Digital Analysis, Retrieval, and Tracing Forensic Toolkit: A Digital Forensic Analysis Platform

1st Muhammad Usama Tayyab
Department of Information and Communication Engineering
The Islamia University of Bahawalpur
Bahawalpur, Pakistan
muhammadusamatayyab4@gmail.com

2nd Muhammad Adeel
Department of Information and Communication Engineering
The Islamia University of Bahawalpur
Bahawalpur, Pakistan
muhammad.adeel0769@gmail.com

3rd Muhammad Ahmad Masood Ul Hassan
Department of Information and Communication Engineering
The Islamia University of Bahawalpur
Bahawalpur, Pakistan
muhammadahmadmasoodulhassan@gmail.com

4th Iftikhar Rasheed
Department of Information and Communication Engineering
The Islamia University of Bahawalpur
Bahawalpur, Pakistan
Iftikhar.rasheed@iub.edu.pk

***Abstract*—The Digital Analysis, Retrieval, and Tracing Forensic Toolkit (DART) is an open-source digital forensic platform designed to streamline disk image analysis for investigators. Built with Python, DART Forensic Toolkit integrates pytsk3 for file system navigation, libewf-python for `E01/SMART` image handling, and VirusTotal API for malware detection, offering a cost-effective alternative to comprehensive forensic platforms like Autopsy and FTK Suite. It supports file carving, metadata extraction, registry analysis, and image verification, achieving 90% recovery of deleted files in 1–2 minutes and Image verification in under 2 minutes. DART Forensic Toolkit's intuitive interface and accessibility make it ideal for academic, law enforcement, and small-scale forensic investigations.**

***Index Terms*—Digital Forensic, Forensic Toolkit, File Carving, Forensic Analysis, Malware Detection**

I. Introduction

The growing complexity of cybercrime demands forensic tools that efficiently analyze diverse digital evidence while remaining accessible to investigators. Comprehensive forensic platforms like Autopsy and FTK Suite are powerful but often face challenges such as steep learning curves or high costs. The Digital Analysis, Retrieval, and Tracing Forensic Toolkit (DART) addresses these issues by providing an open-source, user-friendly platform for disk image analysis.

DART's technical contributions include its modular architecture, leveraging pytsk3 for robust file system access, libewf-python for E01/SMART image support, and PySide6 for an intuitive graphical user interface (GUI). Unlike Autopsy, which can overwhelm novice investigators with its feature density, DART prioritizes simplicity. Compared to FTK Suite's commercial licensing and resource-intensive design, DART offers cost-free access and lightweight operation. Unlike Plaso's timeline-centric approach, DART integrates real-time malware detection via VirusTotal API, enhancing its investigative scope. This paper details DART's development, features, and performance, positioning it as a versatile forensic solution.

II. Related Work

Digital forensics is an evolving discipline shaped by the need to address complex data and advanced cyber threats. This section reviews significant research that influenced the development of the DART Forensic Toolkit, emphasizing open-source frameworks, usability enhancements, and contemporary forensic challenges.

Carrier [1] promotes the use of open-source tools like The Sleuth Kit (TSK), which inspired DART's integration of pytsk3 for dependable file system navigation. Hariyani et al. [4] demonstrate the effectiveness of Python-based tools for forensic evidence collection from Windows hosts, supporting DART's Python-centric design. Research by Fakhouri et al. [16] underscores the necessity for tools to manage emerging technologies, driving DART's support for diverse image formats including E01, raw, and ISO. Studies on file carving by Sethi [5], Al Zaabi and AlShibli [6], and Fatima and Azeem [11] shaped DART's approach to recovering deleted files, incorporating robust error handling for invalid inodes. Malware analysis research [7], highlights the importance of APIs like VirusTotal, which DART employs for real-time threat detection.

