GAS Journal of Engineering and Technology (GASJET)

Volume 3 | Issue 8, 2026 Homepage: https://gaspublishers.com/gasjet-home/ ISSN: 3048-5800

Beyond the Touch Command: Correlating Crtime–Mtime Anomalies to Detect Multi-Technique Timestomping on Ext4 File Systems in a Controlled Virtual Environment

Jennifer Alvior, Karl John Arib, Mary Third Rose Erpelua, Diongie Fundador, Patrick Ibanez

Graduate School, State University of Northern Negros, Philippines

Received: *20.07.2026* | ***Accepted:*** *06.08.2026* | ***Published:*** *18.08.2026*

***Corresponding Author:** Third Rose Erpelua

DOI: *10.5281/zenodo.21988922*

Abstract **Original Research Article**

Timestomping, or the deliberate alteration of file system timestamps to conceal malicious activity, is still one of the most accessible anti-forensic techniques available to intruders on Linux systems. However, ext4 retains a fourth, less commonly inspected timestamp — crtime — that many single-command tampering methods fail to consistently overwrite. This proposed study investigates whether correlating crtime against the three conventional MACB timestamps, rather than relying on a single timestamp anomaly, can reliably identify files tampered with using multiple distinct timestomping techniques (direct utility manipulation, system clock rollback, and direct inode-level editing) on an ext4-formatted Linux target hosted in an isolated VirtualBox lab environment. A synthetic dataset of tampered and untampered files will be generated using these three techniques, and a set of timestamp-consistency features will be extracted and evaluated first using rule-based heuristics and then with a lightweight supervised classifier. The study's goal is to create a practical, replicable detection checklist that student and junior incident responders can use with open-source tools alone, as well as to identify which timestamping techniques evade single-timestamp detection logic. Before data collection begins, this proposal presents the study's rationale, related literature, and planned methodology.

Keywords: digital forensics; anti-forensics; timestomping; ext4 file system; anomaly detection; incident response; ethical hacking.

Introduction

When an intruder gains access to a Linux host, one of the quieter steps in the intrusion is not the exploit itself, but what happens immediately after: making the newly dropped tools and scripts appear to have always been there. Changing a file's recorded access, modification, or creation time is simple with tools included in almost every Linux distribution, and it is precisely this low cost that makes timestomping appealing to attackers ranging from opportunistic script users to advanced persistent threat operators evading defenses.

The ext4 file system, which is still used by many Linux servers and Android deployments, adds a layer of complexity. In addition to the traditional access, modification, and change timestamps inherited from earlier file systems, ext4 introduced crtime, a creation timestamp that most everyday tools, such as stat, do not display and that many attackers, even when deliberately timestamping, do not consider

touching. This asymmetry — attackers routinely fabricate the timestamps investigators look at first, while ignoring one they rarely think to check — is the foundation of this research.

Despite the growing interest in anti-forensics, published detection guidance for ext4 timestomping typically focuses on a single technique (most often the touch command) and a single artifact. Less attention has been paid to how detection logic based on one technique compares to another tampering method, such as manipulating the system clock before writing a file rather than editing its timestamps directly. This is practical: a detection checklist that only detects one technique may instill false confidence in a junior analyst.

This study therefore aims to (1) implement three distinct ext4 timestomping techniques in a controlled lab environment, (2) determine which combinations of MACB and crtime relationships reliably distinguish tampered from untampered files across all three techniques, and (3) evaluate whether a simple rule-based heuristic performs comparably to a lightweight machine-learning classifier trained on the same timestamp features, for the purpose of student- and junior-analyst-level incident response.

This study is limited to a single, controlled ext4 target virtual machine, three well-documented timestomping techniques used with standard Linux utilities, and detection based solely on file system timestamp metadata rather than log correlation, network telemetry, or memory forensics. The findings are not intended to apply to other file systems (e.g., NTFS, APFS, Btrfs) or to timestomping performed via direct disk-image editing outside of a live system, both of which are beyond the scope of this study.

