HPS

Salaheldin Sameh

Comparative Evaluation of File Recovery Tools in Digital Forensics

A Practical Study Using Simulated Data Loss

Introduction Scenario

- Imagine you're running a cluster for a critical **loan management system**
- A scheduled cleanup script goes rogue and starts **deleting or corrupting important files!**
- To recover the lost data, you turn to **open-source file recovery tools**
- **Goal of this project:**
  - Evaluate and compare popular open-source tools
  - Understand strengths and weaknesses in real recovery scenarios

Table of Contents

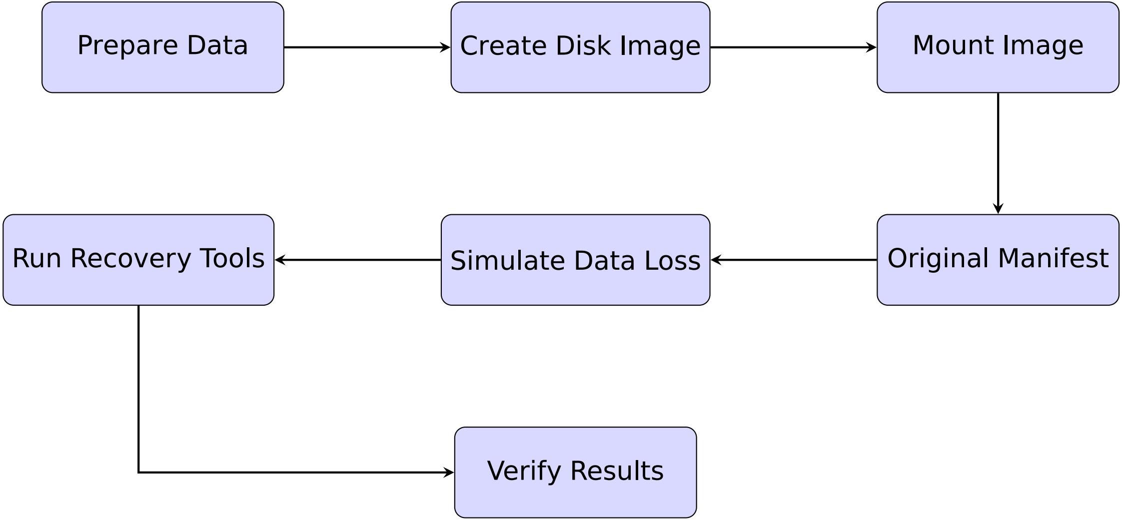
Context & Objectives

- Data loss is a frequent issue in both personal and enterprise environments
- Reliable file recovery is essential in digital forensics
- Many tools exist; their effectiveness depends on the scenario and configuration.
- **Goal:** Systematically compare popular open-source file recovery tools under realistic, controlled conditions

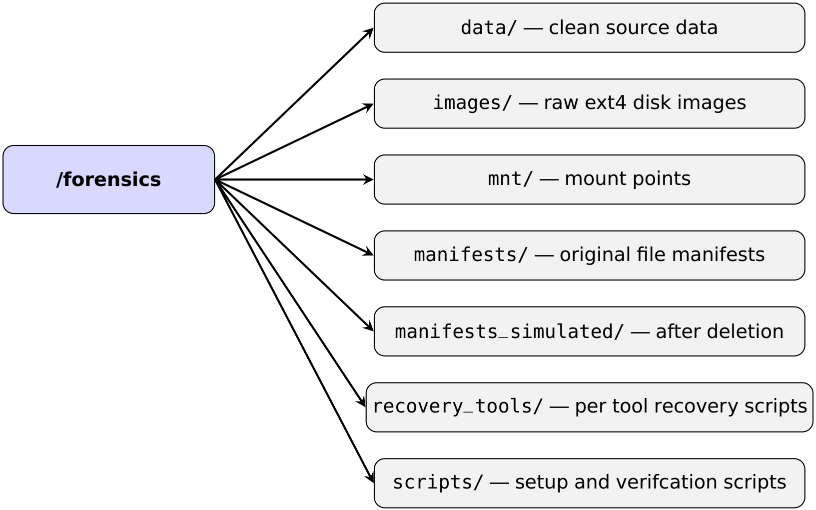
Experimental Setup

- Automated pipeline:
  - Create disk images of various sizes.
  - Populate images with different file types: text, image, docx, etc...
  - Apply random deletion and corruption
- Generate manifests for original images, and for after corruptions/deletions.
- Run tool and benchmark
- Restore images, run another tool and benchmark