Usability is a critical factor in forensic tool design. Safie [10] provides a comprehensive review of digital forensic investigation models, emphasizing intuitive interfaces to reduce training time, which influenced DART's PySide6-based GUI. Patel and Sharma [8] and Ölvecký and Host'ovecký [14] highlight the role of metadata in investigations, prompting DART's capability to extract EXIF data from images. Cross-platform compatibility remains a challenge, as noted by Fakhouri et al. [16], motivating DART's goal to support Windows, Linux, and macOS, though image mounting is currently limited to Windows. Ali [12] and Rani and Jain [13] emphasize registry and image verification techniques in Windows forensics, inspiring DART's registry viewer and hash verification features.

DOI: https://doi.org/10.67535/tsp.000001.016
Published under the ICACNC 2025 Proceedings

A. Acquisition and Analysis

Acquisition involves collecting and preserving digital evidence, such as creating forensic images, while analysis examines the data to uncover insights, like file recovery. The distinction between FTK Imager, a lightweight tool for creating and previewing forensic images, and FTK Suite, a comprehensive platform for advanced analysis and reporting [2].

Fig. 1. Overall procedure of extraction and Analysis in digital forensics

III. Overview and Features

DART is a Python-based forensic toolkit designed for efficient disk image analysis across Windows, Linux, and macOS. It supports a range of forensic tasks through a modular architecture and an intuitive GUI, making it accessible to investigators of varying skill levels.

A. Key Features

- **File Carving**: Recovers deleted files, with improved inode handling to avoid crashes.
- **Detailed File Analysis**: View file content in different formats, such as HEX, text, and application-specific views.
- **EXIF Data Extraction**: Extract and display EXIF metadata from photos.
- **Image Verification**: Verifies image integrity using MD5/SHA1 hashes.
- **Format Conversion**: Convert `E01` disk images to raw format.
- **Registry Viewer**: View and examine Windows registry files.
- **Tree Viewer**: Navigate through the disk image structure, including partitions and files.
- **VirusTotal Integration**: Check files for malware using the Virus Total API.
- **Image Mounting**: Mounts images using Arsenal Image Mounter (Windows only).

B. Supported Formats

TABLE I
SUPPORTED IMAGE FORMATS IN DART FORENSIC TOOLKIT

Image Format	Extensions	Split	Unsplit
EnCase® Image File (EWF)	`.E01, .Ex01`	✓	✓
SMART/Expert Witness Image File	`.s01`	✓	✓
Single Image Unix/Linux DD/Raw	`.dd, .img, .raw`	✓	✓
ISO Image File	`.iso`	–	✓
AccessData Image File	`.ad1`	✓	✓

Table I outlines the supported forensic image formats in the DART toolkit. The tool is compatible with widely used formats such as EnCase `E01/Ex01`, SMART `s01`, `raw/DD` formats, `ISO`, and AccessData `AD1`, supporting both split and unsplit image structures where applicable.

IV. Methodology

Dart's development followed a systematic approach to meet forensic investigation needs.

A. Requirement Analysis

Based on research [1, 4, 10] and analysis of Autopsy and FTK Suite, DART's requirements include:

- Support for forensic image formats (E01, raw, ISO, SMART, AD1).
- Cross-platform compatibility (Windows, Linux, macOS).
- An intuitive GUI for academic learning and investigators of all skill levels
- Features like file carving, registry analysis, metadata extraction, and malware scanning.

B. System Design

DART's system architecture comprises:

- **Core Engine**: Uses pytsk3 for file system navigation and libewf-python for converting `E01` disk images to raw format, with logic to handle deleted file inodes, and Arsenal Image Mounter for Windows-based image mounting.
- **GUI Layer**: Built with PySide6, featuring a tree viewer, analysis tabs (HEX, text), and registry browser, with deleted files marked in gray.
- **External Integrations**: Incorporates VirusTotal API for malware detection.

