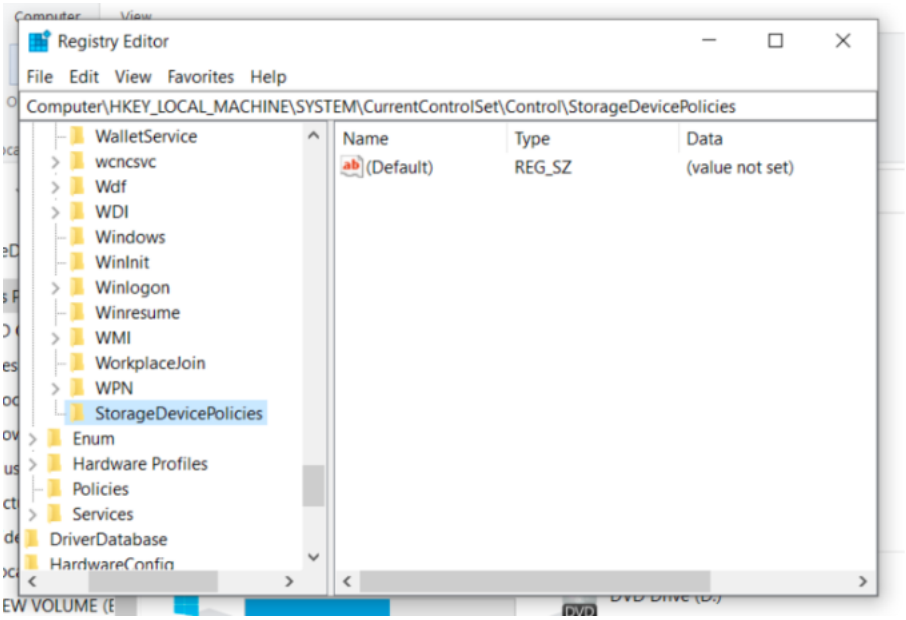


2. Opened **Registry Editor** (regedit).

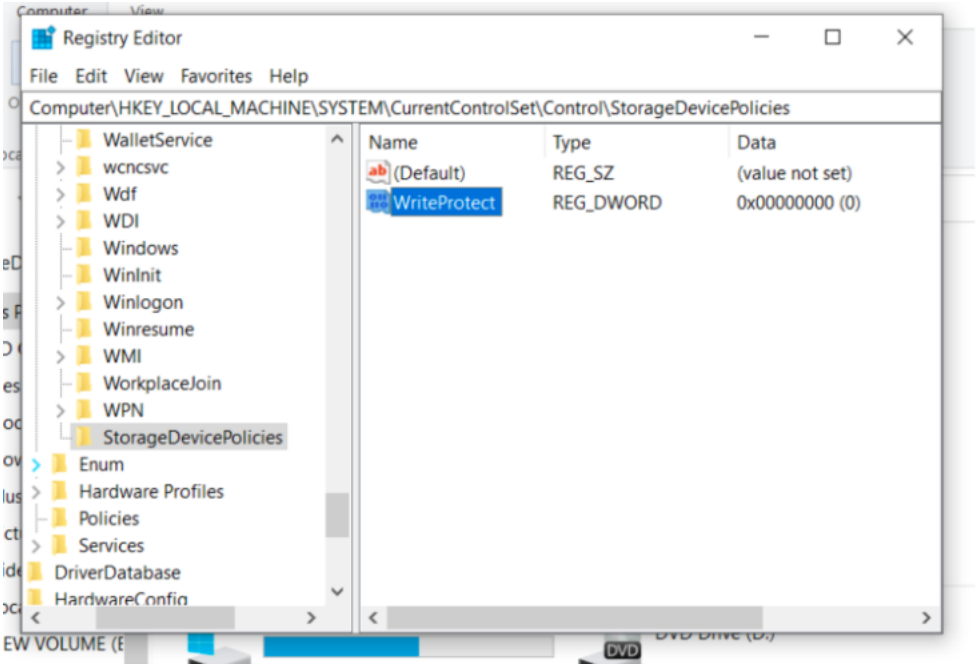


3. Navigated to:

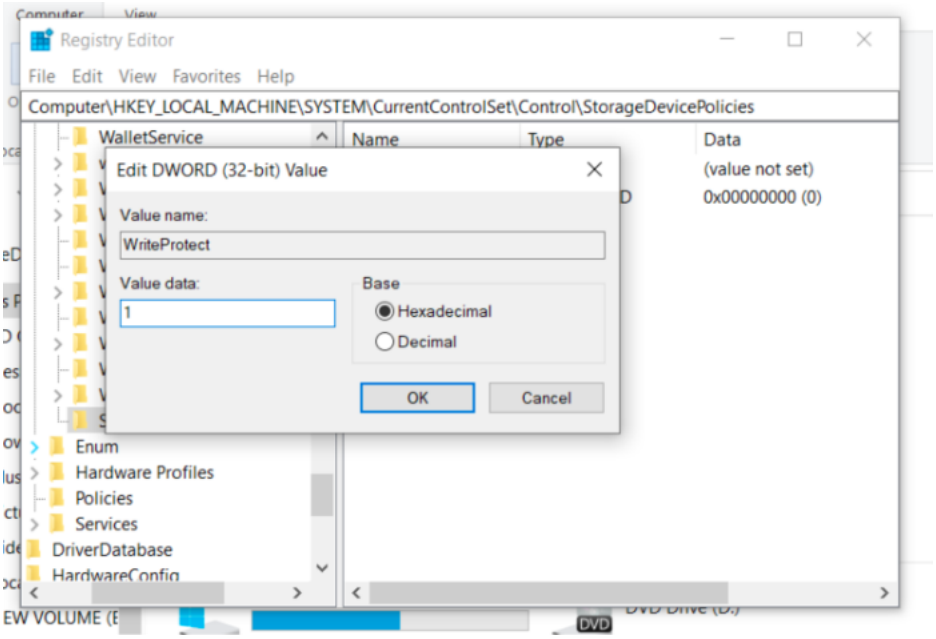
   HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\StorageDevicePolicies



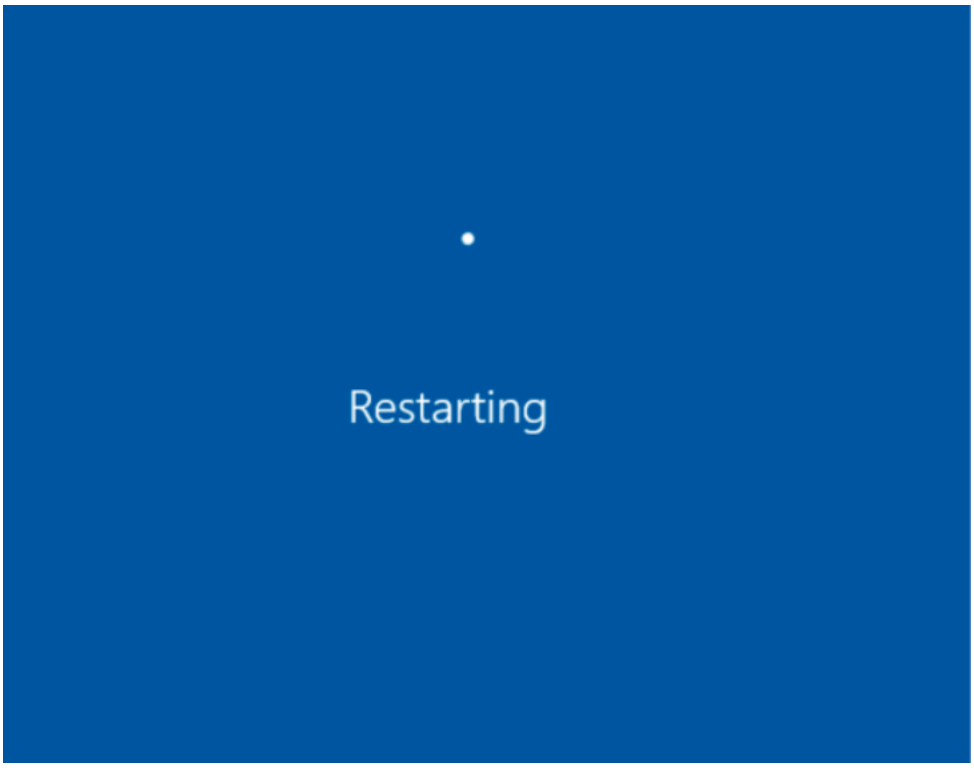
4. Created a new key (StorageDevicePolicies) and DWORD value WriteProtect if not already present.



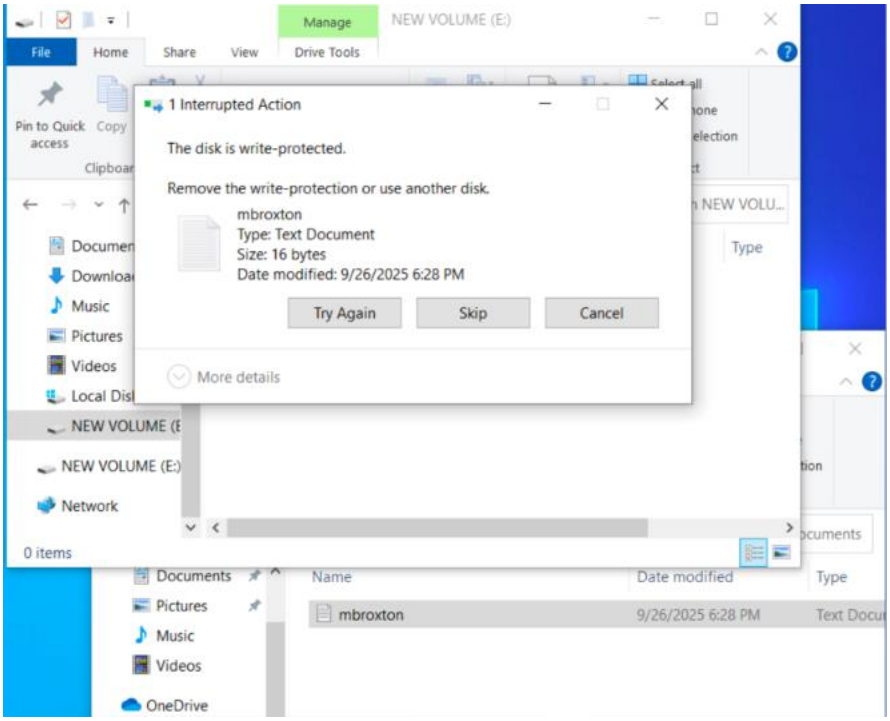
5. Set WriteProtect = 1 to enable write protection.



