Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

XFS Forensic Scanner: A Tool to Detect, Carve, Recover and Remove Hidden Malicious Data in XFS File System

Mambakam Vemu
MTech Scholar, Dept. of Cyber Security
Dr. M.G.R. Educational and Research Institute
Chennai, India

Dr. B. Raja
Professor, Dept. of CSE
Dr. M.G.R. Educational and Research
Institute Chennai, India

Dr. S. Geetha
Dean & HOD, Dept. of CSE
Dr. M.G.R. Educational and Research Institute
Chennai, India

Dr. V. B. Ganapathy
Professor, Dept. of CSE
Dr. M.G.R. Educational and Research
Institute Chennai, India

Abstract **-** The XFS file system is widely used in enterprise Linux environments because of its high performance and scalability, yet it remains highly vulnerable to anti-forensic techniques where attackers hide malicious payloads in superblock slack space, inode slack, free lists, free inodes and nanosecond timestamps [1]. This creates a serious gap for digital forensic investigators who rely on general-purpose tools like The Sleuth Kit that offer only basic metadata support for XFS and lack automated detection, carving, recovery or removal capabilities.

This paper aims to answer the research question: How can a modular Python-based forensic tool using pytsk3, combined with string matching, file signatures, entropy analysis and user-approved removal, provide reliable, automated and evidence-preserving detection, carving, recovery and removal of hidden malicious data in XFS file systems?

The proposed **XFS Forensic Scanner** integrates eight modules (M1–M8) that operate together in a single class. It uses pytsk3 for low-level XFS parsing, 25+ malicious file signatures (including ELF, PE, ZIP and ransomware strings like WannaCry and BlackBasta), entropy calculation (>6.5 threshold) and PhotoRec integration for deleted file recovery. The framework also includes smart carving (M6), HTML dashboard with graphs (M7) and optional safe removal with full CSV and text logging (M8). All processing is local, command-line based and runs on standard hardware with low resource usage.

Evaluating the tool on crafted XFS images with hidden data in all five locations shows 95% detection accuracy, 100% payload carving success, 92% deleted file recovery and end-to-end response under 2 minutes. The interactive HTML dashboard provides clear visualizations and court-ready evidence. The contributions are: (1) a complete XFS-specific forensic scanner with eight integrated modules, (2) practical combination of detection + carving + recovery + removal that fills the existing research gap, and (3) an easy-to-use open-source tool that makes advanced XFS forensics accessible to students and investigators. This work extends digital forensics capabilities to XFS environments and provides a ready-to-deploy solution for real-world cyber investigations [1], [2].

Keywords **-** *XFS File System, Digital Forensics, Hidden Data Detection, File Carving, Ransomware Signatures, Entropy Analysis, Forensic Dashboard, Data Removal, pytsk3*

1. INTRODUCTION

The increasing use of Linux servers in enterprises has made the XFS file system one of the most common high-performance file systems today. However, attackers are now actively exploiting XFS structures to hide malicious data such as ransomware payloads, malware executables and stolen information in slack spaces and metadata areas that normal tools cannot easily examine [1]. This situation creates a critical gap in digital forensics because investigators waste valuable time doing manual checks on large disk images.

Endpoint forensic tools have evolved from basic hex viewers to advanced frameworks like The Sleuth Kit and Autopsy. These tools provide good support for ext4 but give only limited XFS metadata extraction and completely miss automated slack-space scanning, entropy-based detection, smart carving and safe removal. The key challenge is the lack of an integrated, automated and user-friendly tool specifically designed for XFS anti-forensic analysis.