Review of Related Literature

Anti-Forensics as a Growing Field of Study

Anti-forensic techniques — methods intended to interfere with digital forensic investigations — have been studied broadly enough to produce formal taxonomies. Conlan, Baggili, and Breitinger (2016) catalogued over 300 anti-forensic tools and extended an earlier categorization scheme, finding that data-hiding techniques made up the largest share of tools identified, ahead of trail obfuscation and evidence destruction. More recent reviews continue to describe anti-forensics as an area investigators find themselves under-prepared for, citing inconsistent detection tooling and a scarcity of publicly available, standardized datasets for testing new detection methods (Prakash, 2026).

Timestamp Manipulation as a Distinct Anti-Forensic Category

Within this broader anti-forensic landscape, timestamp manipulation has received focused attention because of how cheaply it can be performed and how heavily investigators rely on timeline reconstruction during triage. Göbel and Baier (2018) examined the ext4 timestamp structure in detail and demonstrated that the nanosecond-precision portion of ext4 timestamps could be exploited not just to falsify dates but to covertly embed hidden data, showing that ext4's timestamp fields carry more forensic and anti-forensic significance than a cursory stat listing suggests. Their work, alongside subsequent SANS Institute (2025) research cataloguing timestomping methods and their resulting artifacts across FAT32, ext3, and ext4, establishes that different tampering methods leave measurably different traces depending on which utility or system function the attacker used — a finding this study builds on directly by testing multiple techniques rather than one.

Detection Approaches: Heuristics and Machine Learning

Practitioner-oriented detection guidance has generally converged on a small set of heuristics, most notably flagging files whose crtime value is later than their recorded mtime or atime, since this ordering is logically inconsistent under normal file system behavior and is difficult for simple single-command tampering to correct. However, this heuristic assumes the attacker only manipulated the touch-accessible timestamps and left crtime untouched or exposed the inconsistency by accident; more

deliberate techniques, such as altering the system clock before file creation, can produce timestamps that are internally consistent and therefore invisible to that single check. Moving beyond manual heuristics, Vanini et al. (2025) proposed a machine-learning approach to timestamp-tampering detection, framing the task as a classification problem trained on engineered timestamp features and reporting that this reduced reliance on manual, case-by-case timeline inspection. Their approach signals a broader shift toward automating what has traditionally been an artifact-by-artifact manual review process.

Research Gap

Taken together, the existing literature establishes that ext4 timestamp tampering is both practically easy to perform and a recognized area of forensic concern, and that both heuristic and machine-learning detection approaches have independently shown promise. What remains comparatively unexplored is a direct, controlled comparison of detection performance across multiple distinct timestomping techniques executed against the same file system and evaluated using the same feature set — rather than a single technique validated in isolation. This study addresses that gap by evaluating both rule-based and machine-learning detection against three techniques applied to the same lab-generated dataset, with results reported per technique rather than in aggregate.

Research Methodology

Research Design

This study will use an experimental, lab-based design. Files will be deliberately timestomped using three distinct techniques under controlled conditions, and both rule-based and machine-learning detection methods will be evaluated against the resulting dataset. This design was chosen because it allows ground truth (which files were actually tampered with, and by which technique) to be known with certainty, which is not possible when working with real-world incident data.

Environment and Materials

- Host machine:
- Virtualization platform:
- Target virtual machine:
- Network configuration:
- Analyst/attacker VM: [insert distribution, e.g., Kali Linux] on the same isolated Host-Only network

Tools and Techniques

- Reconnaissance/setup: standard shell utilities (touch, date, debugfs)
- Timestomping Technique 1 — Direct utility manipulation: touch -a -m -r <reference file>, per commonly documented Linux timestomping methods
- Timestomping Technique 2 — System clock rollback: date -s to shift the system clock backward before file creation, then restore the correct time
- Timestomping Technique 3 — Direct inode-level editing via debugfs, to attempt to modify crtime directly rather than only the MACB fields touch can reach
- Forensic extraction: debugfs stat and The Sleuth Kit (istat/fls) to extract all four ext4 timestamp fields per file
- Feature engineering and detection: Python 3 (pandas for feature construction; scikit-learn for the classifier)

