Development of an Automated Network Forensics Analysis Tool Using PCAP Data

Ashwag Alotaibi, Huda Aldawghan, Abdullah Albuali
Department of Computer Networks and Communications,
College of Computer Sciences and Information Technology, King Faisal University, Al-Ahsa, 31982, Saudi Arabia

***Abstract*—Network forensic investigation often requires analysts to inspect packet captures manually, construct protocol filters, correlate traffic events, and prepare evidence summaries. That workflow is powerful for expert investigators but difficult for novice analysts and users with limited forensic experience. NetForensics was designed to simplify the forensic workflow for such users, however, no formal usability study has yet been conducted. NetForensics is a web-based PCAP/PCAPNG network forensic tool that converts packet-capture evidence into an explainable dashboard and downloadable reports. The system implements a six-phase workflow: packet parsing, feature extraction, rule-based detection, correlation, dashboard visualization, and report generation. The implementation uses Python, Flask, Scapy, Chart.js, and modular detection components for DDoS-like fan-in traffic, port scanning, traffic spikes, protocol dominance, and unusual destination ports. The dashboard presents packet counts, flow counts, unique sources, severity, confidence, packet-rate timelines, protocol distribution, destination-port usage, top sources, and alert evidence. The system exports JSON, CSV, and PDF reports so that the same investigation can be reviewed in machine-readable, tabular, and human-readable formats. Experimental evaluation on six synthetic PCAP scenarios containing 124,561 packets demonstrated that NetForensics successfully parsed all captures and correctly triggered the DDoS, port-scan, traffic-spike, and unusual-port detectors in every intended scenario while generating consistent dashboard metrics and JSON, CSV, and PDF reports. The contribution is a complete PCAP-to-report forensic workflow designed for academic laboratories and first-stage incident triage, with emphasis on transparency, usability, explainability, and reproducible source code rather than unsupported benchmark claims.**

Keywords*—*Network forensics; PCAP; PCAPNG; incident response; DDoS detection; port scanning; traffic anomaly detection; Flask; Scapy; digital forensics; dashboard visualization

I. Introduction

Modern organizations rely on digital networks for cloud services, remote access, business communication, and internal operations. This dependency increases exposure to reconnaissance scans, distributed denial-of-service (DDoS) activity, brute-force attempts, botnet communication, and abnormal traffic bursts. Network forensics supports incident response by preserving and analyzing communication traces so that investigators can reconstruct suspicious behavior, identify affected hosts, and document evidence for decision-making [1], [4], [5].

The project uses packet-capture evidence because raw capture files preserve low-level network observations that are often lost in summarized logs. PCAP and PCAPNG records can contain packet timestamps, source and destination IP addresses, transport ports, protocol values, packet lengths, and TCP flag behavior. These fields are directly related to the suspicious behaviors targeted by the system: DDoS-like fan-in patterns require source and destination concentration; port scanning requires repeated port or host probing; traffic spikes require time-window packet counts; and unusual service activity requires destination-port evidence. Recent work on network forensics and intrusion detection confirms the importance of packet and flow features for reconstructing network events and detecting abnormal behavior [2], [3], [13], [29].

A second reason for choosing PCAP/PCAPNG is practical compatibility. PCAP remains the standard output format for many capture tools, while PCAPNG is commonly used by modern packet analyzers because it can support richer capture metadata. Supporting both formats allows the project to accept files produced by common laboratory and operational tools. This is important because the project is not only a detector: it is a web-based investigation workflow. The webpage is the user's entry point, but the forensic value comes from what happens after upload: the server parses the capture, extracts features, raises alerts, correlates the findings, displays the dashboard, and generates reports.

Fig. 1 shows the implemented upload stage. The user selects a PCAP or PCAPNG file through the webpage. The Flask server validates the extension, creates a session identifier, saves the file, and starts the analysis pipeline. This design gives beginner users a guided route from raw network evidence to readable findings without requiring them to manually inspect every packet in Wireshark.

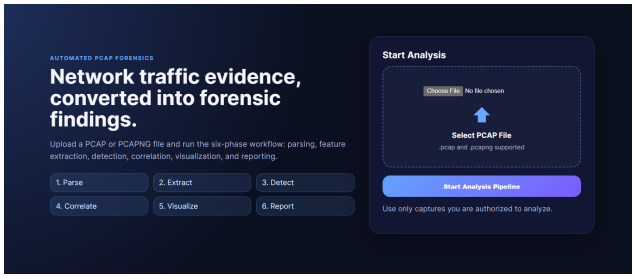

Fig. 1. Web upload interface and relation to the forensic workflow. The webpage is the front-end entry point; the actual analysis is performed by the server-side six-phase pipeline.