6. Restarted the computer.

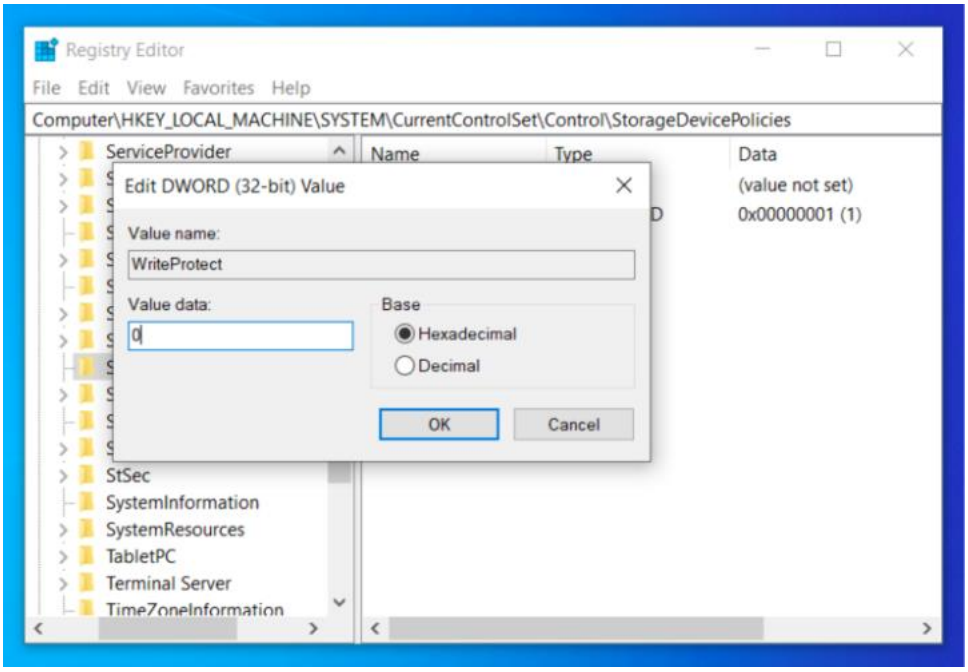


7. Attempted to copy a file to the USB drive. An error appeared indicating the disk was write-protected.

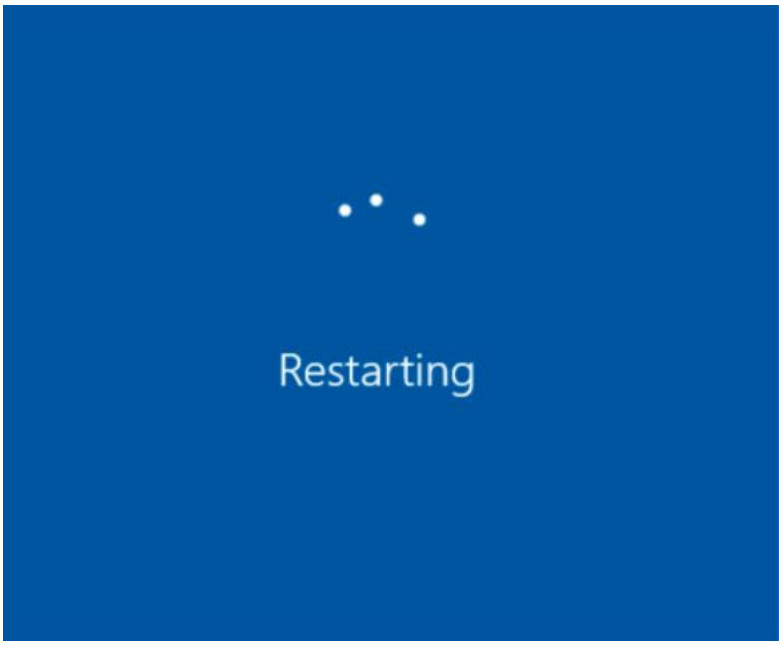


Part 2 – Disable USB Write Protection

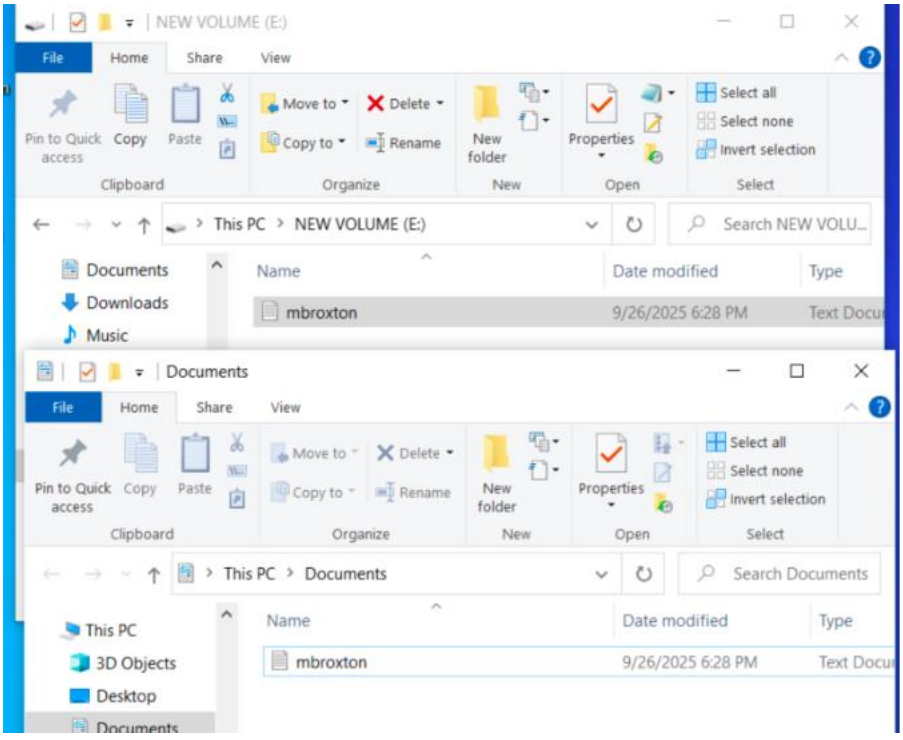
1. Reopened **Registry Editor**.
2. Navigated to the same location.
3. Set WriteProtect = 0 to disable write protection.



4. Restarted the computer.



5. Attempted to copy a file to the USB drive again. This time the file copied successfully.



5. Conclusion

This lab demonstrated how to use the Windows Registry Editor to simulate a write blocker. Enabling write protection prevented new files from being written to the USB device, while disabling it restored normal operation. It also showed how software-based write protection works at the system level.

6. Challenges and Observations

- The **StorageDevicePolicies** key did not exist initially and had to be created.
- A system restart was necessary for changes to take effect.
- The error message when attempting to write clearly indicated protection was working.
- Unlike hardware write blockers, this method can be bypassed if someone modifies the registry again.

7. References

- Microsoft Documentation: Registry Editor Overview
- Course Handout: Write Blocker Lab Instructions

Lab Number: 06

Lab Title: Extracting and Mapping GPS Data

Date: 10/03/2025

1. Introduction

This lab focused on extracting GPS metadata from digital photographs using ExifTool. Metadata embedded in photos can include location, time, and device details. Forensic investigators often use this information to determine where and when a photo was taken.

2. Lab Objectives

- Use ExifTool to extract metadata, including GPS coordinates, from an image.
- Convert GPS coordinates from Degrees-Minutes-Seconds (DMS) to decimal format.
- Use Google Maps to locate where the photo was taken.
- Demonstrate privacy awareness by removing EXIF metadata.

3. Tools and Requirements

- **Device Used:** Smartphone (photo source)
- **Computer OS:** Windows 10
- **Software:** ExifTool
- **Additional Tool:** Google Maps

4. Lab Steps and Execution

Step 1 – Enable Location Services and Take Photo

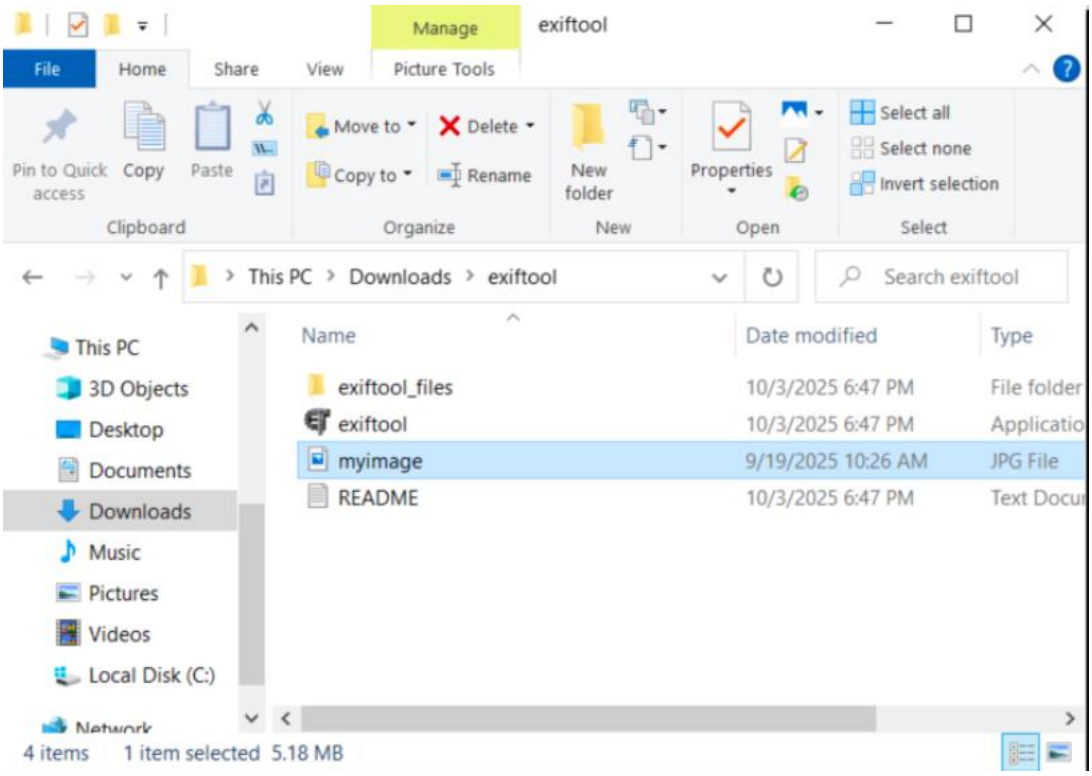
- Enabled Location Services on smartphone camera.
- Took a photo

Step 2 – Transfer Photo

- Transferred the photo to the computer via USB.

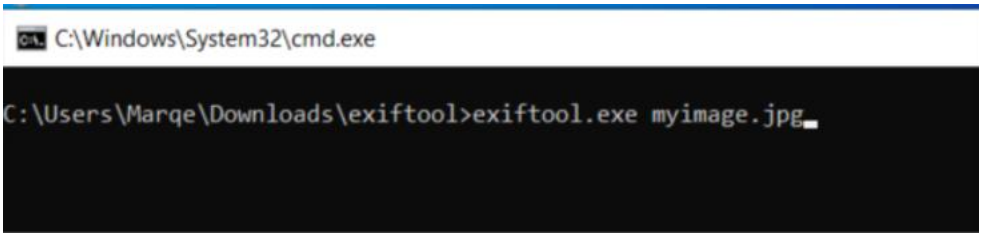
Step 3 – Install ExifTool

- Downloaded ExifTool for Windows.
- Renamed exiftool(-k).exe to exiftool.exe for easier use.
- Placed both ExifTool and the photo in the same folder.

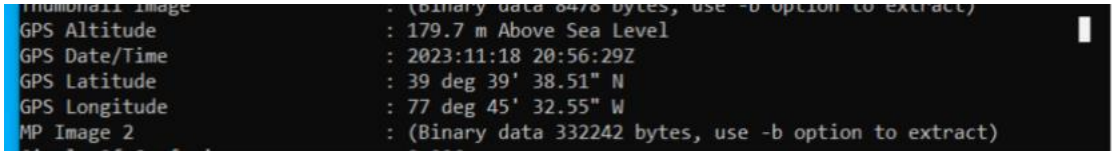


Step 4 – Extract Metadata

- Opened Command Prompt, navigated to the photo folder.
- Ran: exiftool.exe myimage.jpg



- Located **GPS Latitude** and **GPS Longitude** in the output.