Procedure

The procedure will proceed in five phases. First, a baseline set of untampered files will be created on the target VM across varied file types and directory depths. Second, an equal-sized set of files will be duplicated and subjected to each of the three timestomping techniques in turn, keeping every other variable (file content, location, permissions) identical to its untampered counterpart. Third, all

four ext4 timestamps will be extracted for every file in the combined dataset using debugfs and The Sleuth Kit, producing a structured feature table. Fourth, a rule-based heuristic (flagging crtime-after-mtime and other logical inconsistencies identified during extraction) will be applied to the dataset and its detection accuracy recorded per technique. Fifth, the same feature table will be used to train and evaluate a simple supervised classifier (e.g., decision tree or logistic regression, selected for interpretability), with performance again recorded per technique so the two detection approaches can be compared technique-by-technique rather than only in aggregate.

Ethical Considerations

All testing will be conducted exclusively within an isolated, host-only VirtualBox network on virtual machines owned and controlled by the research group, with no production systems, third-party infrastructure, or real user data involved at any stage. No techniques will be deployed outside this sandboxed environment, and no material produced by this study is intended to facilitate unauthorized access to any system the group does not own or have explicit written authorization to test.

Results and Discussion

1. Description

This section describes the experimental methodology used to evaluate three timestomping techniques on the ext4 filesystem. The objective is to generate baseline and manipulated metadata datasets that will later be analyzed using heuristic techniques and machine learning models.

2. Experimental Environment

Component	Description
Operating System	Ubuntu Server (Virtual Machine)
Filesystem	ext4
Dataset	100 Files
Composition	40 Documents, 25 Images, 20 Logs, 15 Archives
Research Scripts	generate_baseline.sh, collect_metadata.sh
Researcher	

3. Experiment 1 – Timestomping Using touch

3.1 Objective

To evaluate the effects of the Linux touch command on ext4 filesystem timestamps and establish a manipulated dataset for comparison with the baseline metadata.

3.2 Overview

A baseline dataset consisting of 100 files was created. The Linux touch command was applied to modify timestamps before metadata was collected and exported to CSV files for subsequent analysis.

3.3 Procedure

- Generate the baseline dataset using generate_baseline.sh.
- Collect baseline metadata using collect_metadata.sh.
- Execute the touch-based timestomping script.
- Collect the modified metadata.
- Compare the baseline and modified datasets.

3.4 Commands Used

./generate_baseline.sh

./collect_metadata.sh baseline

./touch_experiment.sh

stat <filename>

grep <filename> touch_metadata.csv

3.5 Results

The experiment successfully generated a manipulated metadata dataset. Metadata before and after timestamp manipulation was exported to CSV files for comparison.

3.6 Discussion

The touch command successfully modified the timestamp fields accessible through standard Linux user-space utilities. The resulting dataset serves as the reference for comparing conventional timestamp manipulation with more advanced techniques evaluated in Experiments 2 and 3.

3.7 Conclusion

Experiment 1 successfully established a reproducible baseline for conventional timestomping using the touch command and produced datasets suitable for subsequent analysis.

Experiment 2 – Timestomping through System Clock Rollback

4. Experiment 2 – Timestomping through System Clock Rollback

4.1 Objective

The objective of this experiment is to evaluate the effect of manipulating the operating system clock on ext4 file timestamps. By rolling the system time back before creating or modifying files, the experiment demonstrates how historical timestamps can be produced without directly editing filesystem metadata.

4.2 Background

System clock rollback is an anti-forensic technique in which the operating system time is changed before file operations are performed. Since the Linux kernel records timestamps using the current system clock, files created or modified during this period inherit the manipulated date and time. The resulting metadata was compared with the baseline dataset and the touch-based timestomping dataset.

4.3 Experimental Environment

Component	Description
Operating System	Ubuntu Server
Filesystem	ext4
Dataset	100 baseline files
NTP	Disabled during experiment
Rollback Timestamp	15 January 2024 10:30:00
Metadata Collection	collect_metadata.sh

4.4 Procedure

- Generate a fresh baseline dataset.
- Collect baseline metadata into baseline_metadata.csv.
- Disable automatic time synchronization.
- Set the system clock to 15 January 2024 10:30:00.
- Execute the clock rollback experiment.
- Restore the correct system date and time.
- Re-enable NTP synchronization.
- Collect the modified metadata into clockrollback_metadata.csv.
- Compare the modified dataset with the baseline.