Also, as shown in Fig. 1, the web-based upload interface for NetForensics, which serves as the first step towards forensic analysis. The users have the ability to upload their PCAP or PCAPNG files using the interface provided, and they can begin the analysis with just one click. After uploading the file, there is automatic verification of the file, generation of a unique session ID, and storage of the raw capture data.

TABLE I. FOUR OBJECTIVES AND HOW THEY WERE ACHIEVED

Objective	Implementation achievement
PCAP/PCAPNG parsing	Implemented a Scapy-based parser that reads uploaded captures and extracts timestamp, source IP, destination IP, ports, protocol, length, and TCP flags. The parser also groups packets into flow-like communication records.
Feature extraction	Implemented packet, host, port, protocol, and timeline feature extraction. The tool calculates total packets, flows, unique sources, unique destinations, packet-rate buckets, top talkers, destination-port frequencies, and protocol distribution.
Suspicious behavior detection	Implemented rule-based detectors for DDoS-like fan-in behavior, SYN-heavy flooding, vertical and horizontal port scans, traffic spikes, protocol dominance, and unusual ports. Each alert includes evidence and recommendations.
Dashboard and reporting	Implemented a Flask dashboard with metric cards, charts, correlation explanation, alert table, and downloads. The report generator exports JSON, CSV, and PDF evidence files.

Upon completion of the above procedures, the server will automatically trigger the analysis process, which comprises parsing, feature extraction, detection, correlation, visualization, and reporting.

II. PROBLEM STATEMENT AND MOTIVATION

The traditional approach to network forensics using tools like packet analyzers and security monitors provides comprehensive information; however, these traditional solutions demand that the user knows what evidence to look for, how to filter the evidence, and how to interpret it. These challenges create an accessibility problem for learners, newcomers, and small labs that cannot afford professional analysts. The student could see the packet flow but not know how to analyze whether it involves scanning, flooding, or normal communication. Likewise, an incident response analyst may want a fast summary view before determining whether he/she needs a deeper forensics analysis.

While many research studies of network IDS report very accurate models of attack detection, the majority of them are more focused on achieving high classification accuracy rather than explaining how the classification was done. There are several recent studies discussing problems of class imbalance, feature reliability, and dataset construction that may affect intrusion-detection performance [7], [11]–[13], [27], [28]. In [16], the authors also explore adversarial data generation for improving intrusion-detection performance under imbalanced training conditions. NetForensics can serve as an example of an accessible, explainable forensic tool designed to explain suspicious activities identified based on PCAP/PCAPNG files.

III. RESEARCH OBJECTIVES AND ACHIEVEMENT

The report highlights four goals for the implementation process. Table I details how each goal was addressed in the implementation process.

IV. CONTRIBUTION

The main contributions are:

- End-to-end forensic investigation workflow development to automatically convert raw PCAP/PCAPNG artifacts into forensic reports.
- Modular six-step architecture of packet parsing, feature extraction, behavioral detection, evidence correlation, visualization, and report generation, thereby enabling individual steps to progress independently.
- Explainable forensic reasoner based on rules to produce understandable evidence statements, not mere detection decisions.
- Evidence traceability framework to preserve consistency between uploaded packet captures and forensic reports by means of session tracing, meta-information collection, and cryptographic hashing.
- User-friendly forensic system aimed at academic labs and initial stage DFIR forensics.

V. RELATED WORK OR LITERATURE REVIEW

A. Overview of Existing Work

As per recent studies, there are three interesting directions: Firstly, network forensics focuses on evidence acquisition, packet analysis, and reconstruction of events [1], [3]–[6]. Secondly, intrusion detection studies reveal that the features of the traffic may assist in DDoS, DoS, reconnaissance, and anomaly detection, provided proper preprocessing and evaluation techniques are used [10], [11], [24]–[26]. Lastly, a recent survey reveals the application of ML and DL, although with caution, as there are issues regarding datasets [7], [19], [20], [29], [30]. Table II presents the comparison of recent work relevant to the project.

DDoS and DoS detection remains a major research topic because attack behavior can be volumetric, distributed, protocol-specific, or application-layer oriented. In recent research on DDoS and IoT security, statistical techniques, ML/DL classification techniques, botnet analysis, and deployment issues have been considered [8], [9], [21]–[23]. This effort is not claimed to be a substitute for a complete IDS system. Instead, it employs heuristic algorithms to detect patterns indicative of attacks, including many-to-one source attacks, anomalous spikes in packet rates, and SYN flood attacks.