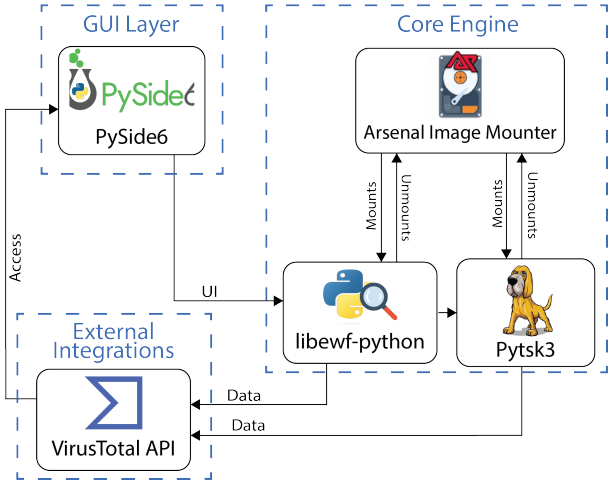

Fig. 2. DART system architecture, detailing interactions among PySide6, pytsk3, VirusTotal API, libewf-python, and Arsenal Image Mounter.

C. System Flow

The Fig. 3. illustrates the overall flow of the system begins with initializing the interface, setting up the GUI and managers, followed by two primary paths: loading a forensic image by opening an image file or converting `E01` image to `DD/RAW` format. After loading, the tool navigates the disk structure to explore the hierarchy, allowing investigators to mount or unmount the disk or select evidence for analysis. The analysis phase enables viewing and analyzing data, with verification steps including checking image hashes for integrity and using the VirusTotal API for malware analysis. Finally, the process concludes with exporting results, saving files or directories as needed.

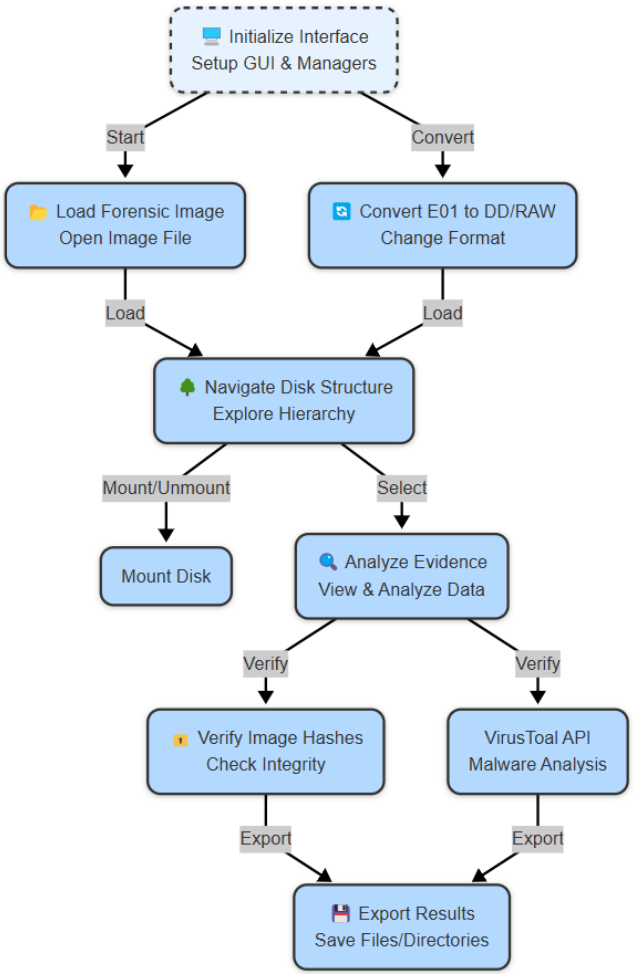

Fig. 3. System Flow of the DART Forensic Toolkit

D. Technology Stack

We utilized a strong suite of technologies to guarantee DART Forensic Tool's reliability and efficiency.

- **pytsk3**: Handles file system access and file carving, with custom exception handling for invalid inodes (e.g., 'ValueError: invalid literal for int()').
- **libewf-python**: Processes E01/SMART images, supporting split/unsplit formats and MD5/SHA1 hash verification.
- **PySide6**: Powers the GUI with tree views, HEX/text analysis, and registry browsing..
- **Arsenal Image Mounter**: Mounts images on Windows, with plans for cross-platform solutions.