Experimental Setup: Workflow Diagram



Project Structure Overview



Prepare Data (prepare_data_dirs.sh)

- Initializes clean data directories for 100MB, 1GB, and 5GB targets
- Fills each directory with realistic sample files (text, image, binary)
- Ensures directories reach 95% of target capacity to simulate near-full disks

Create Disk Images (regenerate_images.sh)

- Generates raw `.img` files from prepared data directories (100MB, 1GB, 5GB).
- Adds a buffer to account for ext4 metadata and overhead
- Creates ext4 filesystem using `mkfs.ext4`
- Mounts the image, copies the dataset, and unmounts
- Ensures a clean, repeatable disk image for every test

Generate Original Data Manifest (`generate_original_manifest.sh`)

- Scans mounted image directories for all files
- Computes SHA-256 hashes for each file
- Writes per-image CSV manifest with: `image, size, file_path, action, hash`

(sample) manifest_100mb:

```
image,size,file_path,action,hash
disk_100MB.img,100mb,/copy_453_file1.pdf,keep,3df79d34ab...
disk_100MB.img,100mb,/copy_172_file2.pdf,keep,f6edcd8a1b...
disk_100MB.img,100mb,/copy_223_pride.txt,keep,eae7160bb8...
```

Simulate Deletion and Corruption (`simulate_deletion_and_corruption.sh`)

- Iterates over each mounted image (`mnt_100MB`, etc.)
- Reads the original manifest and selects:
  - **20% of files to delete**
  - **10% of files to corrupt** (overwrite 512 bytes)
- Creates another manifest while deleting/corrupting:
  - Action marked as `deleted` or `corrupted`
  - Hashes recomputed (or blank if deleted)
- Writes results to `manifests_simulated/manifest_*.csv`

Result: Each image now reflects simulated forensic loss with accurate tracking of what happened to each file

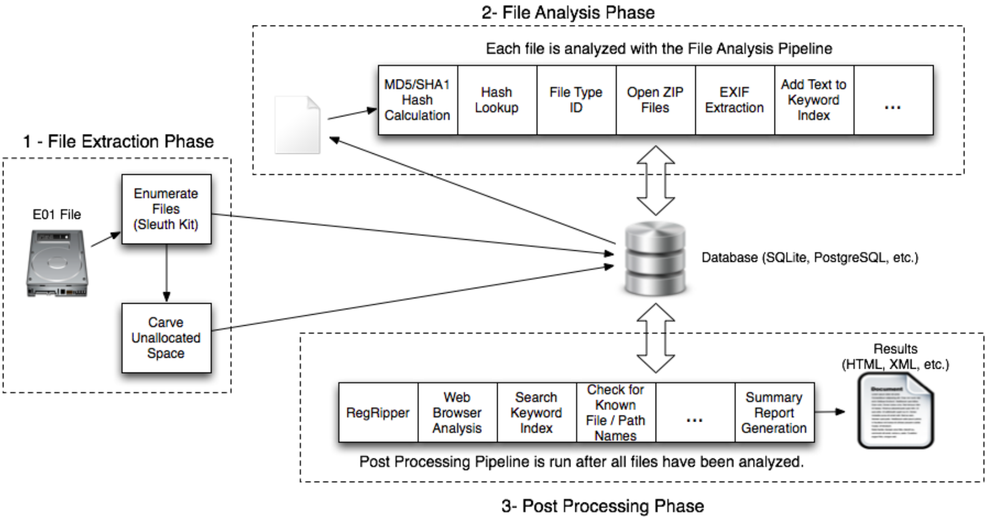
Recovery Tools Overview

- **TSK (The Sleuth Kit)**: Metadata-based recovery
- **PhotoRec**: File carving by signature; ignores metadata
- **Scalpel**: Header/footer carving; parallel and configurable

TSK Internals and Recovery Pipeline

- **Phase 1 – File Extraction**:
  - Parses raw disk image (e.g., E01) and identifies filesystem
  - Uses `fls` to enumerate all files via metadata (including deleted ones)
  - Uses `icat` to extract file content based on inode references
- **Phase 2 – File Analysis**:
  - Computes hashes (MD5/SHA1), identifies file types, opens ZIPs, extracts EXIF
  - Text content is indexed for keyword search
- **Phase 3 – Post Processing**:
  - Allows search/filtering (e.g. by RegRipper, keyword index, browser history).
  - Generates HTML/XML reports.