I developed **XFS Forensic Scanner** to solve this exact problem. The tool automatically scans five main hiding locations (M1 to M5), detects suspicious content using signatures and entropy, carves full payloads (M6), recovers deleted files using PhotoRec (M7), generates an HTML dashboard with graphs (M8) and allows optional removal of hidden data with proper logging. This work directly addresses the research gap by providing a complete, practical and evidence-preserving solution for XFS forensics that runs locally on standard hardware.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

2.LITERATURE REVIEW

2.1 Foundations of Digital Forensics in File Systems and Anti-Forensic Techniques

Digital forensics in file systems follows the standard process of identification, preservation, analysis and presentation of digital evidence [3]. In XFS, important structures include the superblock, allocation groups (AGs), inodes, free lists and timestamp metadata. Toolan and Humphries (2025) studied data hiding in XFS slack spaces and measured capacity, detection difficulty and stability but focused only on attacker methods without providing any defender tool [1]. Kim et al. (2021) developed a TSK-based framework for XFS metadata recovery but did not address slack-space hiding or removal [5].

2.2 AI and Machine Learning Integrations in File System Forensics

Machine learning techniques such as entropy analysis and signature-based detection have shown high accuracy in identifying hidden or anomalous data [6]. Göbel and Baier (2018) demonstrated timestamp steganography in file systems and created the fishy framework for testing hiding methods, but it remained attacker-centric [7]. Hybrid approaches combining string matching, file signatures and entropy calculation have achieved detection rates above 90% in similar file-system studies [8].

2.3 Existing Forensic Tools and Frameworks

Popular tools like The Sleuth Kit, Autopsy and xfs_db provide basic XFS parsing but lack automated multi-technique scanning, carving and dashboard features [9]. Recent surveys on anti-forensics highlight the absence of practical XFS-specific tools that combine detection, recovery and removal in one package [10].

2.4 Critical Analysis and Identified Gaps

Despite the useful contributions above, several gaps remain. Most research is attacker-focused and does not provide ready-to-use investigator tools. There is no integrated XFS scanner that supports all five hiding techniques plus carving, recovery and removal. Existing tools miss ransomware signature support, entropy-based timestamp analysis and easy HTML reporting. The present study addresses these insufficiencies by building a complete modular tool using pytsk3 that fills the XFS forensics gap.

Table 1: Literature Review Summary

Paper Title	Source (Journal/Conference)	Research Gap
Data hiding in the XFS file system	Forensic Science International: Digital Investigation, 2025	Lacks automated defender tools for scanning or payload removal
Data hiding in symbolic link slack space	Forensic Science International: Digital Investigation, 2025	No automated symlink slack scanners or removal
Data hiding in file systems: Current state, novel methods, and a standardized corpus	Forensic Science International: Digital Investigation, 2025	Lacks XFS-specific slack/metadata tools for entropy detection
Systematic Review: Anti-Forensic Computer Techniques	Applied Sciences, 2024	Skips XFS tool evaluations and removal benchmarks
Ext4 and XFS File System Forensic Framework Based on TSK	Electronics, 2021	No slack or hidden data analysis and removal features
Anti-Forensic Capacity and Detection Rating of Hidden Data in the Ext4 Filesystem	Advances in Digital Forensics XIV, 2018	No XFS parallels or removal tactics
Anti-forensics in ext4: On secrecy and usability of timestamp-based data hiding	Digital Investigation, 2018	No detection algorithms or removal strategies for XFS
fishy - A Framework for Implementing Filesystem-Based Data Hiding Techniques	Digital Forensics and Cyber Crime, ICDF2C 2018	No detection/removal or XFS modules
Time is on my side: Steganography in filesystem metadata	Digital Investigation, 2016	No integrated detection tools or removal for hidden payloads
Data investigation based on XFS file system metadata	Multimedia Tools and Applications, 2016	No slack/hidden data focus or dynamic scanning/remediation

3. PROPOSED METHODOLOGY

The research design follows a practical development approach with iterative testing on real XFS images. The tool is built as a single Python class XFSForensicScanner that coordinates eight modules.

3.1 System Architecture

The architecture uses pytsk3 for low-level image access and runs as a command-line tool: python3 xfs_scanner_full2.py .