TABLE II
TECHNOLOGY STACK OF THE DART FORENSIC TOOLKIT

Component	Technology	Purpose
Core Engine	pytsk3	File system navigation and file carving.
Image Handling	libewf-python	Convert E01 disk images to raw format.
GUI	PySide6	User-friendly interface with tree and analysis views.
Image Mounting	Arsenal Image Mounter	Mount forensic disk images.
Malware Detection	VirusTotal API	Real-time malware scanning.

Table II highlights the essential technology and libraries that form the foundation of DART, selected for their ability to address specific forensic tasks while ensuring accessibility and cross-platform potential.

V. IMPLEMENTATION

This section details the implementation of key modules and their workflows, emphasizing how each contributes to the toolkit's overall capabilities.

A. File Carving

The file carving module uses `pytsk3` to recover deleted files from disk images. The workflow begins with scanning the file system for unallocated inodes. The Fig. 4 illustrates the file carving process in the DART Forensic Toolkit. It begins with reading a disk in chunks and checking if a file type is selected. If so, it proceeds to carve files, validate signatures, extract and verify content. If the content is valid, the file and its thumbnail are saved, and the results are updated in a table. The process iterates through signatures and disk chunks until completion or an error is encountered.

B. EXIF Data Extraction

The EXIF data extraction module extracts metadata from image files to provide investigative insights. The Fig. 5 illustrates the step-by-step process of handling image files for EXIF metadata extraction. It begins with file reception and validation, checks for JPEG format, and attempts to extract EXIF data. If tags are successfully retrieved, they are processed and displayed in an HTML table. Errors during any step are logged, and the display is cleared accordingly.

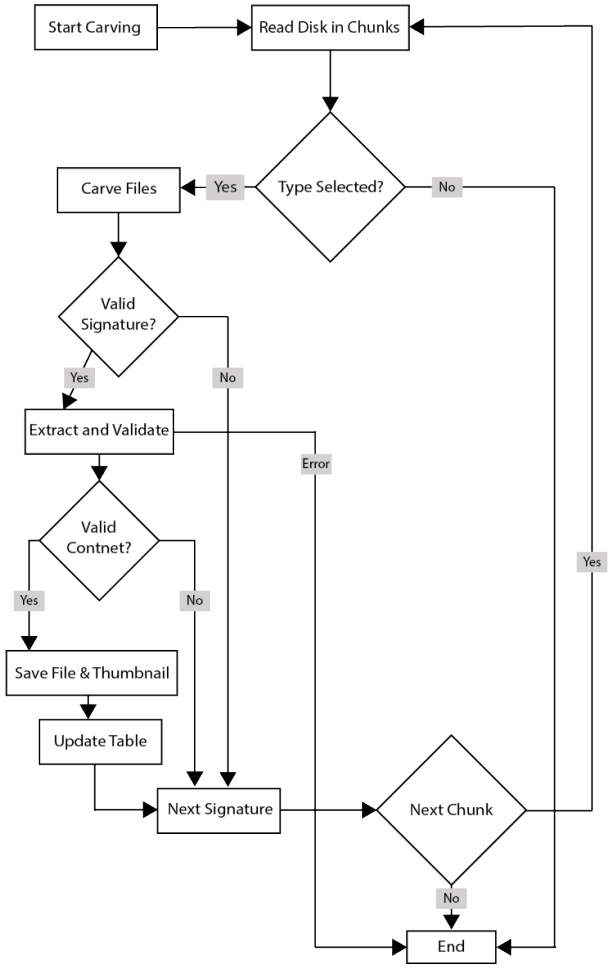

Fig. 4. File carving process in DART, recovering deleted files with inode error handling