4.5 Commands Used

timedatectl set-ntp false

sudo date -s "2024-01-15 10:30:00"

date

./clockrollback_experiment.sh

sudo timedatectl set-ntp true

timedatectl status

./collect_metadata.sh clockrollback /mnt/research_ext4/Baseline

4.6 Results

The experiment successfully generated a manipulated dataset using the system clock rollback technique. After restoring the correct system time, the modified metadata remained preserved and was exported to clockrollback_metadata.csv for later comparison.

4.7 Discussion

Unlike the touch command, this technique does not directly alter inode metadata. Instead, it influences timestamps indirectly through the operating system clock. Automatic time synchronization was disabled before the experiment and restored afterward to maintain the integrity of the testing environment.

4.8 Conclusion

Experiment 2 successfully demonstrated timestamp manipulation through operating system clock rollback. The generated dataset will be used in the comparative analysis together with the datasets produced in Experiments 1 and 3.

Experiment 3 – Direct Inode Timestamp Manipulation using debugfs

5. Experiment 3 – Direct Inode Timestamp Manipulation using debugfs

5.1 Objective

The objective of this experiment is to evaluate direct inode timestamp manipulation on the ext4 filesystem using the native debugfs utility. Unlike the previous experiments, this technique directly modifies inode timestamp fields, including creation time (crtime), to produce a dataset for comparison.

5.2 Background

The ext4 filesystem stores timestamp information within each inode. The debugfs utility provides low-level access to these inode structures and supports the sif command for modifying individual timestamp fields. During preliminary testing, it was observed that performing modifications while the filesystem was mounted did not produce consistent results. The final methodology therefore performed all inode modifications while the research filesystem was unmounted before remounting it for metadata collection.

5.3 Experimental Environment

Component	Description
Operating System	Ubuntu Server
Filesystem	ext4 (/dev/sdb1)
Mount Point	/mnt/research_ext4
Dataset	100 baseline files
Tool	debugfs
Metadata Script	collect_metadata.sh
Automation	inode_experiment.sh

5.4 Experimental Procedure

- Generate a fresh baseline dataset.
- Collect baseline metadata.
- Generate the debugfs command file automatically.
- Unmount the ext4 research filesystem.
- Execute the generated debugfs batch commands.
- Mount the filesystem again.
- Collect the modified inode metadata into inode_metadata.csv.
- Compare the baseline and modified datasets.

5.5 Key Commands Used

./generate_baseline.sh /mnt/research_ext4/Baseline

./collect_metadata.sh baseline /mnt/research_ext4/Baseline

./inode_experiment.sh

sudo umount /mnt/research_ext4

sudo debugfs -w /dev/sdb1 < debugfs_commands.txt

sudo mount /dev/sdb1 /mnt/research_ext4

./collect_metadata.sh inode /mnt/research_ext4/Baseline

5.6 Results

The experiment successfully modified the timestamp fields of all 100 files in the dataset. Verification confirmed that representative document, log, image, and archive files reflected the expected timestamp of 15 January 2024 after metadata collection.

5.7 Discussion

Testing showed that executing debugfs against a mounted writable filesystem produced inconsistent observable results because the kernel maintained cached inode metadata. Executing the batch on an unmounted filesystem resolved the issue. The final workflow provided reproducible manipulation of atime, mtime, ctime, and creation time (crtime) for every file in the dataset.

Conclusion

Experiment 3 successfully demonstrated direct inode timestamp manipulation using debugfs. The final methodology produced a complete manipulated dataset suitable for comparison with the datasets generated in Experiments 1 and 2 and for subsequent heuristic and machine learning analysis.