TSK Workflow Diagram



PhotoRec Internals

- **File Carving Approach:**
  - Ignores the file system structure, and scans raw disk image for known file headers and footers
- **Block Scanning Logic:**
  - Scans block by block
  - If a known header is detected, PhotoRec tries to reconstruct the file forward until an end condition is met
- **Recovery Conditions:**
  - Works even if the partition is missing or damaged
  - May produce partial or corrupted files if overwritten

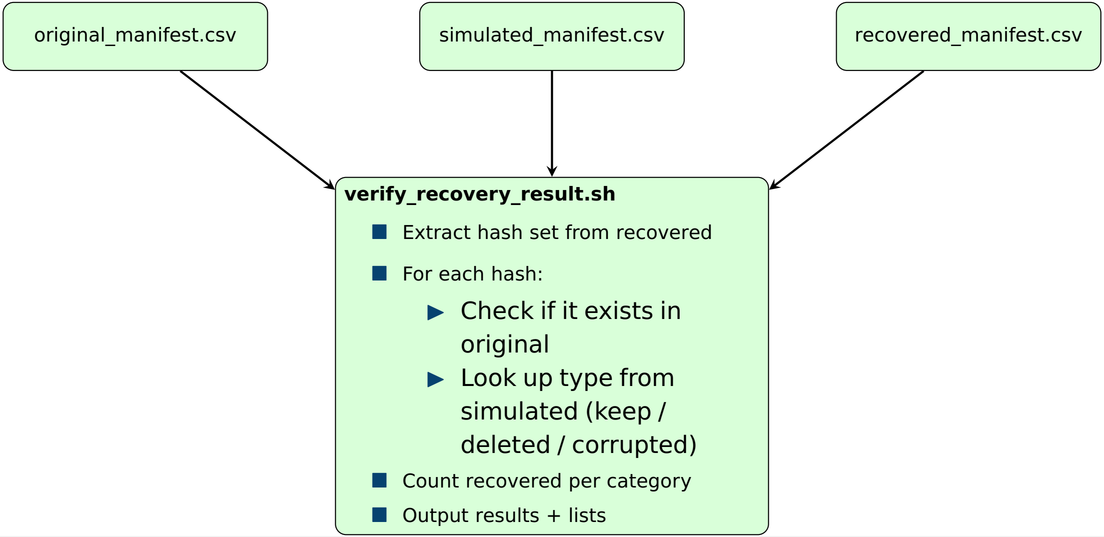
Scalpel Internals

- Carves files using header/footer patterns (defined in `scalpel.conf`)
- Pattern search via Boyer-Moore — fast and memory-efficient
- Ignores filesystem metadata entirely
- Supports multithreaded carving
- Outputs files by type into organized folders