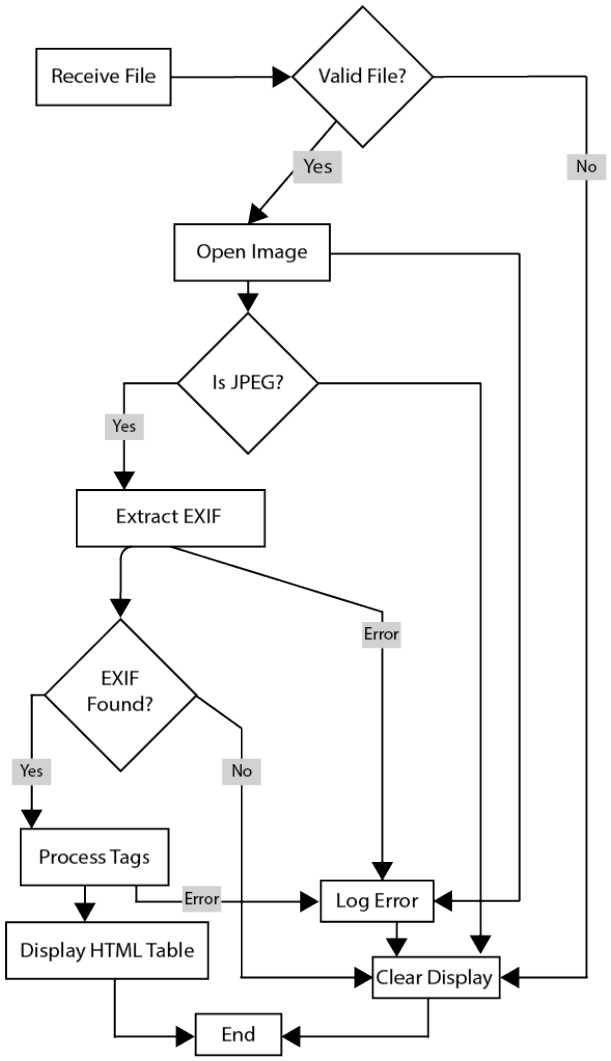

Fig. 5. EXIF data extraction process in DART, displaying metadata from images.

C. E01 Format Conversion

The E01 format conversion module enables analysis of `E01` images by converting them to `RAW` & (`dd`) format. The Fig. 6 illustrates the procedure to convert an EnCase (`E01`) image to `DD/RAW` format. Depending on the source type—image file or physical drive—the user selects relevant inputs, output formats, and directories. The process continues to conversion and input validation, with the final step being the generation of the output file or an error message if the input is invalid.

D. Hash Verification of Image Integrity

The hash verification module ensures evidence integrity by computing and comparing MD5/SHA1 hashes of images. The Fig. 7 depicts the verification process of disk image integrity using hash values. The system calculates hashes and identifies if the image is of the EWF format. It either compares stored vs. computed hashes or simply retrieves the calculated values. The results are displayed along with file size and path, with options to save or copy the hash.

VI. Evaluation

DART was evaluated to assess its functionality, compatibility, and performance across different platforms and image formats.

A. Testing

1) Image Formats: The tool has primarily been tested with `dd` and `E01` image files. These formats are well-supported, and the system handled them consistently during extraction, hashing, and conversion operations.

2) Cross-Platform Compatibility:

a) Windows (10 & 11): On Windows 10 & 11, DART was tested using Arsenal Image Mounter for disk image mounting. All features—including file carving, hash verification, and malware scanning—performed as expected. The graphical user interface rendered smoothly, and no visibility issues were observed in dark mode.