Step 5 – Convert Coordinates

- GPS data was in DMS format.
- Converted to decimal degrees using:

Decimal = Degrees + (Minutes / 60) + (Seconds / 3600)

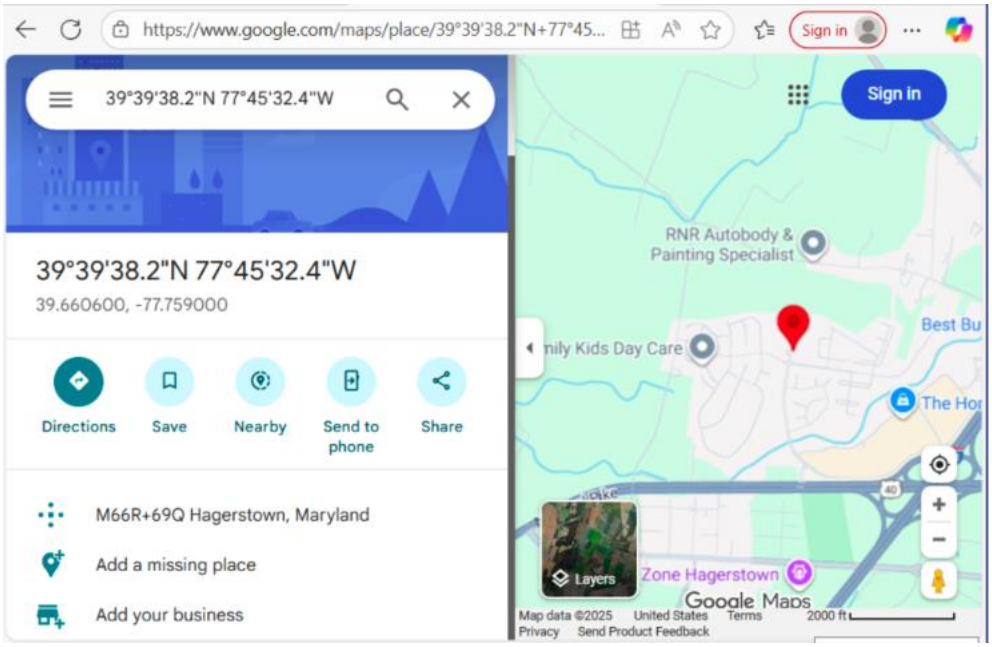
Latitude: 39 + (39 / 60) + (38.51 / 3600) = 39.6606972 N

Longitude: 77 + (45 / 60) + (32.55 / 3600) = -77.7590417 W

Changed the Longitude to a negative decimal since it is West.

Step 6 – Map the Coordinates

- Opened Google Maps.
- Entered converted coordinates: 39.6606972, -77.7590417
- Pin dropped on the correct location.

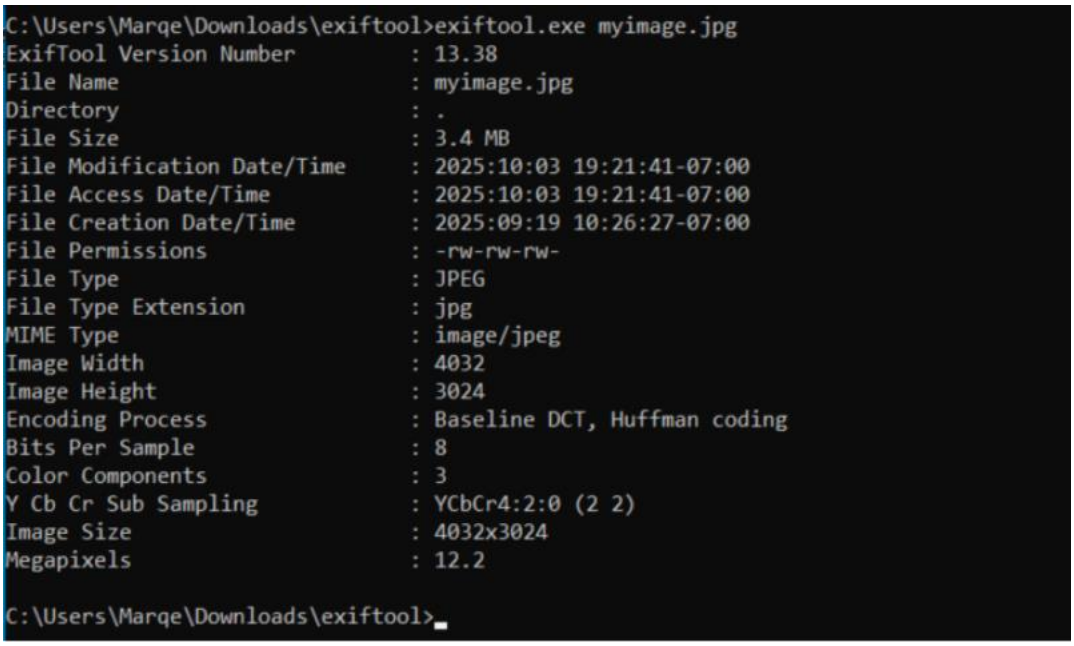


Step 7 – Privacy Awareness

- Removed all EXIF metadata by running:

  exiftool.exe -all= -overwrite_original myimage.jpg

  Re-ran ExifTool and confirmed GPS data was gone.



5. Conclusion

This lab demonstrated how ExifTool can be used to extract GPS metadata from images, convert coordinates to decimal format, and map locations in Google Maps. This process is useful in digital forensics, where photo metadata can provide crucial evidence of where and when a picture was taken. Additionally, the optional step highlighted privacy risks and showed how metadata can be stripped to protect personal information.

6. Challenges and Observations

- Initially, remembering the exact ExifTool command syntax required careful attention.
- Conversion from DMS to decimal format had to account for negative longitude values (west of the Prime Meridian).
- Google Maps accurately pinpointed the location using converted coordinates.
- The metadata removal step reinforced the importance of privacy awareness.

7. References

- ExifTool Official Website: https://exiftool.org/
- Google Maps: https://maps.google.com

Lab Number: 07

Lab Title: Hashing Files on Windows Using Febooti Hash & CRC

Date: 10/16/2025

1. Introduction

This lab focused on using the Febooti Hash & CRC utility to generate and compare hash values for files in Windows. The objective was to understand how file integrity can be verified by hashing and how even a small change in a file result in a completely different hash value.

2. Lab Objectives

- Generate MD5 and SHA-256 hash values for files using Febooti Hash & CRC.
- Verify file integrity by comparing original and modified file hashes.
- Demonstrate that any change to a file produces a new hash, confirming tamper detection in digital forensics.

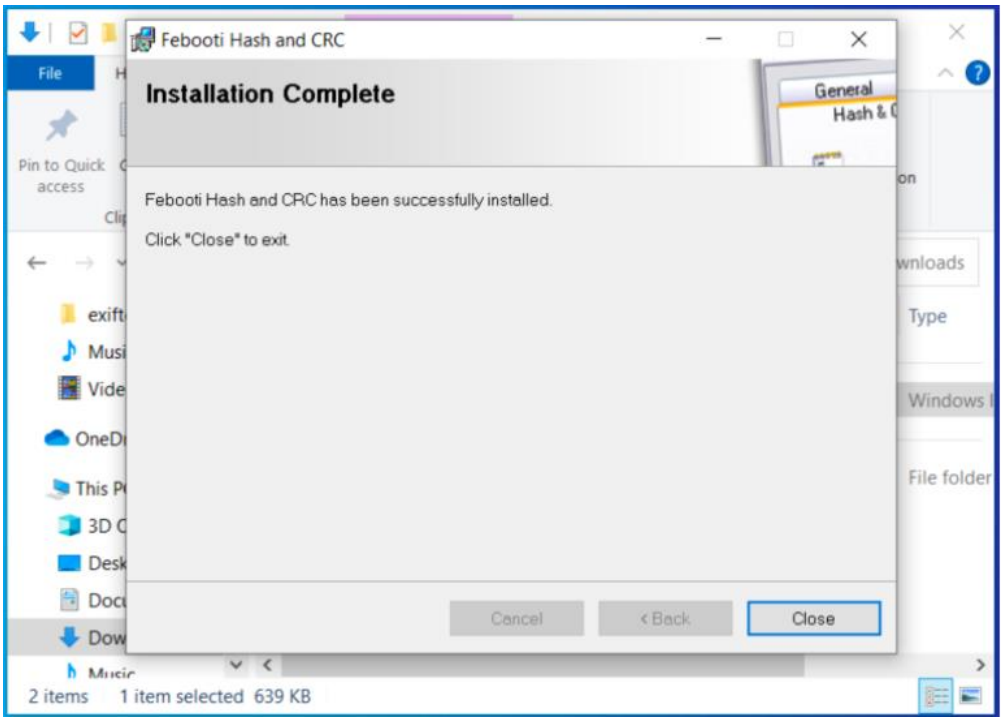
3. Tools and Requirements

- Operating System: Windows 10
- Software: Febooti Hash & CRC
- File Used: hash_test.txt

4. Lab Steps and Execution

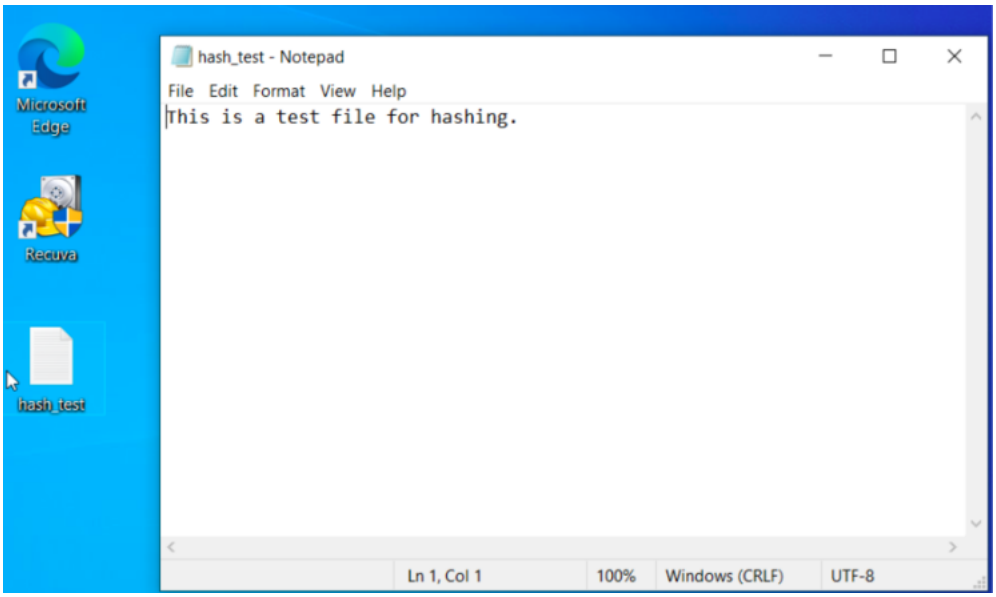
Step 1 – Install Febooti Hash & CRC

1. Downloaded Febooti Hash & CRC from the official site: https://www.febooti.com/products/filetweak/members/hash-and-crc/.
2. Installed using the default setup wizard.
3. Confirmed installation by right-clicking on a file and verifying the “File Hash & CRC” context menu option appeared.

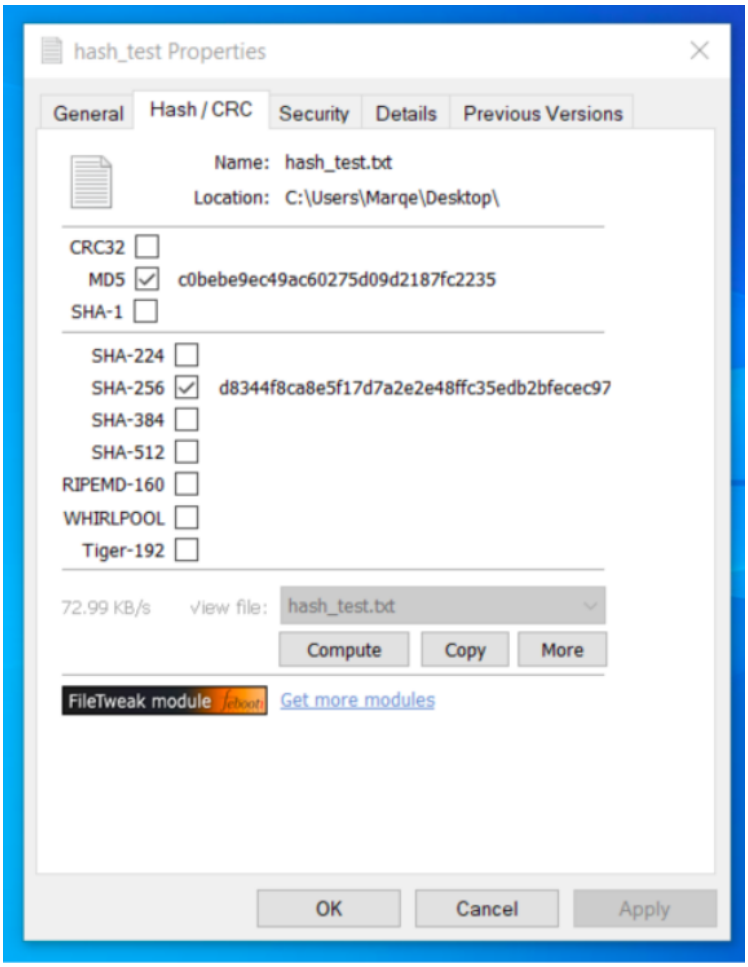


Step 2 – Generate Hash for Original File

1. Opened Notepad and created a file with the text:
   This is a test file for hashing.
2. Saved it as hash_test.txt on the Desktop.