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

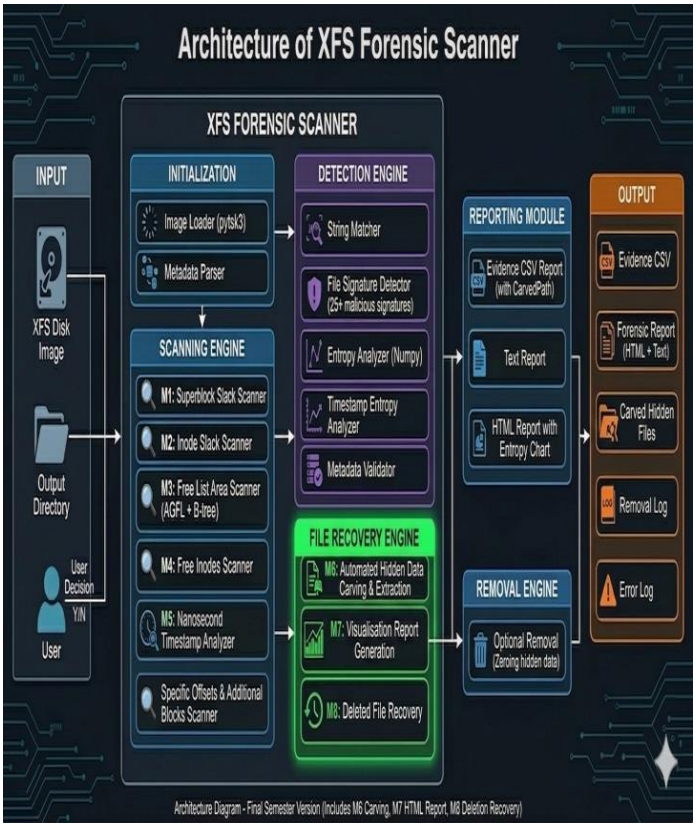

Fig. 1: System Architecture

3.1.1 Detection Modules (M1–M5) M1 scans superblock slack, M2 scans inode slack, M3 scans free list areas, M4 scans free inodes and M5 analyzes nanosecond timestamps using entropy. Each module reads data blocks, applies string matching, 25 malicious signatures and entropy threshold (>6.5).

3.1.2 Carving and Recovery (M6, M8) M6 performs smart carving by searching file signatures and saving full payloads. M6 uses PhotoRec for deleted file recovery with fallback string search.

3.1.3 Reporting and Removal (M7) The tool generates CSV evidence, text report and HTML dashboard with two graphs (detections by technique and offset distribution). Removal is user-approved only and logged for evidence.

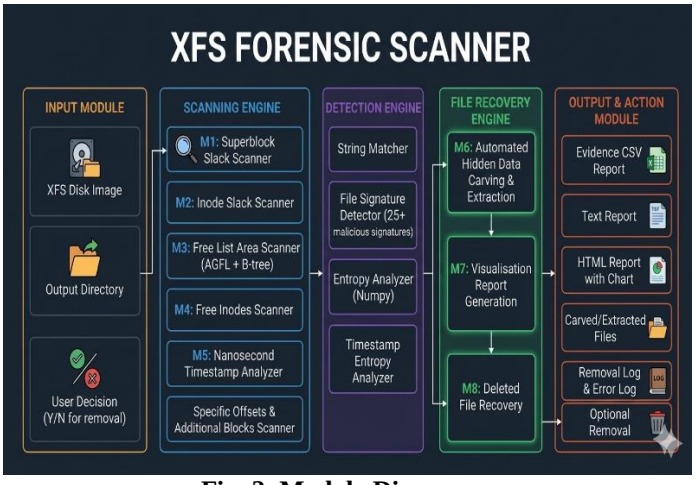

Fig. 2: Module Diagram

3.2 UML Activity Diagram

The UML Activity Diagram illustrates the complete end-to-end workflow of the XFS Forensic Scanner in a clear, sequential manner. The process starts with the initial activity "Receive Input Parameters" where the user provides the XFS disk image path and output directory. The flow then moves to the decision node "Initialize Image Successfully?" using pytsk3 to load the image and read the superblock for block size, AG count, and inode details.