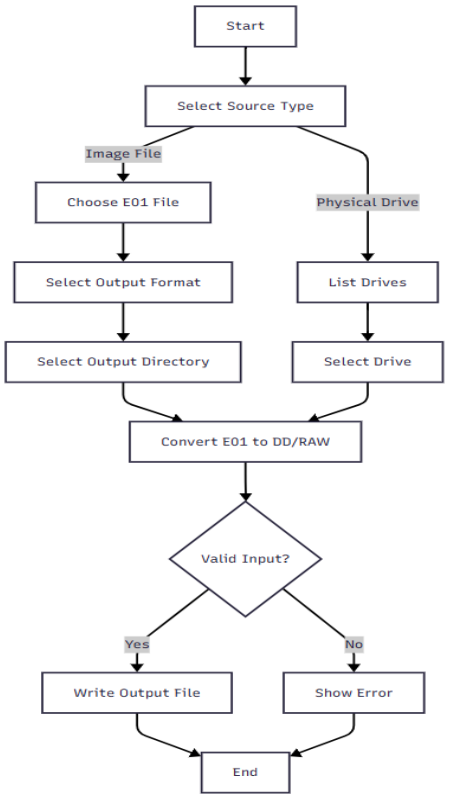

Fig. 6. E01 to Raw Format Conversion Process in DART

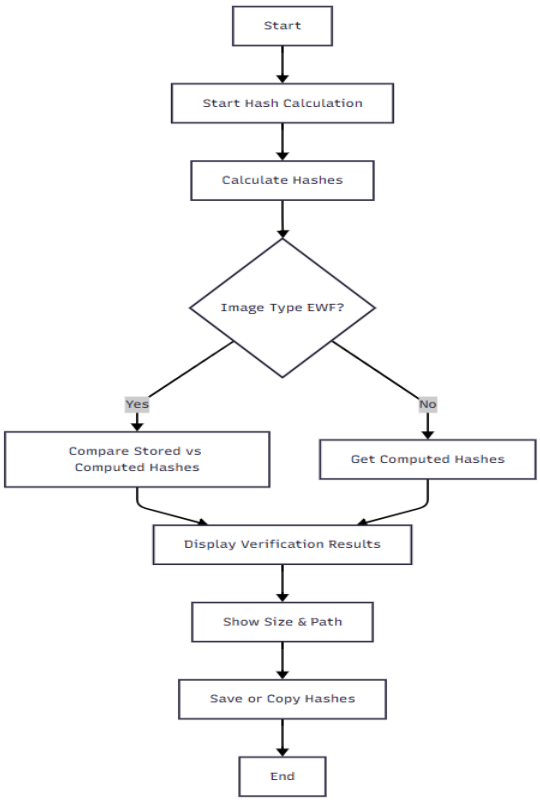

Fig. 7. Hash verification process in DART, ensuring evidence integrity.

b) Kali Linux: On Kali Linux, DART operated successfully with all core functionalities. However, disk image mounting is currently unsupported due to its reliance on Arsenal Image Mounter, a Windows-only utility.

Fig. 8 depicts the DART Forensics Toolkit in operation. The interface displays successful file carving from a disk image, highlighting its efficiency and user-friendly GUI during forensic analysis.

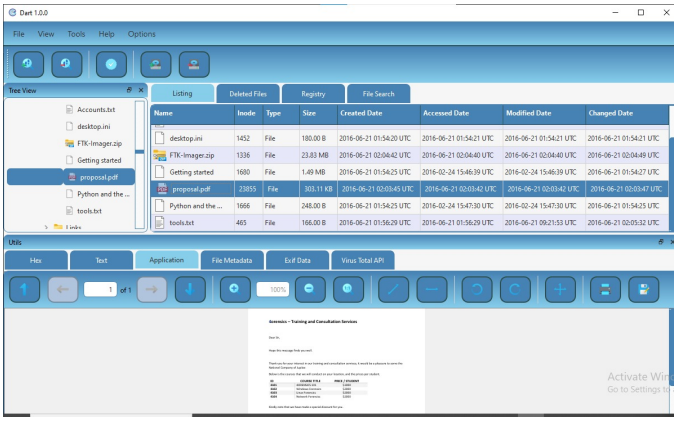

Fig. 8. DART Forensic Toolkit running and performed successful file carving

B. Performance

Performance tests were conducted on 1–15 GB `dd` and `E01` with NTFS file systems. Key results include:

1) File Carving: A recovery rate of approximately 90% was achieved for deleted data from a 14.3 GB disk image, as demonstrated in Fig. 8 and Fig. 9

2) Detailed File Analysis: View file contents in multiple formats including HEX, plain text, and application-specific views to support in-depth forensic analysis.