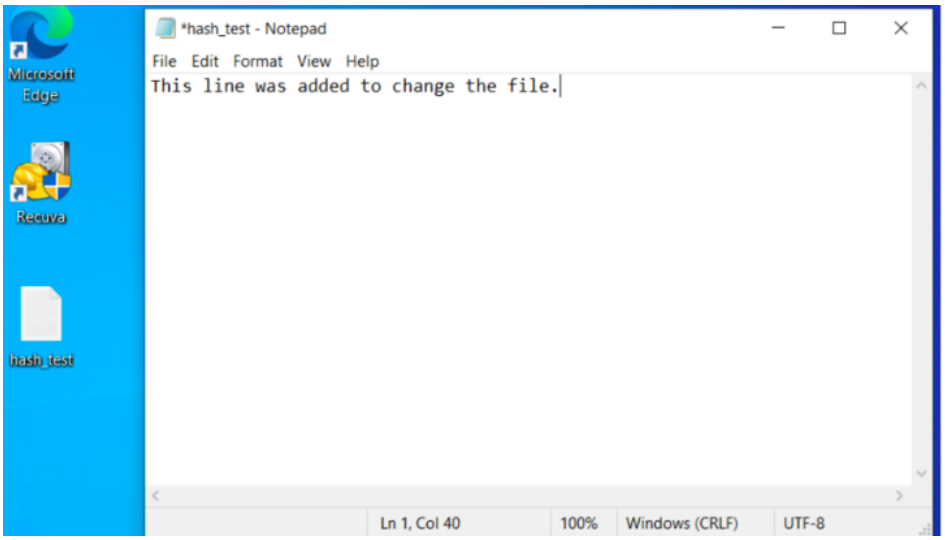
3. Right-clicked the file → Properties “Hash/CRC” → generated MD5 and SHA-256 hashes.



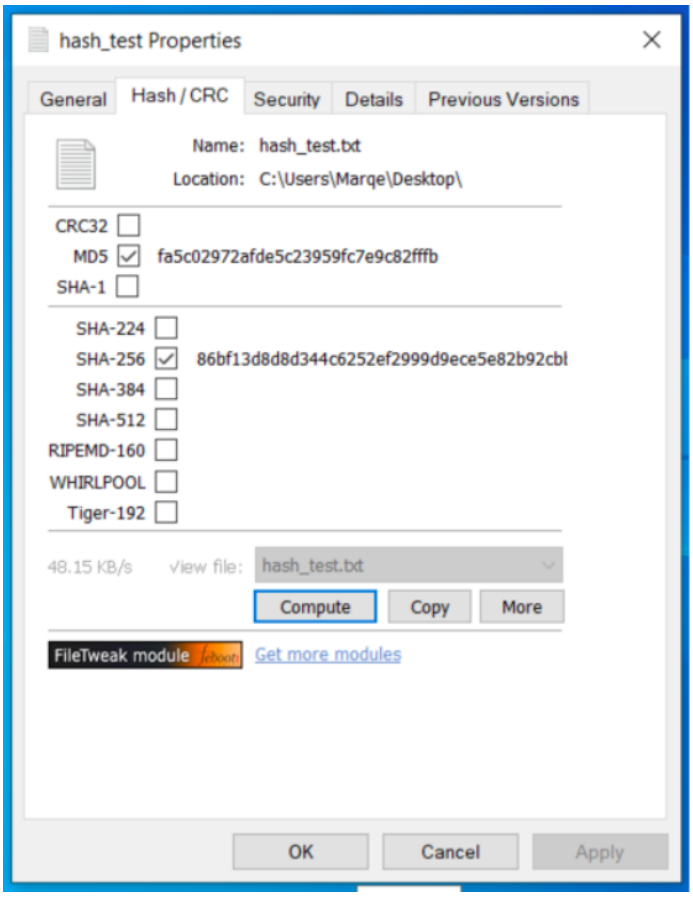
Step 3 – Modify File and Compare Hashes

1. Reopened hash_test.txt in Notepad.
2. Added the line:

   This line was added to change the file.



3. Saved the file again.
4. Right-clicked → Properties “Hash/CRC” → regenerated MD5 and SHA-256 hashes.



Step 4 – Analyze Results

- Compared the two sets of hashes from before and after modification.
- Both MD5 and SHA-256 values were completely different after changing one line.
- This property is known as the avalanche effect, meaning small modifications in the input result in large, unpredictable changes in the hash output.

5. Conclusion

This lab demonstrated how hashing ensures data integrity and authenticity in digital forensics. Using Febooti Hash & CRC, I verified that even any character change alters the entire hash value. This proves that hashing is a reliable way to detect tampering or corruption in digital evidence.

6. Challenges and Observations

- Remembering to select both MD5 and SHA-256 required double-checking the settings.
- The difference in hashes clearly illustrated the sensitivity of hashing algorithms.
- Febooti's context menu integration made it easy to hash files directly without command-line tools.
- The lab reinforced that hashing is fundamental to forensic validation.

8. References

- Febooti Hash & CRC: https://www.febooti.com/products/filetweak/members/hash-and-crc/
- NIST Hash Function Validation Documentation

Lab Number: 08

Lab Title: Extracting Metadata from an MS Office File

Date: 10/23/2025

1. Introduction

This lab focused on using file properties and metadata to identify hidden information in Microsoft Office documents. Metadata can reveal when a file was created, who created it, when it was last modified, and other embedded information useful in forensic analysis. The objective was to analyze how metadata changes when a document is edited, saved, or downloaded again, and to understand how timestamps can provide insight into user activity.

2. Lab Objectives

By completing this lab, I was able to:

- Identify file metadata within Microsoft Word documents.
- Record and compare Date Created, Date Modified, and Author fields.
- Observe how metadata changes after modifying and saving a file.
- Explain why file creation and modification dates differ when downloading versus saving locally.
- Understand the forensic importance of file metadata in investigations.

3. Tools and Requirements

- **Operating System:** Windows 10
- **Software:** Microsoft Word or Word Online
- **Alternative Tools:** DOCX Editor (for local editing)
- **File Used:** `metadata_test.docx`

4. Lab Steps and Execution

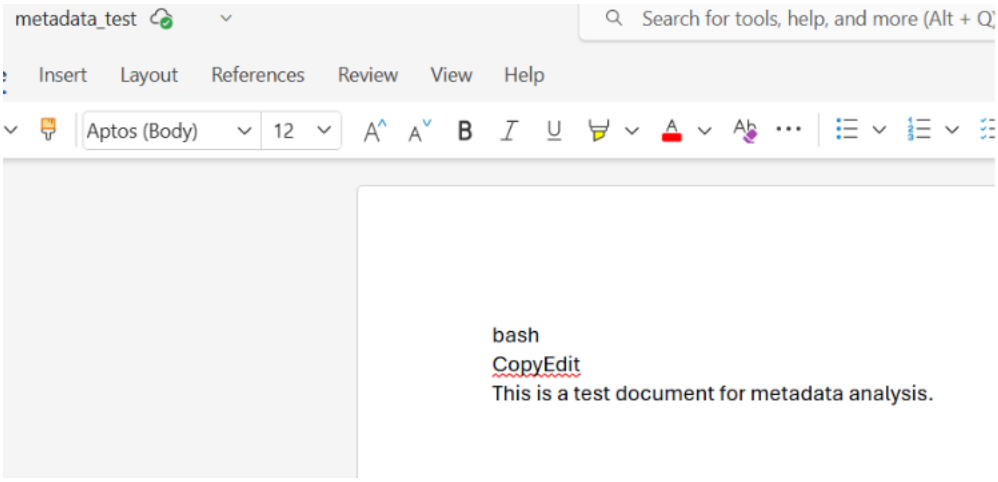
Step 1 – Create a Test Document

1. Opened Microsoft Word and created a new document.
2. Typed:

```
bash
CopyEdit
```

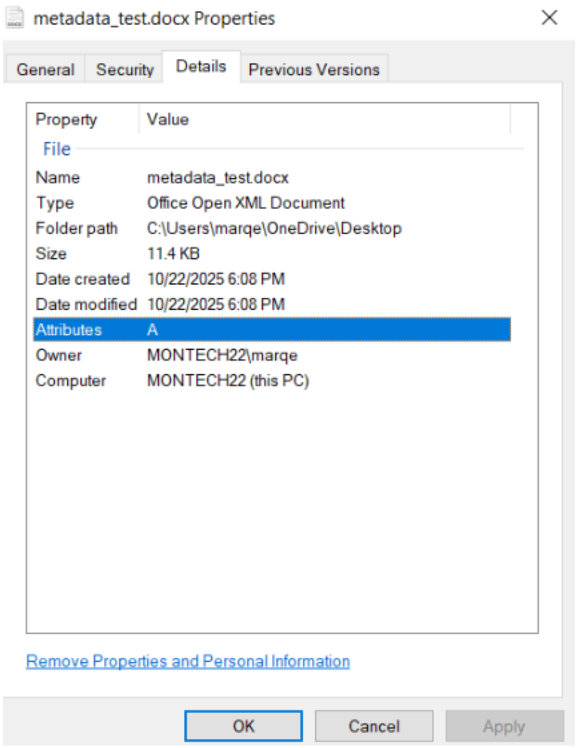
```
This is a test document for metadata analysis.
```

3. Saved the file as `metadata_test.docx`.



Step 2 – View Metadata in File Properties

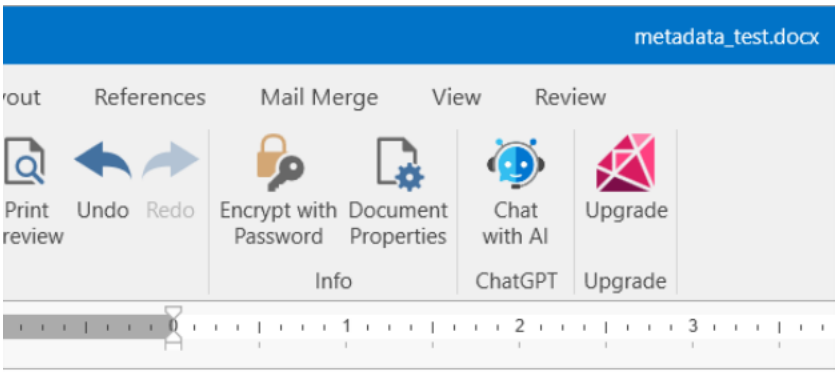
1. Checked file properties → Right-click → Properties → Details tab.
2. Recorded the following metadata values:
    a. Date Created
    b. Date Modified
    c. Author



Step 3 – Modify the Document and Check Metadata Again

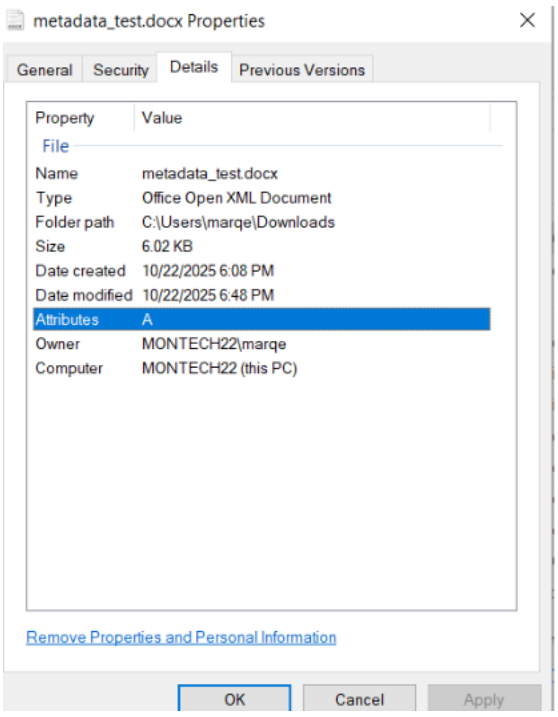
1. Opened the same file directly from the local drive.
2. Added a new line of text:

```
pgsql
CopyEdit
This is an updated version of the document.
```




```
pgsql
CopyEdit
This is an updated version of the document.
```

3. Saved the file.
4. Rechecked properties.
    a. The **Date Created** remained the same.
    b. The **Date Modified** updated to the current time.