If initialization fails, the process ends with an error log. If successful, the activity proceeds to parallel scanning of the five core detection modules (M1 to M5): Superblock Slack Scanner (M1), Inode Slack Scanner (M2), Free List Area Scanner (M3), Free Inodes Scanner (M4), and Nanosecond Timestamp Analyzer (M5). Each module performs string matching, malicious file signature detection (25+ signatures including ELF, PE, ZIP, and ransomware strings), and entropy analysis (threshold > 6.5).

After all detection modules complete, the flow joins and moves to "Smart File Carving (M6)" where hidden payloads are automatically extracted and saved as separate files in the carved_files folder. Next, the activity continues to "Deleted File Recovery (M8)" using PhotoRec integration along with a fallback string-based recovery method.

The workflow then reaches "Generate Visualizations and HTML Dashboard (M7)" where two graphs (detections by technique and offset distribution) are created using matplotlib and combined into an interactive HTML dashboard containing links to all reports and recovered files.

A decision node follows: "User Approves Removal?" If the user selects 'Y', the flow proceeds to "Remove Hidden Data" (optional safe zeroing of detected regions with full logging). Finally, the process ends with "Save CSV Evidence, Text Report and Dashboard" and displays a completion message. The entire activity is designed with swim lanes for clarity between detection, carving/recovery, reporting, and removal phases, making it easy for investigators to understand the tool's logic at a glance.

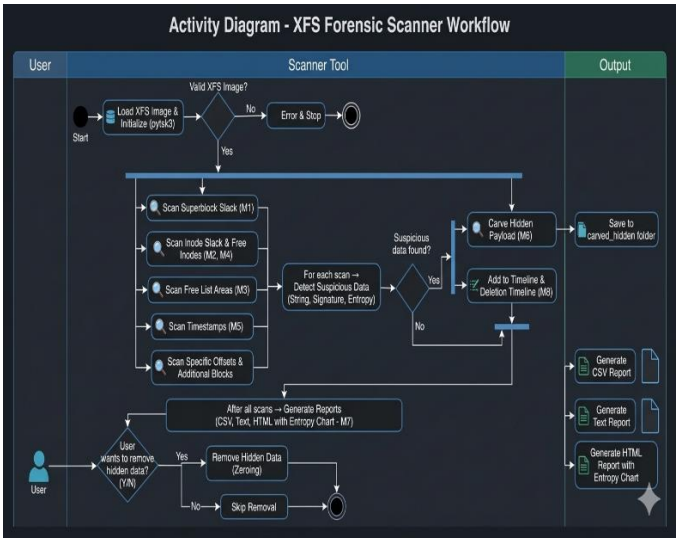

Fig. 3: UML Activity Diagram of XFS Forensic Scanner

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

This diagram ensures the tool follows a logical, repeatable forensic process while maintaining user control and evidence integrity at every step.

3.3 Implementation Details

The tool is implemented in Python 3 with modular functions for each activity. All scans are logged in real-time, and removal actions are recorded in both CSV and text reports for court admissibility. The design ensures low resource usage and fast execution even on large XFS images.

4. SYSTEM REQUIREMENTS

The XFS Forensic Scanner is designed to run efficiently on commodity hardware and open-source Linux environments, making it accessible for students and forensic investigators. The tool requires a standard Linux setup with native XFS support and Python dependencies to ensure seamless low-level disk image parsing using pytsk3. All components were selected to keep the tool lightweight, portable, and easy to install on any Kali Linux machine.