3) E01 Format Conversion: A 14.3 GB disk image was successfully converted from `E01` to `dd` format.

4) Hashes Verification: Verified integrity of 14.3 GB disk image under 2 minutes using MD5/SHA1/SHA256 hashes as demonstrated in Fig. 9

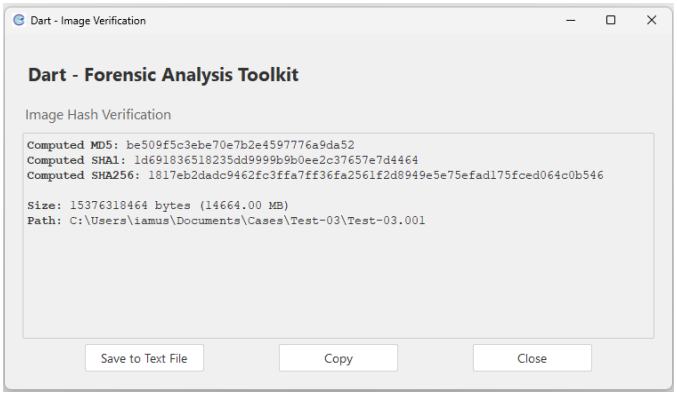

Fig. 9. Image Hashes Verification in DART Forensic Toolkit

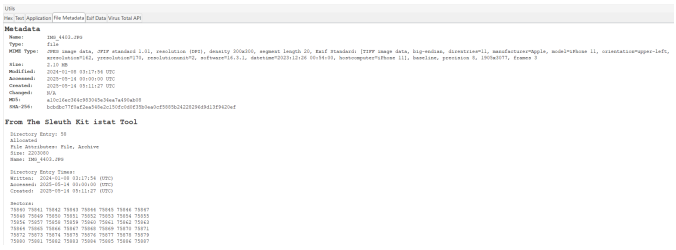

Fig. 10. Meta-data of JPEG file retrieved using DART Forensic Toolkit

TABLE III
COMPARISON OF FORENSIC ANALYSIS PLATFORMS

Metric	DART	Autopsy	FTK Suite
File Carving (15 GB)	∼1–2 min, recovery up to 90%	∼3–5 min, recovery ∼90%	∼3–5 min, recovery ∼92%
Image Verification	Under 2 min	2–3 min (built-in hash analysis)	1–2 min
Cross-Platform Support	Partial (Windows/Linux; Mac not yet tested)	Yes (Java-based; Windows/Linux/Mac)	Windows-only
Ease of Use	High (lightweight GUI, minimal setup)	Moderate (feature-rich, steeper learning curve)	High (professional UI, comprehensive features)
Cost	Free (Open-source)	Free	Commercial

5) Meta-data: The metadata of a JPEG file was successfully retrieved, as shown in Fig. 10.

VII. RESULTS & CONCLUSION

DART demonstrates strong performance as a lightweight, open-source forensic toolkit, achieving a 90% recovery rate for deleted files in 1–2 minutes and Image verification in under 2 minutes. Its modular design, leveraging pytsk3 and libewf-python, ensures efficient disk image analysis, while the VirusTotal API integration adds real-time malware detection capabilities. Compared to Autopsy's complexity and FTK Suite's commercial model, DART offers a balanced solution with an intuitive GUI, cross-platform potential, and accessibility for resource-constrained investigators. These results position DART as a valuable tool for academic and small-scale forensic investigations.

VIII. FUTURE SCOPE

DART's roadmap includes several enhancements to broaden its capabilities:

- **Mobile Forensics Support**: Extend DART Forensic Tool to analyze mobile device images, addressing the growing prevalence of mobile-based evidence.
- **Cross-Platform Image Mounting**: Image mounting currently works only on Windows using the Arsenal Image Mounter executable. The aim is to make this feature work across all platforms.
- **Advanced Automation**: Integrate machine learning for evidence prioritization and anomaly detection, streamlining large-scale investigations.