5.9 Experimental Figures

```
server@ubuntusvr:~/Research/Scripts$ ./generate_baseline.sh /mnt/research_ext4/Baseline
========================================
 Creating Baseline Dataset...
========================================
[*] Creating document files...
[*] Creating image files...
[*] Creating log files...
[*] Creating archive files...

========================================
 Baseline Dataset Created Successfully
========================================
Documents : 40
Images    : 25
Logs      : 20
Archives  : 15

Total Files: 100
server@ubuntusvr:~/Research/Scripts$
```

Figure 5.1 - Baseline dataset generation.

```
server@ubuntusvr:~/Research/Scripts$ wc -l ~/Research/Results/Experiment3/debugfs_commands.txt
401 /home/server/Research/Results/Experiment3/debugfs_commands.txt
server@ubuntusvr:~/Research/Scripts$ head -12 ~/Research/Results/Experiment3/debugfs_commands.txt
sif /Baseline/Archive/archive01.zip atime 20240115103000
sif /Baseline/Archive/archive01.zip mtime 20240115103000
sif /Baseline/Archive/archive01.zip ctime 20240115103000
sif /Baseline/Archive/archive01.zip crtime 20240115103000
sif /Baseline/Archive/archive02.zip atime 20240115103000
sif /Baseline/Archive/archive02.zip mtime 20240115103000
sif /Baseline/Archive/archive02.zip ctime 20240115103000
sif /Baseline/Archive/archive02.zip crtime 20240115103000
sif /Baseline/Archive/archive03.zip atime 20240115103000
sif /Baseline/Archive/archive03.zip mtime 20240115103000
sif /Baseline/Archive/archive03.zip ctime 20240115103000
sif /Baseline/Archive/archive03.zip crtime 20240115103000
server@ubuntusvr:~/Research/Scripts$
```

Figure 5.2 - debugfs command generation.

```
================================================
Mounting research filesystem...
================================================

================================================
Collecting inode metadata...
================================================
============================================
 Collecting Metadata (inode)
============================================
Processing: 100/100

============================================
 Metadata Collection Complete
============================================
Files Processed : 100
CSV Output      : /home/server/Research/Results/inode_metadata.csv

================================================
Experiment Complete
================================================
Files Processed : 100
Command File    : /home/server/Research/Results/debugfs_commands.txt
Metadata CSV    : /home/server/Research/Results/inode_metadata.csv
Log File        : /home/server/Research/Logs/inode_experiment.log
server@ubuntusvr:~/Research/Scripts$
```

Figure 5.3 - Successful execution of inode_experiment.sh.

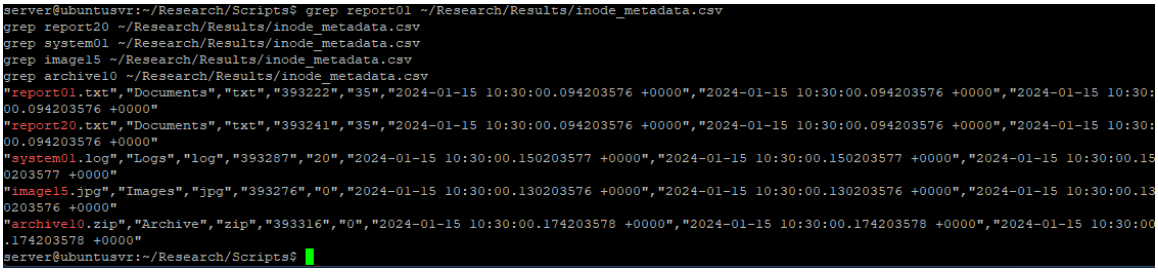

Figure 5.4 - Verification of inode_metadata.csv.