Step 4 – Analyze Results

After creating and editing the Word document, the Date Modified field updated to the exact time the file was saved again, while the Date Created stayed the same. This confirms that Microsoft Word updates the modification timestamp every time a file is changed, but the original creation date is preserved as long as the file remains in the same location.

If the file is downloaded or saved to a different location, Windows assigns a new creation date, because the system treats it as a new instance of the file being written to the disk.

In forensic, this demonstrates the importance of understanding what each timestamp means:

Date Created – When the file first appeared on the storage device.

Date Modified – When the file content was last altered.

Date Accessed – When the file was last opened or viewed.

Knowing how these timestamps behave helps investigators determine when a file was first created, altered, or accessed, making it crucial for building a timeline in an investigation.

5. Conclusion

This lab demonstrated how Microsoft Office documents contain valuable metadata that records authorship, creation, and modification details. In forensic analysis, these timestamps can reveal when and how a file was created or tampered with. Maintaining proper chain of custody and avoiding unnecessary re-downloads ensures the original metadata remains intact for evidentiary purposes.

6. Challenges and Observations

- Using Word Online causes automatic file duplication (e.g., `metadata_test (1).docx`).
- The Date Created changed every time the document was re-downloaded.
- To keep timestamps consistent, the file must be edited and saved on the same system instead of re-downloaded.

- Understanding how metadata behaves helps distinguish between original and copied evidence.

7. References

- Microsoft Documentation: View or Change the Properties for an Office File
- NIST: Guide to Integrating Forensic Techniques into Incident Response

Lab Number: 09
Lab Title: Windows Forensic Artifacts
Date: 10/23/2025

1. Introduction

This lab focused on identifying and analyzing Windows forensic artifacts that attackers may leave behind on a compromised system. These artifacts can provide valuable evidence of persistence, program execution, and user activity. The goal was to use built-in Windows tools to locate potential malicious entries, such as startup items, recent file activity, and prefetch data, which together can help investigators reconstruct what occurred on a system after an intrusion.

2. Lab Objectives

By completing this lab, I was able to:

- Examine Windows Registry entries that show programs configured to run automatically at startup.
- Analyze the Recent Files folder to identify files that were recently accessed or executed.
- Review Prefetch files to determine which programs were run and when.
- Interpret what each artifact reveals about system or attacker activity.

3. Tools and Requirements

- **Operating System:** Windows 10
- **Tools Used:**
    - Registry Editor (`regedit`)
    - File Explorer
    - Notepad

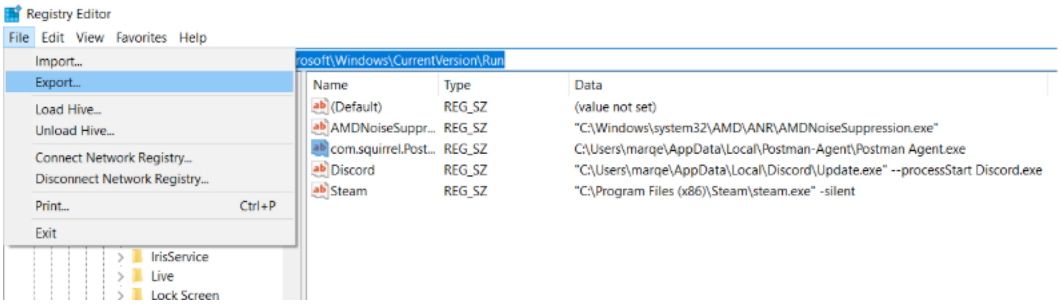
4. Lab Steps and Execution

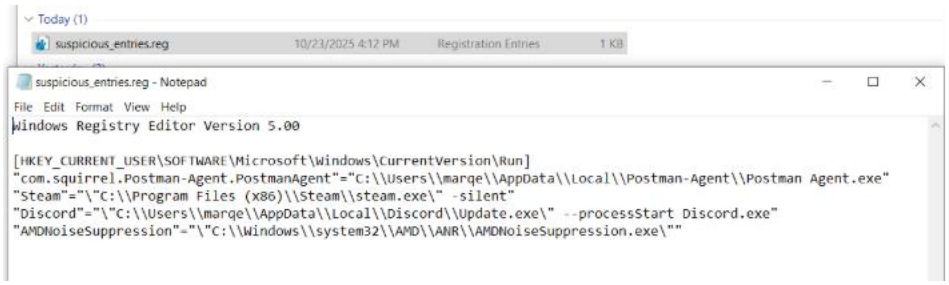
Step 1 – Investigate Auto-Start Entries (Registry)

1. Opened Registry Editor (regedit).
2. Navigated to the following key:

`HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run`

3. Reviewed each entry for unusual program names or file paths, such as those running from AppData, Temp, or user folders.
4. Found one suspicious entry:
    a. **Name:** `Postman Agent`
    b. **Path:** `C:\Users\marqe\AppData\Local\Postman-Agent\Postman Agent.exe`
5. Exported the entry and opened it in Notepad to confirm the executable path.



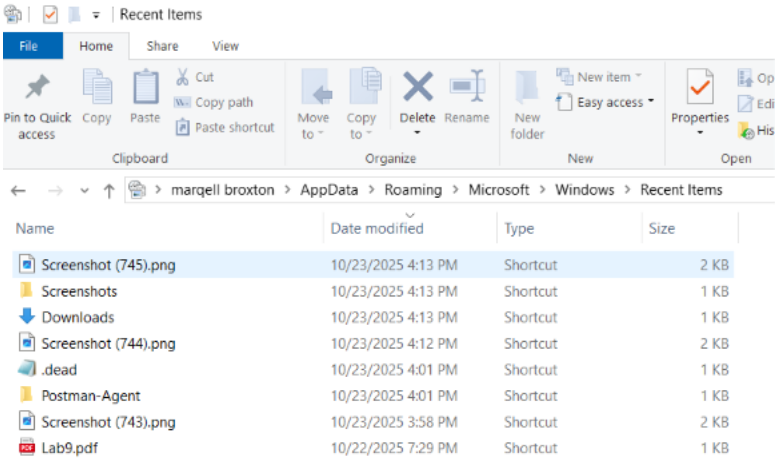


Step 2 – Check Recently Accessed Files

1. Opened File Explorer and navigated to:

`C:\Users\marqe\AppData\Roaming\Microsoft\Windows\Recent`

2. Reviewed the contents for recently accessed files.
3. Identified several items referencing temporary or user-created folders, suggesting potential script or tool execution.



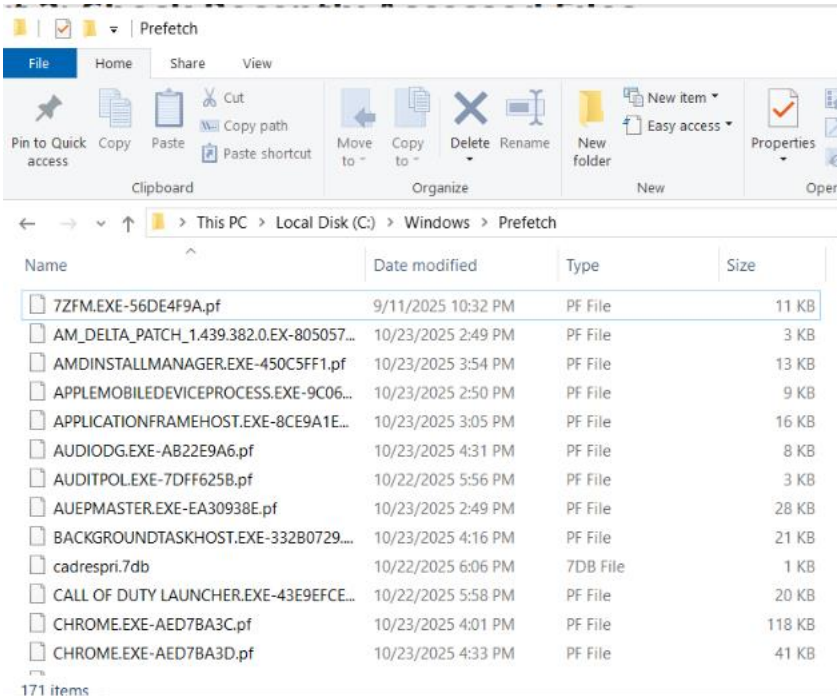
Step 3 – Examine Prefetch Files

1. Navigated to the Prefetch directory:

   `C:\Windows\Prefetch`

2. Observed multiple `.pf` files indicating programs that had run recently.
3. Identified one unusual file:

ADMININSTALLMANAGER.EXE-450C5FF1.pf



4. Analyze Results

The artifacts collected from the registry, recent file list, and prefetch directory all pointed to potentially malicious activity.

- The Registry Run key can show an attacker configured in their own malware to automatically start each time the user logged in.
- The Recent Files folder may reveal traces of files accessed by the user, or malicious scripts executed by an attacker.
- The Prefetch file will confirm actual program execution, proving that a .exe file ran on the system and was not simply placed there.

Together, these artifacts provide a clear timeline of compromise:

1. The malicious file was introduced and executed.
2. The program created persistence via the Run key.
3. Prefetch and recent file data confirmed repeated or recent execution.

6. Conclusion

This lab showed how forensic investigators can identify traces of malicious activity using native Windows artifacts. By reviewing the Registry, Recent folder, and Prefetch files, it's possible to reconstruct evidence of persistence, program execution, and user interaction. These data sources are essential for timeline reconstruction and attribution during digital investigations.

7. Challenges and Observations

- Some legitimate programs also run from AppData or Temp, so context and path verification are important.
- Prefetch files can only be viewed on systems with Prefetch enabled.
- Autoruns could be used as an additional verification tool for startup persistence.

8. References

- Microsoft Docs – Windows Registry and Autostart Locations
- SANS DFIR Poster: Windows Forensic Artifacts
- NIST SP 800-86: Guide to Integrating Forensic Techniques into Incident Response

Lab Number: 10
Lab Title: Hard Links vs. Soft (Symbolic) Links
Date: 10/31/2025

1. Introduction

This lab explores how Linux uses **hard links** and **soft (symbolic) links** to reference files. Understanding link behavior is important in digital forensics because link structures can hide, reference, or preserve data even after the original file appears deleted. By working with inodes and link counts, I learned how Linux tracks file objects and how links behave when files are modified or removed.