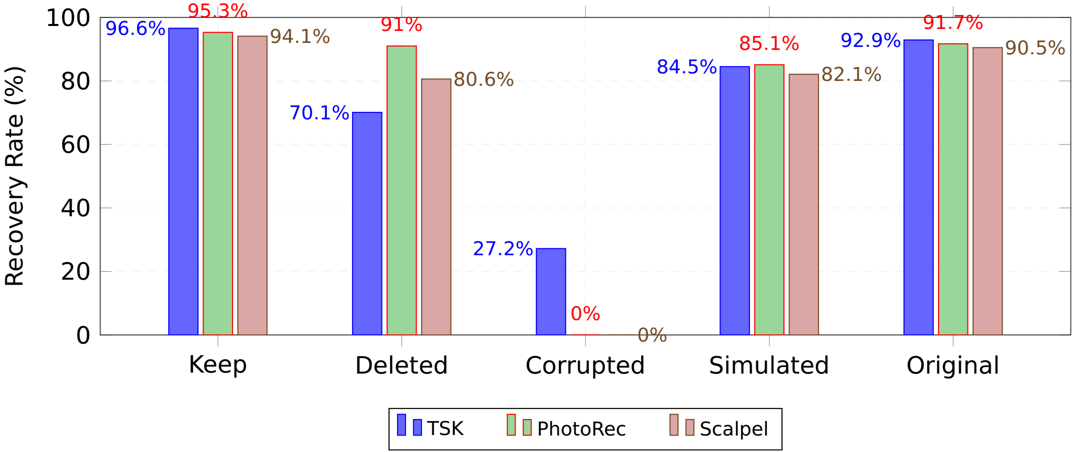
Evaluation Metrics

- **Recovery Rate:** Success for intact, deleted, and corrupted files
- **Performance:** Time taken to complete recovery
- **Usability:** Scripting, CLI options, documentation
- **Parallelism:** Multi-core support

Recovery Rate: Hash Comparison Workflow



Recovery Rate Results



Recovery Rate Results: Interpretation

- **PhotoRec** excels at deleted file carving without metadata
- **TSK** recovers both deleted and some corrupted files (metadata-aware)
- **Scalpel** is strong with deleted files if configured properly, but not for corruption

Performance Evaluation

- **Goal:** Measure how quickly each tool completes recovery
- **Metric:** Elapsed time to process a 100MB image
- **Method:**
  - Used `time ./recover.sh ...` for each tool
- **Results:**
  - TSK: 42.3 seconds
  - PhotoRec: **26.8 seconds**
  - Scalpel: 33.1 seconds
- **Conclusion:** PhotoRec is fastest; TSK is slowest due to metadata scanning

(Subjective) Usability Comparison

- **Goal:** Evaluate how easy each tool is to use and automate
- **Criteria:**
  - CLI options
  - Scriptability
  - Documentation
  - Output clarity
- **Results:**
  - PhotoRec: **High** – simple CLI, config-less batch mode
  - TSK: Medium – flexible but requires manual file system inspection
  - Scalpel: Low – config-heavy, manual setup for header/footer patterns
- **Conclusion:** PhotoRec is most user-friendly; Scalpel is complex

Parallelism Support

- **Goal:** Identify which tools benefit from multi-core systems
- **Observations:**
  - TSK: **None** – single-threaded metadata analysis
  - PhotoRec: None – single-threaded block scanning
  - Scalpel: **Full** – explicit multi-threaded recovery
- **Impact:**
  - Scalpel's performance scales with core count
  - TSK and PhotoRec has no parallelism benefit
- **Conclusion:** Scalpel leverages multiple CPUs best

Key Observations (1/2)

- **Recovery success is tightly coupled to the type of data loss**:
  - Deleted files are often recoverable
  - Corrupted files pose a greater challenge
- **Metadata-based tools** (e.g., TSK) can recover incomplete or fragmented files — but fail if metadata is missing
- **Carving-based tools** (e.g., PhotoRec, Scalpel) ignore metadata:
  - Effective for deleted data
  - Ineffective against corruption

Key Observations (2/2)

- **Output quality affects forensic usability**:
  - PhotoRec recovers raw data but not names/paths
  - TSK preserves more context (when metadata exists)
- In real-world recovery, a **single tool may not be sufficient**:
  - Combining metadata-aware and carving tools may be necessary

Future Work

- Scale up tests to larger datasets (e.g., 1GB+)
- Run on different file system types:
  - ext4, NTFS, FAT32, XFS
- Benchmark on the university cluster
- Explore additional tools (e.g., Foremost, RecoverPy)
- Finalize tool comparison and write full report

References

Carrier, B. (2024). *The Sleuth Kit & Autopsy*. Retrieved from `https://www.sleuthkit.org/sleuthkit/`

Grenier, C. (2024). *PhotoRec – Digital Picture and File Recovery*. Retrieved from `https://www.cgsecurity.org/wiki/PhotoRec`

Golden, J., Richard, G. G. (2005). *Scalpel: A Frugal, High Performance File Carver*. Retrieved from `https://github.com/sleuthkit/scalpel`

National Institute of Standards and Technology (NIST). (2014). *CFTT Report: File Carving Tools*. Retrieved from `https://www.nist.gov/itl/cftt`

Nelson, B., Phillips, A., Steuart, C. (2018). *Guide to Computer Forensics and Investigations* (6th ed.). Cengage Learning.