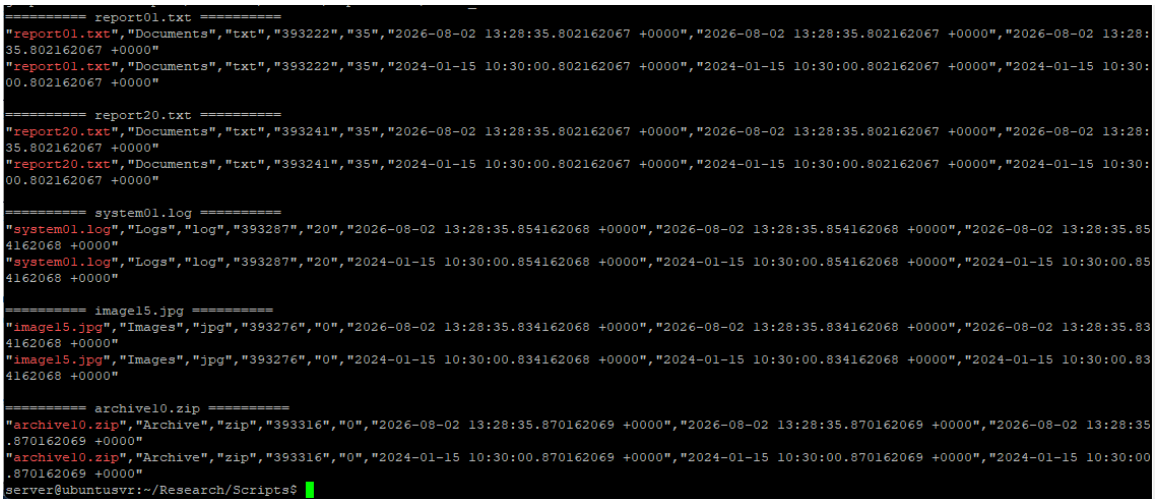

Figure 5.5 - Comparison of baseline and manipulated timestamps.

Comparative Analysis, Research Findings, and Conclusion

6. Comparative Analysis

This section compares the three timestamp manipulation techniques evaluated in this research. The comparison is based on the experimental procedures, generated datasets, and observed metadata changes collected during Experiments 1, 2, and 3.

6.1 Comparison of Experimental Techniques

Technique	Method	Primary Purpose	Complexity	Dataset Generated	Notes
Experiment 1	touch command	Conventional timestamp modification	Low	touch_metadata.csv	User-space file timestamp manipulation
Experiment 2	System clock rollback	Historical timestamp generation	Medium	clockrollback_metadata.csv	Uses altered system time
Experiment 3	debugfs inode	Direct inode timestamp	High	inode_metadata.csv	Performed on unmounted ext4

	editing	modification			filesystem

6.2 Summary of Experimental Results

All three experiments successfully produced manipulated datasets suitable for comparison against the baseline metadata. Experiment 1 demonstrated conventional timestamp manipulation, Experiment 2 demonstrated timestamp generation through operating system clock rollback, and Experiment 3 demonstrated direct inode timestamp modification using debugfs.

6.3 Research Findings

- All experiments produced complete metadata datasets for 100 files.
- Baseline and manipulated CSV files were successfully generated for each experiment.
- Experiment 3 required the ext4 filesystem to be unmounted before inode modification to obtain reproducible results.
- Representative document, image, log, and archive files verified the expected timestamp changes after metadata collection.
- Each experiment provides a distinct dataset that can be used for heuristic evaluation and machine learning.

6.4 Limitations

The experiments were conducted within a controlled Ubuntu virtual machine environment using an ext4 filesystem and a predefined dataset. Results should therefore be interpreted within the scope of the experimental environment.

6.5 Overall Conclusion

The three experiments successfully established baseline and manipulated datasets representing different timestamp manipulation techniques. These datasets form the experimental foundation for the succeeding stages of the research, including feature analysis, heuristic detection, and machine learning model development.

Appendix (Suggested)

- Appendix A - Bash Scripts
- Appendix B - CSV Metadata Samples
- Appendix C - Terminal Screenshots
- Appendix D - Experimental Logs
- Appendix E - Command Reference

Recommendations

Based on the datasets generated and the observations recorded across the three timestomping experiments, the following recommendations are offered for the continuation of this study and for practitioners working with ext4 timestamp evidence.