REFERENCES

[1] B. Carrier, "The Sleuth Kit and Autopsy: Open Source Digital Forensics Tools," 2023. [Online]. Available: https://www.sleuthkit.org/

[2] AccessData, "Forensic Toolkit (FTK) Suite User Guide," 2024. [Online]. Available: https://www.accessdata.com/products-services/forensic-toolkit-ftk

[3] OpenText, "EnCase Forensic," 2024. [Online]. Available: https://www.opentext.com/products/encase-forensic

[4] A. Hariyani, J. Undaiva, N. Vaidya, and A. Patel, "Forensic evidence collection from Windows host using Python-based tool," in *Proc. Int. Conf. Comput. Commun. Mach. Learn. (ICCCMLA)*, 2022, pp. 85–90. doi: 10.1109/ICCCMLA56841.2022.9989295

[5] P. C. Sethi, "File carving: Analyzing data retrieval in digital forensics," *Int. J. Comput. Inf. Technol.*, vol. 13, no. 3, 2024. doi: 10.24203/cpps7896

[6] M. Al Zaabi and A. S. AlShibli, "A review of JPEG file carving: Challenges, techniques, and future directions," *Appl. Comput. J.*, vol. 5, no. 1, pp. 372–385, 2025. doi: 10.52098/acj.20255124

[7] "A survey on malware analysis techniques: Static, dynamic, hybrid and memory analysis," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 8, no. 4-2, pp. 1662, 2018. doi: 10.18517/ijaseit.8.4-2.6827

[8] P. C. Patel and B. K. Sharma, "Meta data as a part of digital forensic investigation," *Int. J. Sci. Res. Develop. (IJSRD)*, vol. 3, no. 8, p. 392, 2015.

[9] J. Metz, "Plaso: Open source timeline analysis," 2024. [Online]. Available: https://plaso.readthedocs.io/

[10] S. Safie, "A comprehensive review of the evolution and future directions of digital forensic investigation model," *Int. J. Emerg. Technol. Adv. Eng.*, 2023. doi: 10.46338/IJETAE0723_01

[11] F. Fatima and E. Azeem, "Data carving – The art of retrieving deleted data as evidence," *LGU Int. J. Electron. Crime Invest. (IJECI)*, vol. 6, no. 2, 2022. doi: 10.54692/ijeci.2022.0602101

[12] M. Ali, "Role of Windows Registry forensics in digital forensics investigation," *Int. J. Electron. Crime Invest.*, vol. 2, no. 3, p. 9, Jun. 2018. doi: 10.54692/ijeci.2018.020318

[13] A. Rani and A. Jain, "Digital image forensics—Image verification techniques," in *Digital Image Processing: Concepts, Methodologies, Tools, and Applications*, Sep. 2020, pp. 221–234. doi: 10.1007/978-981-15-5566-4_19

[14] M. Ölvecký and M. Host'ovecký, "Digital image forensics using EXIF data of digital evidence," in *Proc. 2021 19th Int. Conf. Emerg. eLearning Technol. Appl. (ICETA)*, 2021, pp. 282–286. doi: 10.1109/ICETA54173.2021.9726649

[15] D. Dunsin, M. C. Ghanem, K. Ouazzane, and V. Vassilev, "A comprehensive analysis of the role of artificial intelligence and machine learning in modern digital forensics and incident response," *Forensic Sci. Int.: Digit. Invest.*, vol. 48, 301675, 2024. doi: 10.1016/j.fsidi.2023.301675

[16] H. Fakhouri, M. Alsharaiah, A. Al Hwaitat, M. Alkhalaileh, and F. Dweikat, "Overview of challenges faced by digital forensic," in *Proc. 2024 Int. Conf. Cybersecurity Resilience (ICCR)*, 2024, pp. 1–8. doi: 10.1109/ICCR61006.2024.10532850

[17] Arsenal Recon, "Arsenal Image Mounter (AIM) Walkthrough" 2024. [Online]. Available: https://arsenalrecon.com/arsenal-image-mounter-aim-walkthrough