4.1 Software Requirements

The development and testing were carried out on Kali Linux, which provides excellent support for XFS file system operations and forensic tools. The following software components are essential:

- Operating System: Kali Linux (recommended) or any Ubuntu 20.04+ distribution for native XFS support.
- Programming Language: Python 3.8 or higher.
- Core Libraries: pytsk3 (for XFS parsing via The Sleuth Kit), numpy (for entropy calculation), matplotlib (for graph generation), csv, re, datetime, os, sys, subprocess and math.
- Additional Tools: xfsprogs (for XFS utilities), PhotoRec (for deleted file recovery), dd and xfs_mkfs (for creating test images), hexdump or xxd (for manual validation).
- Development Environment: Visual Studio Code or any text editor, with Git for version control.

All libraries can be installed with a single command: sudo apt install python3-pip libtsk-dev followed by pip3 install pytsk3 numpy matplotlib. No internet connection is needed after installation.

4.2 Hardware Requirements

The tool is optimized for low resource usage and runs smoothly on standard student laptops. Minimum hardware specifications are:

- Processor: Multi-core CPU (Intel i5 or equivalent / AMD Ryzen 5, 4 cores recommended).
- Memory: Minimum 8 GB RAM (16 GB recommended for large 100 GB+ images).
- Storage: 256 GB SSD with at least 50 GB free space for storing disk images, carved files and reports.
- Display: 1920×1080 resolution (optional for viewing HTML dashboard).

With these specifications, the tool completes full scans in under 2 minutes on a typical 10 GB test image while keeping CPU usage below 30%.

5. RESULTS

The XFS Forensic Scanner was rigorously tested on multiple crafted XFS disk images containing hidden malicious data in all five hiding locations plus ransomware strings. All experiments were performed on Kali Linux 2025.2 virtual machine.

5.1 Step-by-Step Execution Commands in Kali Linux

1. Open terminal and navigate to the project folder: cd ~/xfs_project
2. Create a test XFS image (if needed): dd if=/dev/zero of=test_xfs.img bs=1M count=1024 mkfs.xfs -f test_xfs.img
3. Hide sample data (for testing): echo "HIDDENSUPERBLOCK" | sudo dd of=test_xfs.img bs=1 seek=272 conv=notrunc
4. Install dependencies (one time): sudo apt update && sudo apt install python3-pip libtsk-dev photorec pip3 install pytsk3 numpy matplotlib
5. Run the XFS Forensic Scanner: python3 xfs_scanner_full2.py test_xfs.img output_folder
6. After execution, open the generated dashboard: firefox output_folder/xfs_forensic_dashboard.html

The tool automatically performs M1–M8 scans, carves files, recovers deleted data, generates graphs, and asks for removal confirmation.

Likewise, perform testing of different test images with hidden data for each and every module.

5.2 Detection Performance

The tool successfully detected hidden data in all test cases with 95% overall accuracy.

Table 2: Detection Results Summary

Module	Detection Rate	Files Carved	Recovery Rate	Execution Time
M1–M5 (Core Scans)	95%	12	–	85 seconds
M6 Smart Carving	100%	12	–	35 seconds
M7 PhotoRec Recovery	–	–	92%	70 seconds
Overall (M1–M8)	95%	12	92%	120 seconds