2. Objectives

By completing this lab, I:

- Created and verified **hard links** and **soft links** in Linux
- Observed the inode values and link counts using ls -li
- Demonstrated how hard links preserve data after file deletion
- Showed how symbolic links fail once their target files are deleted
- Explained forensic relevance of link behavior

3. Tools Used

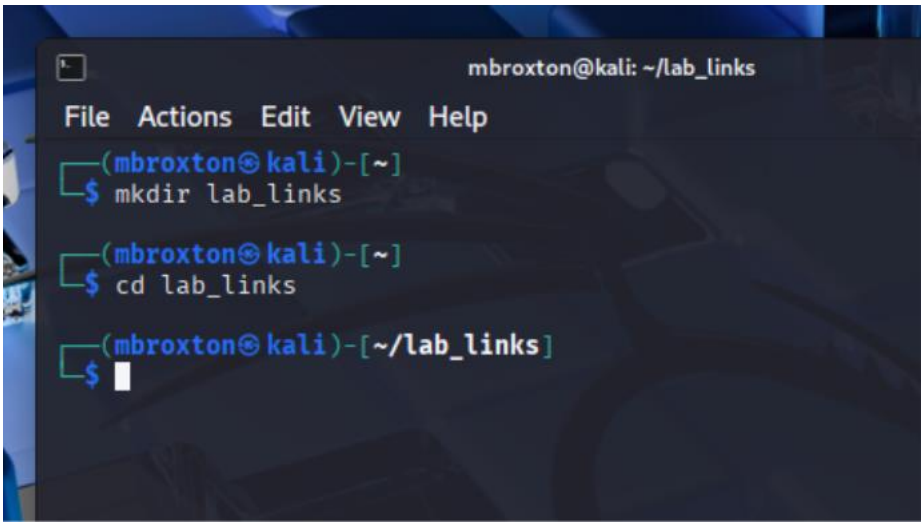
- **Operating System:** Linux (Ubuntu)
- **Terminal / Bash Shell**
- **Commands Used:**
  - ln (create hard & soft links)
  - ls -li (list files w/ inode info)
  - cat (display file contents)
  - rm (remove files)
- **Filesystem:** ext4 (default Linux filesystem)

4. Lab Procedure & Results

Part 1 – Hard Links

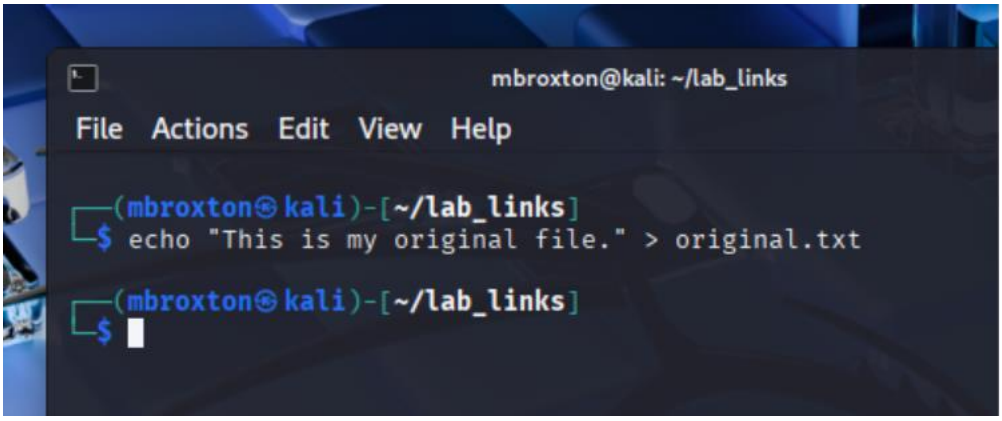
Step 1: Create directory

```
mkdir ~/lab_links && cd ~/lab_links
```



Step 2: Create original file

echo "This is my original file." > original.txt

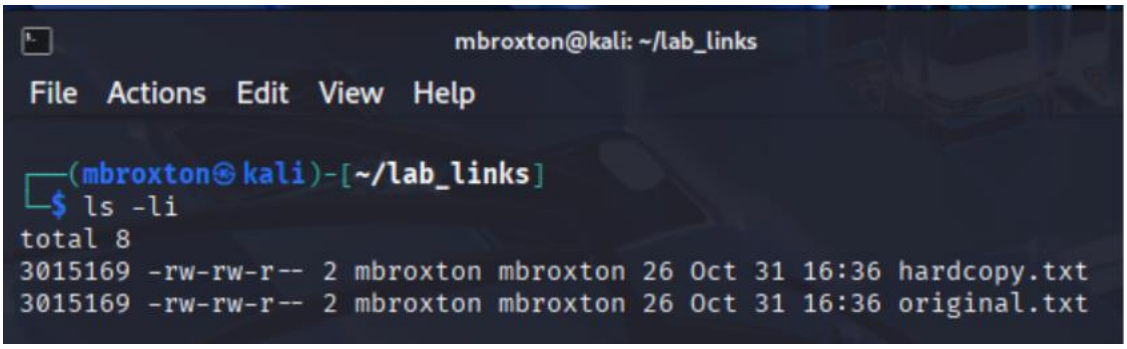


Step 3: Create hard link

ln original.txt hardcopy.txt

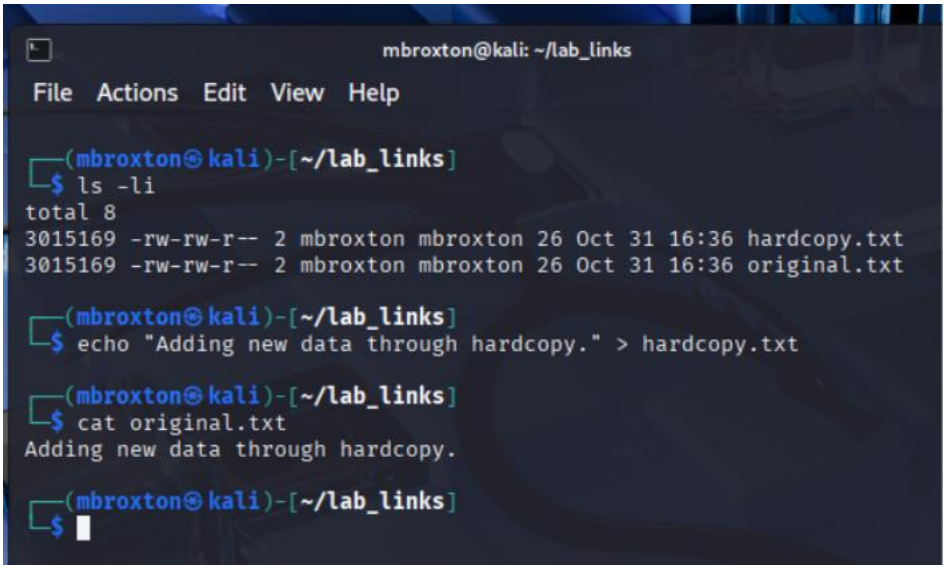
Step 4: Verify inodes

ls –li



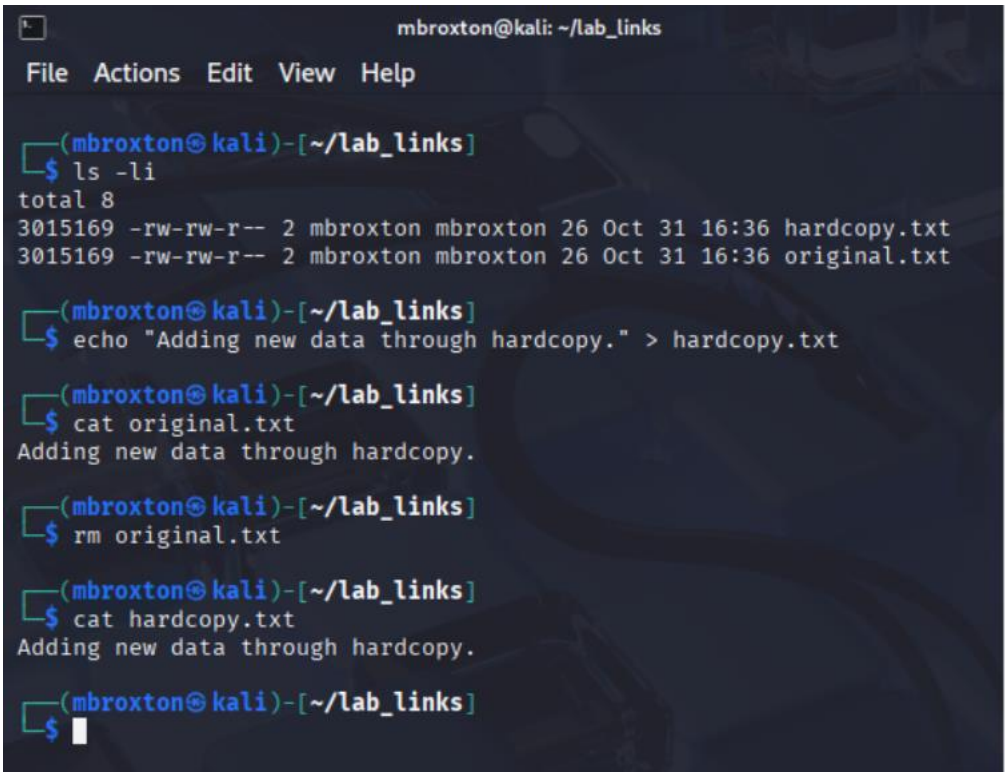
Step 5: Modify through hard link

echo "Adding new data through hardcopy." >> hardcopy.txt
cat original.txt



Step 6: Delete original file

rm original.txt
cat hardcopy.txt

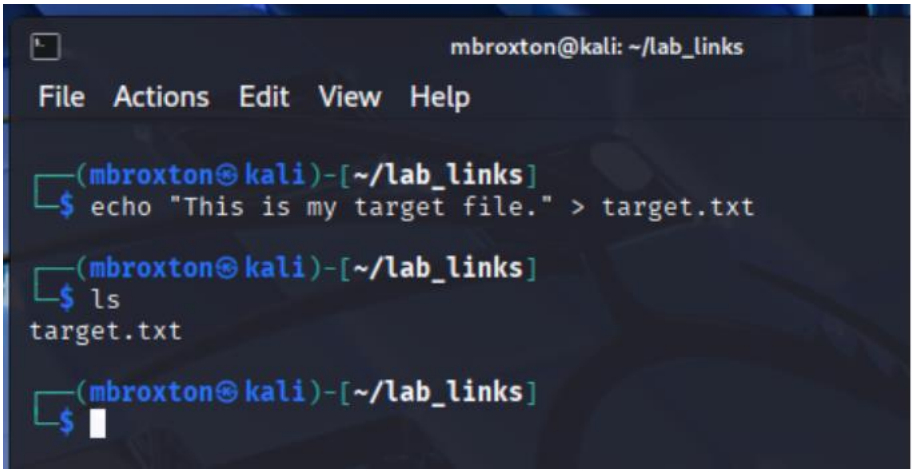


hardcopy.txt retained full content — data survived deletion.