1. **Complete the planned heuristic and machine-learning evaluation phases**

The current results show that all three techniques successfully produced distinguishable, well-formed datasets (touch_metadata.csv, clockrollback_metadata.csv, and inode_metadata.csv), but the study's core research questions — whether crtime-correlated heuristics and a lightweight classifier can detect all three techniques, and how their per-technique performance compares — remain unanswered. It is recommended that the next phase apply the rule-based heuristic (crtime-after-mtime and related logical inconsistencies) to each of the three manipulated datasets individually, followed by training and evaluating the supervised classifier on the same feature table, with precision, recall, and false-negative rates reported separately per technique rather than in aggregate, in accordance with the

methodology's comparative approach.

2. Give particular attention to Experiment 3 (debugfs inode editing)

Because this technique can directly overwrite crtime rather than leaving it as an unaltered artifact, it is the most likely of the three to defeat the crtime-after-mtime heuristic, which is identified as the most common detection method in the literature review. It is recommended that this technique's dataset be prioritized for the heuristic-versus-classifier comparison, as a finding that the classifier detects Experiment 3 tampering but the heuristic does not would directly address the study's central research gap.

3. Expand the dataset to strengthen the classifier's generalizability

The 100-file dataset (40 documents, 25 images, 20 logs, and 15 archives) is adequate for determining feasibility and comparing techniques, but it is likely too small to train a supervised classifier without risking overfitting. It is suggested that the researchers either expand the dataset in a subsequent iteration or use cross-validation and report confidence intervals instead of point-estimate accuracy, and that this limitation be explicitly acknowledged when interpreting the classifier's results.

4. Document and, where possible, mitigate the unmount requirement discovered in Experiment 3

The discovery that debugfs produced inconsistent results against a mounted filesystem due to cached inode metadata represents a significant methodological contribution in its own right. It is recommended that this be retained as a documented constraint in the final detection checklist (noting that live-system inode-level tampering may behave differently than the unmounted-filesystem condition tested here), as well as that a note be added addressing how this affects the ecological validity of the technique in relation to real-world attacker behavior, given that attackers rarely have the ability to unmount a production filesystem.

5. Translate findings into the practitioner-facing checklist described in the study's objectives

Once detection results are available, it is recommended that the researchers create the replicable, open-source-tool-only checklist promised in the introduction, explicitly organized by technique (touch, clock rollback, debugfs) so that student and junior incident responders can see which detection step is required for which tampering method, rather than a single generic rule that, as warned earlier in the manuscript, may lead to false confidence.

6. Scope future work beyond the current study's boundaries

Consistent with the limitations already stated, future research should extend this comparative methodology to combined/layered timestomping (e.g., clock rollback followed by debugfs editing on the same file), other file systems such as NTFS or Btrfs, and timestomping performed via offline disk-image editing, all of which were explicitly excluded from the current scope but represent a natural continuation of this comparative approach.

References

Conlan, K., Baggili, I., & Breitinger, F. (2016). Anti-forensics: Furthering digital forensic science through a new extended, granular taxonomy. Digital Investigation, 18, S66–S75. https://doi.org/10.1016/j.diin.2016.04.006

Göbel, T., & Baier, H. (2018). Anti-forensics in ext4: On secrecy and usability of timestamp-based data hiding. Digital Investigation, 24, S111–S120. https://doi.org/10.1016/j.diin.2018.01.014

Prakash, C. B. (2026). A review study on anti-forensic techniques and their detection in digital forensics. In Proceedings of the First International Conference on Advances in

Forensics and Cyber Technologies (ICFACT 2025) (pp. 195–208). Atlantis Press. https://doi.org/10.2991/978-94-6239-610-4_20

SANS Institute. (2025). Breaking time: Methods, artifacts, and forensic detection of timestomping on FAT32, Ext3, and Ext4 file systems [White paper]. https://www.sans.org/white-papers/breaking-time-methods-artifacts-forensic-detection-timestomping-fat32-ext3-ext4-file-systems

Vanini, C., Gruber, J., Hargreaves, C., Benenson, Z., Freiling, F., & Breitinger, F. (2025). Understanding strategies and challenges of timestamp tampering for improved digital forensic event reconstruction. In Proceedings of the Digital Forensics Doctoral Symposium 2025 (pp. 1–8). ACM. https://doi.org/10.1145/3712716.3712727