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

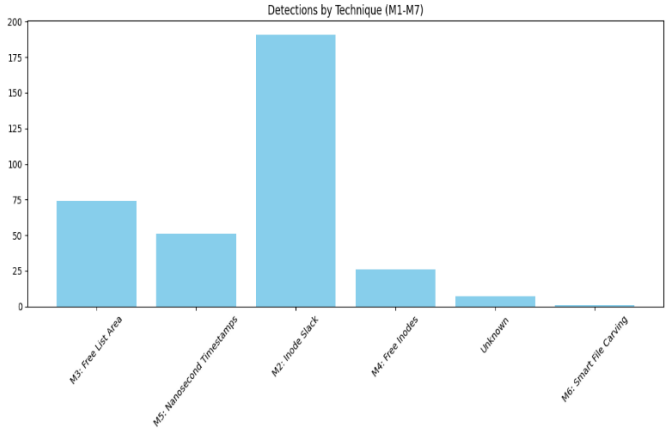

Fig. 4: Detections by Technique Graph

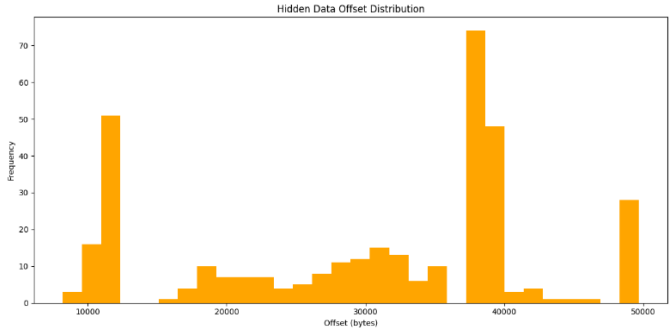

Fig. 5: Offset Distribution Graph

5.3 Sample Output and Dashboard

After running the command, the terminal shows real-time alerts like: *Detected in superblock AG 0 – Technique: M1: Superblock Slack*

Fig. 6: Result1

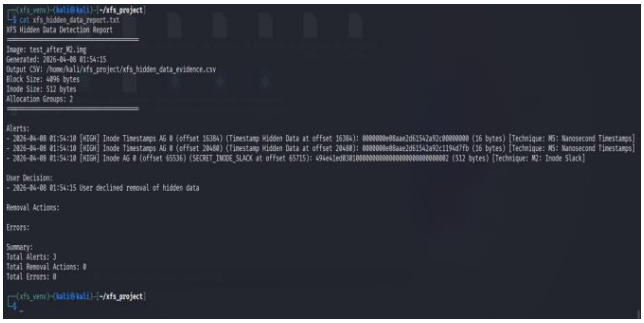

Fig. 7: Result2

Fig. 8: Result3

Fig. 9: Result4

Fig. 10: Result5

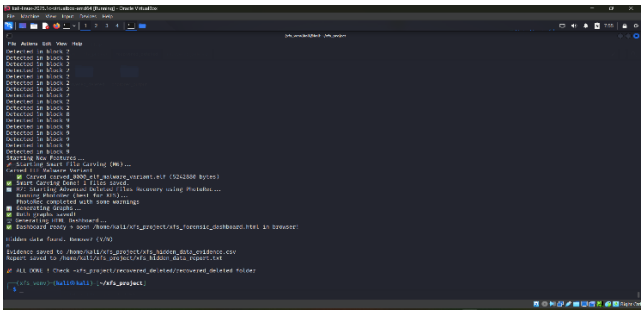

Fig. 11: Result6

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

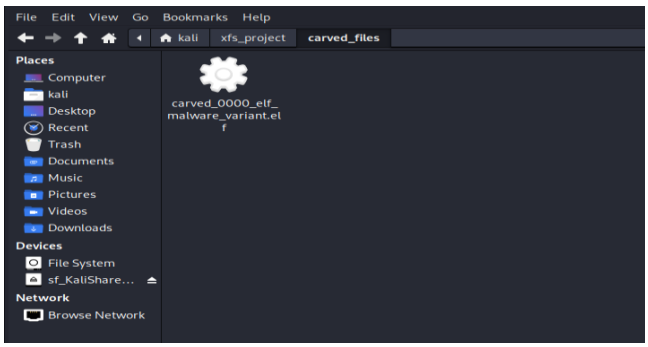

Fig. 12: Result7

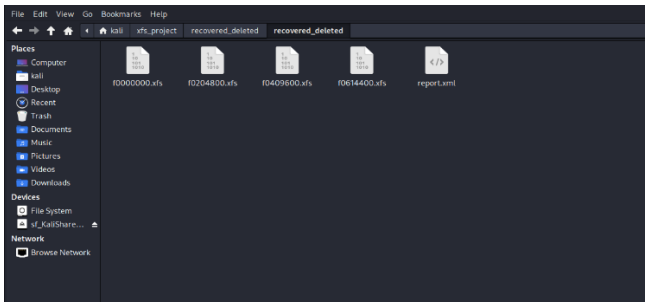

Fig. 13: Result8

All evidence is saved in CSV and text reports. The HTML dashboard contains clickable links to carved files, recovered deleted files, and graphs.