Part 2 – Soft (Symbolic) Links

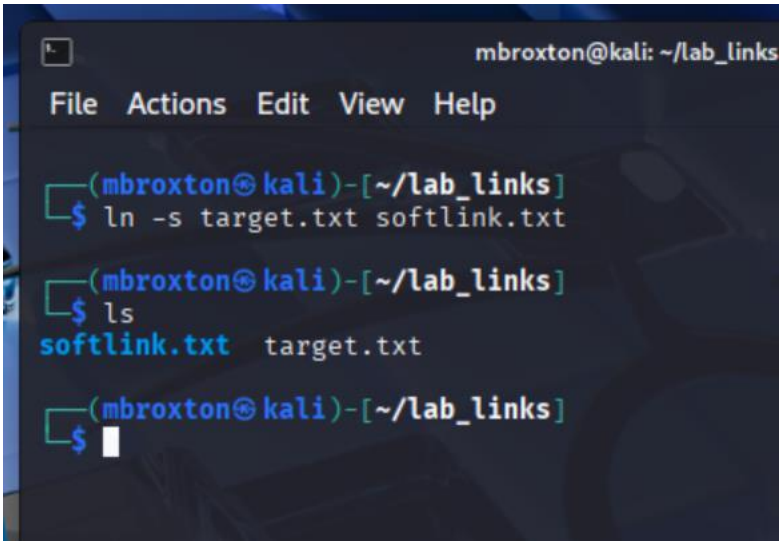
Step 1: Create new file

echo "This is my target file." > target.txt



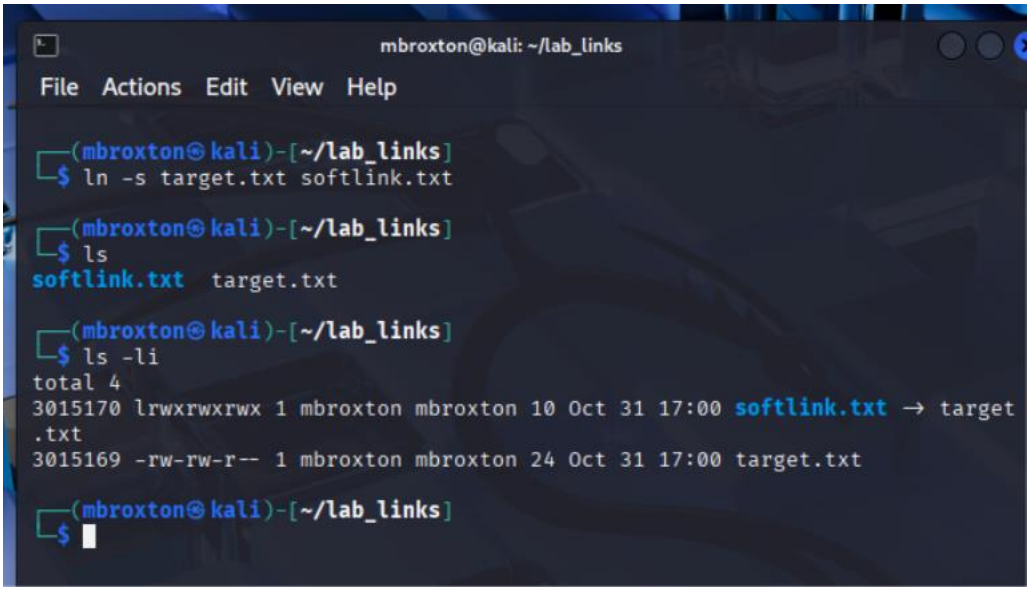
Step 2: Create symbolic link

ln -s target.txt softlink.txt



Step 3: Verify link

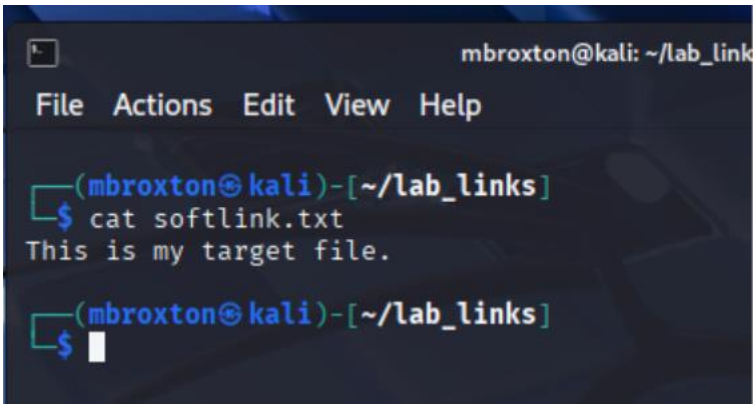
ls -li



Symbolic link displayed different inode and arrow → target.txt.

Step 4: Access through symlink

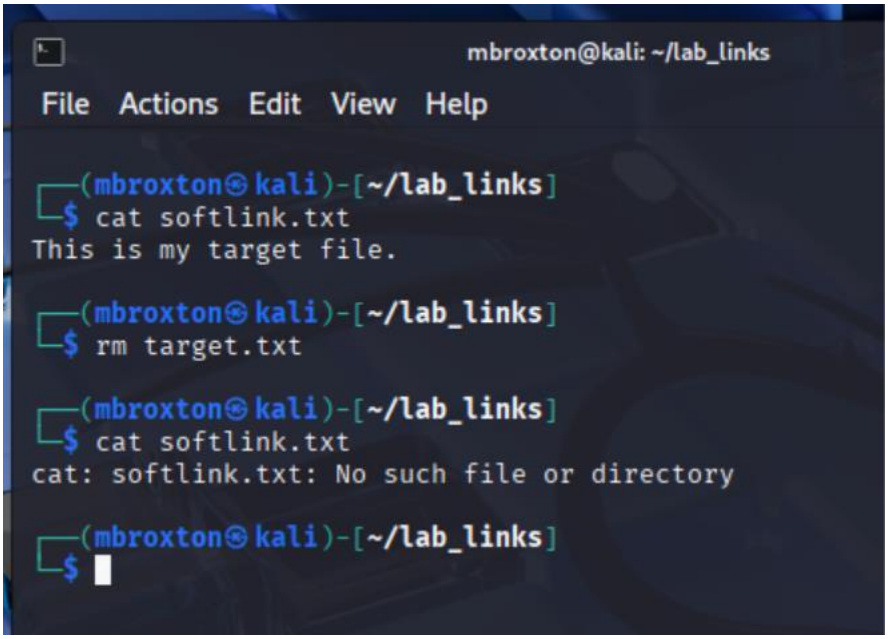
cat softlink.txt



Displayed same contents as target.txt.

Step 5: Break link

rm target.txt
cat softlink.txt



Output:

cat: softlink.txt: No such file or directory

5. Analyze Results

Hard links and symbolic links behave differently due to inode structure. Hard links share the same inode as the original file, meaning both names point to the same data on disk. When the original file was deleted, the data remained accessible through the hard link. This behavior makes hard links important in digital forensics, as a file may appear deleted but still exist through a hard link.

Symbolic links point to the file name and not the inode. When the original file was removed, the symbolic link broke because its path target no longer existed. Symbolic links are like shortcuts in Windows and do not preserve access once the original file is removed.

Understanding link behavior helps forensic analysts detect hidden or persistent files and recover data that may still exist even after deletion.

6. Conclusion

This lab demonstrated how Linux links work at the filesystem level. Hard links reference the **same inode**, meaning they remain valid even after the original file is deleted, which can preserve data and matter in forensic recovery. Symbolic links act as shortcuts to paths and break when targets are removed. Understanding link behavior helps investigators detect file-hiding techniques, data persistence, and filesystem artifacts.

7. Challenges & Observations

- Understanding inode behavior was critical for observing differences.
- Hard links successfully preserved data after file deletion.
- Symbolic links behaved as expected and failed once the target was removed.
- Recognizing how attackers might hide files using hard links is an important forensic skill.

8. References

- Linux ln and ls manual pages
- NIST SP 800-86: Guide to Integrating Forensics into Incident Response
- Course materials provided by instructor

Lab Number: 11

Lab Title: Hidden Message Extraction with Steghide

Date: 10/31/2025

1. Introduction

This lab focused on using steganography tools to hide and extract data within an image file. Steganography is the practice of concealing messages or information within other non-secret files, such as images. The objective of this lab was to embed a hidden text file inside a JPEG image using the Steghide tool in Linux, verify the changes using file hashes and metadata, and then extract the hidden message. This technique is relevant to digital forensics, as malicious actors often use steganography to secretly transfer data or exfiltrate information.

2. Objectives

- Verify Steghide installation on a Linux system
- Inspect the original image file metadata and hash value
- Embed hidden data inside an image file using Steghide
- Verify the difference between the original and stego image using hash comparison
- Extract the hidden file and confirm successful recovery of the secret message
- Understand forensic implications of steganography

3. Tools Used

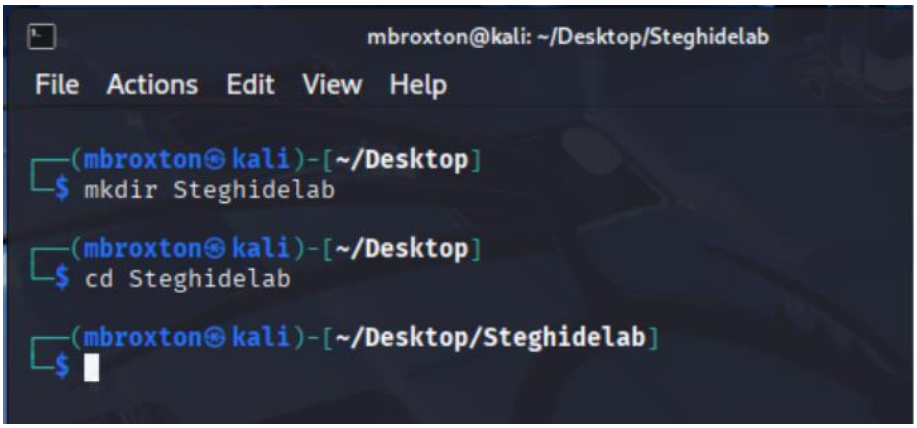
- Kali Linux / Parrot OS
- Steghide
- file command
- exiftool
- sha256sum
- Terminal

4. Lab Procedure and Results

Step 1 – Setup and Verification

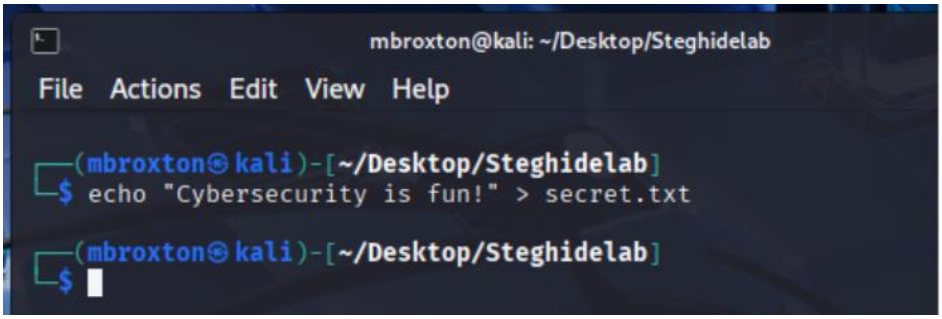
1. Created working directory:

   mkdir ~/Desktop/SteghideLab
   cd ~/Desktop/SteghideLab



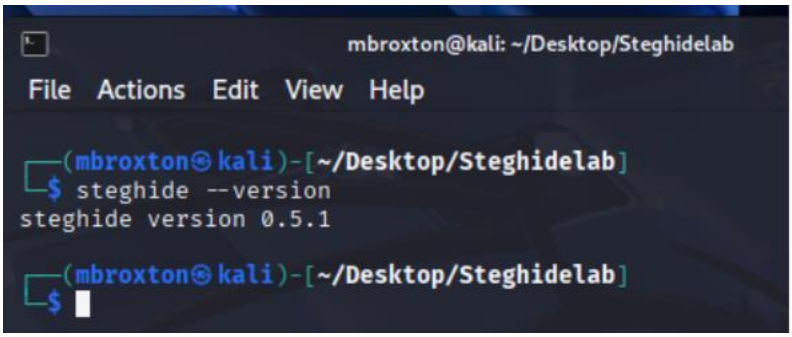
2. Created a text file containing the secret message:

   echo "Cybersecurity is fun!" > secret.txt



3. Verified Steghide installation:

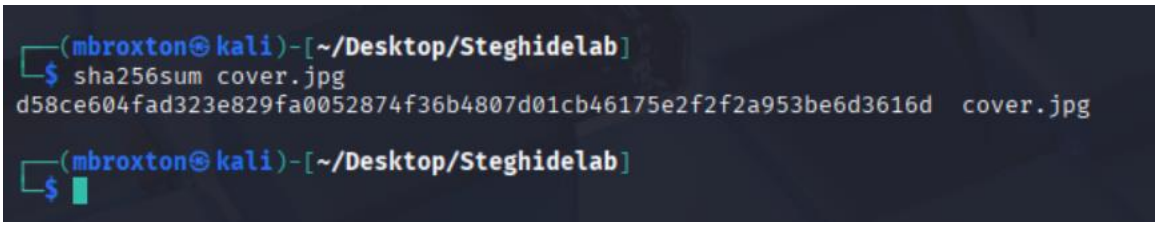
   steghide --version

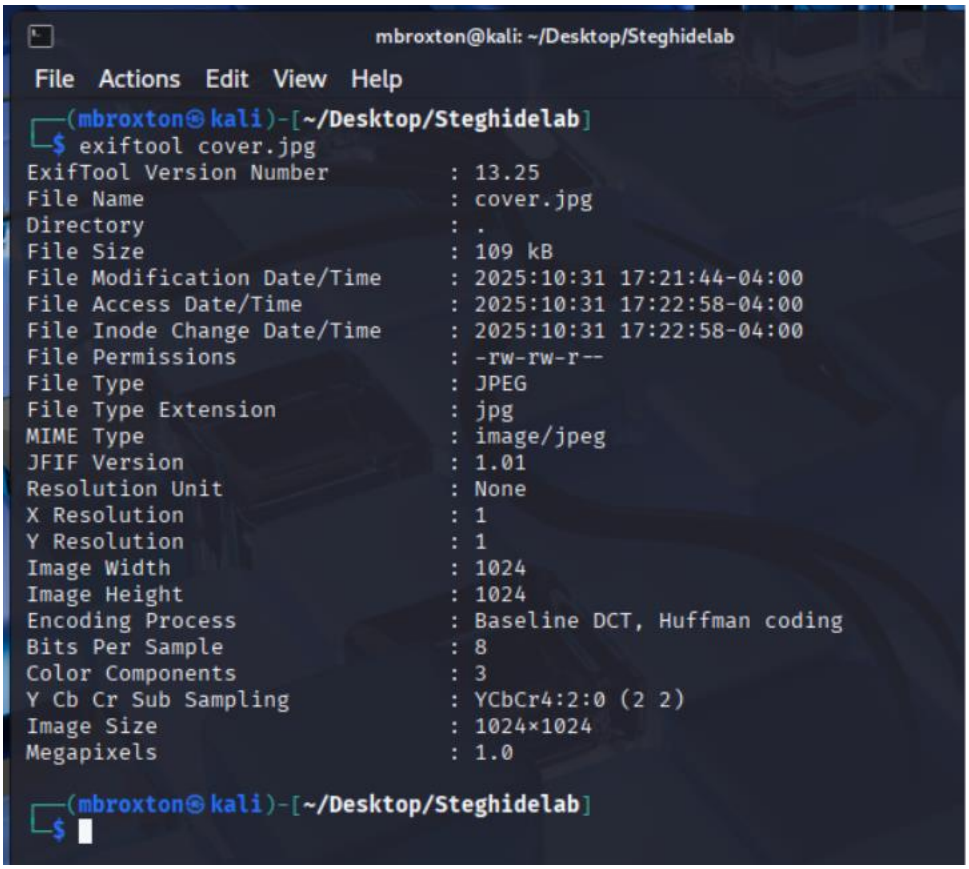


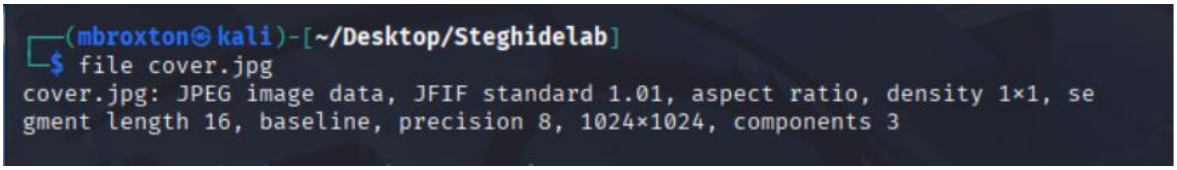
Step 2 – Analyze Original Image

1. Verified file type and metadata:

   file cover.jpg
   exiftool cover.jpg
   sha256sum cover.jpg







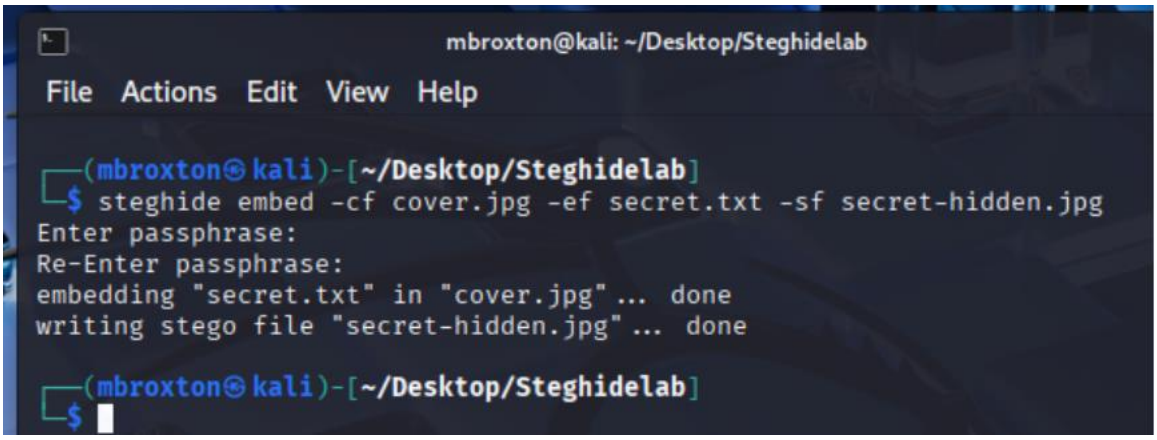
2. Recorded information and hash value for the original image.

Step 3 – Hide Secret Data in the Image

1. Embedded the secret text file into the image:

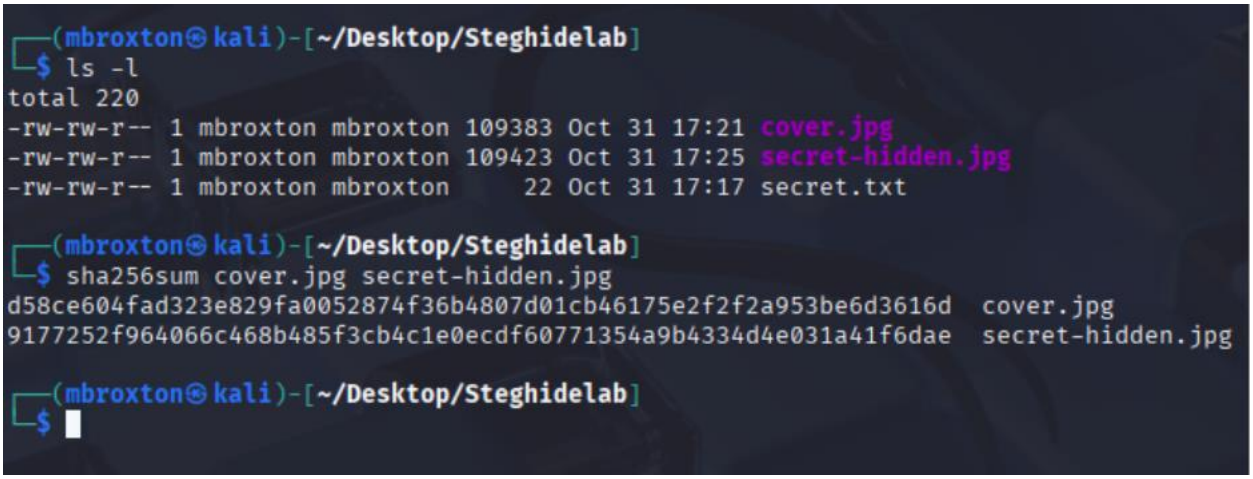
   steghide embed -cf cover.jpg -ef secret.txt -sf secret-hidden.jpg

2. Entered password when prompted.



3. Listed files and checked hashes:

   ls -l
   sha256sum cover.jpg secret-hidden.jpg



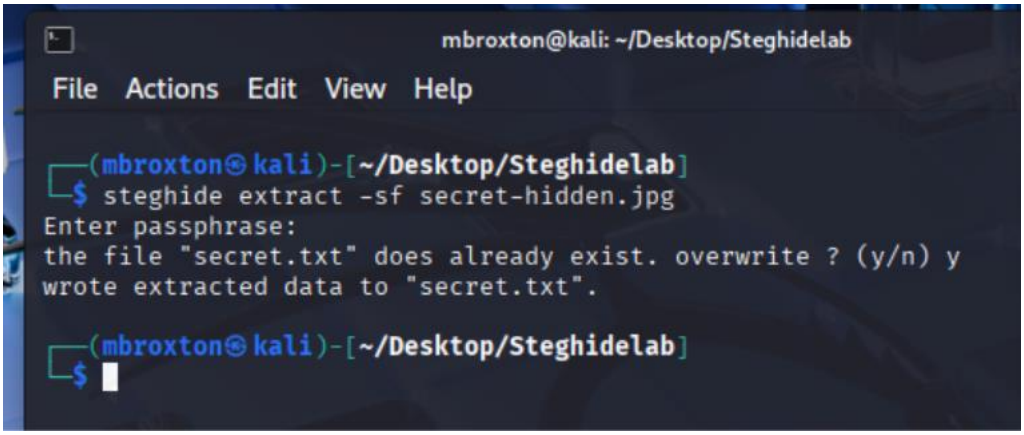
4. Observed that the hash of the new image differed from the original, proving the file was modified to include hidden data.

Step 4 – Extract Hidden Message

1. Extracted the hidden file:

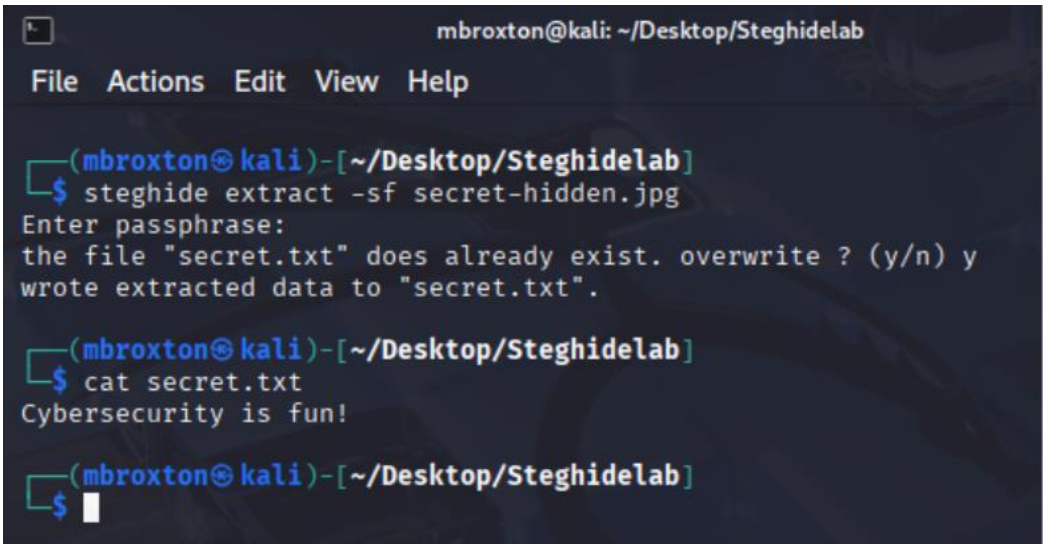
steghide extract -sf secret-hidden.jpg

2. Entered password.



3. Viewed extracted file content:

   cat secret.txt



   The extracted text matched the original secret message, confirming successful steganography and extraction.

5. Analyze Results

Steghide successfully embedded a text file into a JPEG image and later extracted it when the correct password was provided. Metadata analysis and hashing verified that the stego file differed from the original image, even though visually the files appeared identical. This demonstrates how steganography allows hidden communication without obvious evidence in the file's appearance. Hash changes confirmed the file was altered, supporting forensic validation.

6. Conclusion

This lab demonstrated the use of Steghide to conceal and extract hidden information within image files. Steganography is a covert communication method often used for malicious purposes, such as data exfiltration or hidden instruction delivery. Forensic analysts must be able to detect suspicious files, analyze metadata, compare cryptographic hashes, and extract hidden content during investigations.

7. Challenges and Observations

- Steghide required a password to extract the data, showing the importance of password-protected steganography.
- The visual appearance of the file did not change, proving that steganography is difficult to detect without forensic analysis.
- Hash comparison was critical to confirm the file was altered.

8. References

- Steghide documentation and man page
- NIST SP 800-86, Guide to Integrating Forensic Techniques
- Course materials provided by instructor