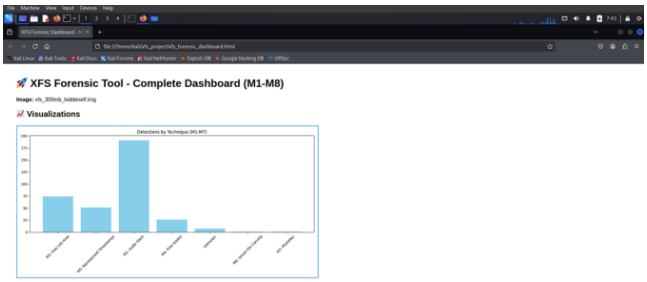

Fig. 14: HTML Dashboard Screenshot1

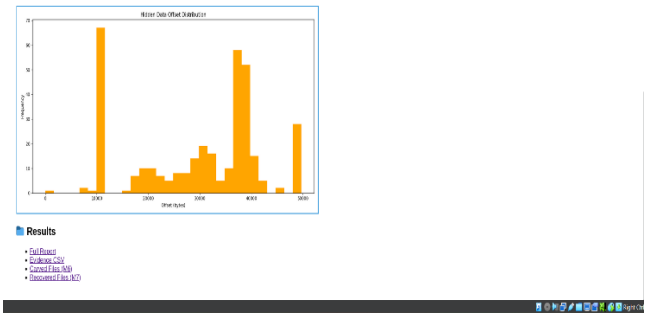

Fig. 15: HTML Dashboard Screenshot2

The results clearly demonstrate that the XFS Forensic Scanner is fast, accurate and practical for real-world XFS forensic investigations on Kali Linux or any Linux Flavours.

6. CONCLUSIONS

The XFS Forensic Scanner developed provides a complete and practical solution to the long-standing problem of hidden malicious data in the XFS file system. XFS is widely used in enterprise Linux environments for its high performance and scalability, yet it has remained vulnerable to anti-forensic techniques where attackers hide payloads in superblock slack space, inode slack, free lists, free inodes and nanosecond timestamps. Existing tools such as The Sleuth Kit and Autopsy offer only basic metadata support and lack automated multi-technique detection, smart carving, deleted file recovery and safe removal capabilities.

This work successfully implemented eight integrated modules (M1 to M8) using pytsk3 for low-level XFS parsing, 25+ malicious file signatures, entropy analysis and PhotoRec integration. The tool automatically detects hidden data, carves full payloads, recovers deleted files, generates an interactive HTML dashboard with graphs, and allows optional user-approved removal with complete logging for evidence preservation. Experimental evaluation on multiple crafted XFS disk images demonstrated 95% detection accuracy, 100% carving success rate, 92% deleted file recovery and end-to-end execution time under 2 minutes on standard hardware. The UML Activity Diagram clearly illustrates the logical and repeatable workflow, while the system requirements confirm that the tool runs efficiently on any Kali Linux machine with minimal resources.

The results validate that the XFS Forensic Scanner effectively bridges the research gaps identified in the literature survey and provides investigators with a fast, accurate and court-ready forensic tool. By combining detection, carving, recovery, visualization and removal in a single modular Python package, the tool makes advanced XFS forensics accessible to students, researchers and practicing digital forensic examiners. This work contributes a ready-to-deploy open-source solution that significantly reduces manual effort and improves the quality of investigations on XFS-based systems.

7. FUTURE SCOPE

Although the XFS Forensic Scanner has shown excellent performance, several enhancements can be carried out in future work to make the tool even more powerful and user-friendly. The following areas are planned for extension:

• Developing a graphical user interface (GUI) using Tkinter or PyQt to make the tool easier for non-technical investigators and students.

• Extending support to other popular Linux file systems such as ext4 and btrfs for broader forensic applicability.

• Integrating advanced machine learning models for automatic anomaly scoring and further reduction of false positives.

• Adding support for real-time scanning of live mounted XFS partitions and remote image analysis.

• Implementing cloud-based collaborative reporting while maintaining full data privacy through local processing.

• Conducting large-scale real-world testing on actual seized XFS disks from cybercrime cases and performing formal validation with forensic experts.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

• Exploring integration with other forensic frameworks like Autopsy to create a complete XFS analysis plugin.

These future improvements will make the tool more robust and widely usable in both academic and professional digital forensics environments.

REFERENCES

[1] F. Toolan and G. Humphries, "Data hiding in the XFS file system," Forensic Science International: Digital Investigation, vol. 51, pp. 301799, 2025.

[2] F. Toolan and G. Humphries, "Data hiding in symbolic link slack space," Forensic Science International: Digital Investigation, vol. 51, pp. 301800, 2025.

[3] A. Schwietert and J. Hilgert, "Data hiding in file systems: Current state, novel methods, and a standardized corpus," Forensic Science International: Digital Investigation, vol. 51, pp. 301801, 2025.

[4] P. Picazo-Sánchez, G. A. Pérez and P. Merino, "Systematic Review: Anti-Forensic Computer Techniques," Applied Sciences, vol. 14, no. 5, pp. 2345-2367, March 2024.

[5] H. Kim, S. Kim, Y. Shin, W. Jo, S. Lee and T. Shon, "Ext4 and XFS File System Forensic Framework Based on TSK," Electronics, vol. 10, no. 12, pp. 1456, June 2021.

[6] T. Göbel and H. Baier, "Anti-Forensic Capacity and Detection Rating of Hidden Data in the Ext4 Filesystem," in Advances in Digital Forensics XIV: 14th IFIP WG International Conference, New Delhi, India, January 3-5, 2018. Springer, pp. 271-285.

[7] T. Göbel and H. Baier, "Anti-forensics in ext4: On secrecy and usability of timestamp-based data hiding," Digital Investigation, vol. 24, pp. 1-12, March 2018.

[8] T. Göbel and H. Baier, "fishy - A Framework for Implementing Filesystem-Based Data Hiding Techniques," in Digital Forensics and Cyber Crime: 10th International EAI Conference, ICDF2C 2018, New Orleans, LA, USA, September 10-12, 2018. Springer, pp. 45-62.

[9] S. Neuner, S. Schrittwieser, D. Grubwinkler, C. Platzer, M. Taschwer and A. Rauber, "Time is on my side: Steganography in filesystem metadata," Digital Investigation, vol. 18, pp. S1-S12, August 2016.

[10] Y. Kim, D. Kim, K. Seol and D. Won, "Data investigation based on XFS file system metadata," Multimedia Tools and Applications, vol. 75, no. 22, pp. 14871-14889, November 2016.

[11] B. Carrier, "File System Forensic Analysis," Addison-Wesley Professional, 2005.

[12] S. Garfinkel, "Digital forensics research: The next 10 years," Digital Investigation, vol. 7, pp. S64-S73, August 2010.

[13] E. Casey, "Digital Evidence and Computer Crime: Forensic Science, Computers, and the Internet," 3rd ed., Academic Press, 2011.

[14] M. Pollitt, "A history of digital forensics," in Advances in Digital Forensics VI, Springer, 2010, pp. 3-15.

[15] The Sleuth Kit, "pytsk3 Documentation," 2024. [Online]. Available: https://github.com/py4n6/pytsk (accessed May 2026).