Explainability is an additional topic of relevance. Studies on explainable AI applied to intrusion detection reveal that analysts need explanations for why something was detected, rather than just the outcome itself [17], [18]. NetForensics applies the same principle using rule-based evidence statements. If a scan is flagged, the dashboard states which source contacted many ports or hosts. If a DDoS-like event is flagged, the dashboard states the destination, packet count, source diversity, and related rate indicators. This makes the result easier to verify in a packet analyzer.

B. Critical Analysis

The literature provides strong evidence that automated traffic analysis is useful, but it also reveals limitations. Many ML and DL approaches are evaluated on preprocessed flow datasets, while a forensic user may begin with a raw

TABLE II. COMPARISON OF RECENT WORK RELEVANT TO THE PROJECT

Study	Year	Method / focus	Key features	Relevance to NetForensics
Sikos [1]	2020	Packet analysis for network forensics	Deep packet inspection, evidence value, packet-level investigation	Supports the decision to use packet-capture evidence rather than only summary logs.
Rosay et al. [2]	2021	Corrected IDS dataset from CIC-IDS2017	Flow extraction issues, feature correction, dataset reliability	Shows why dataset and feature quality must be discussed carefully.
Rizvi et al. [3]	2022	AI for network forensics	Expert systems, ML, DL, automated classification	Supports automation but also highlights false-positive concerns.
Alansari et al. [4]	2023	Network-crime detection and investigation model	Capture, preserve, reconstruct, analyze, document	Aligns with the project's evidence-to-report workflow.
Belalis et al. [14]	2023	Reconnaissance and port-scan evasion	Port-scan features and evasion behavior	Supports multi-indicator scan detection instead of a single threshold.
Ghani et al. [15]	2023	Small feature vector IDS	UNSW-NB15, NSL-KDD, feature reduction	Supports compact forensic feature design.
Altulaihan et al. [10]	2024	DoS detection in IoT	DT, RF, kNN, SVM, anomaly profiles	Supports traffic-based DoS/DDoS detection logic.
Gelgi et al. [8]	2024	IoT botnet DDoS review	Botnet architecture, DDoS taxonomy, ML/DL detection	Provides recent DDoS context and threat evolution.
Pekar and Jozsa [13]	2024	ML anomaly detection across dataset integrity levels	Refined CICIDS2017 flows and RF evaluation	Justifies avoiding unsupported benchmark claims.
Musthafa et al. [12]	2024	Balanced IoT IDS	Class balancing, feature selection, ensembles	Motivates transparent preprocessing and class-balance awareness.
Rahman et al. [7]	2025	IoT IDS survey	Data extraction, metrics, ML and DL trends	Supports the need for usable IDS design and evaluation criteria.
Anis et al. [19]	2025	Deep-learning NIDS survey	Dataset analysis, supervised/unsupervised methods	Provides recent background for future ML extensions.

PCAP/PCAPNG file. Some studies report classification performance but do not explain how a user should move from a prediction to a forensic report. Other studies use benchmark datasets that may contain labeling problems, class imbalance, leakage-prone identifiers, or unrealistic traffic mixtures [2], [13], [29], [30]. As a result, high accuracy alone is not enough for an academic forensic tool.

A second limitation is usability. Existing packet tools are powerful but expert-oriented. Existing IDS models can be accurate but are often not designed as complete web-based investigation workflows. A student or novice investigator needs step-by-step outputs: upload, analysis progress, metric cards, graphs, alert explanations, and exportable reports. This practical workflow is less visible in much of the model-centered literature.

C. Practical Comparison with Existing Tools

There are many mature tools available that support packet inspection, intrusion detection, and network analysis, such as Wireshark, Zeek, Suricata, and NetworkMiner. NetForensics does not aim to replace any of those tools or claim higher efficacy in terms of detection rate. What NetForensics aims at is integrating PCAP uploading, rule-based forensic triaging, explanation of the results, and multiple report formats generation into a single streamlined procedure. Table III summarizes a real-world comparison of NetForensics with some popular tools for network analysis and forensics.

The objective of NetForensics is not to replace mature packet-analysis or IDS platforms. Instead, it fills a narrower usability gap by combining PCAP upload, feature extraction, rule-based forensic triage, dashboard explanation, and multi-format report generation in one workflow. This makes it suitable for academic laboratories, early-stage responders, and structured forensic demonstrations.

TABLE III. PRACTICAL COMPARISON WITH EXISTING NETWORK ANALYSIS AND FORENSIC TOOLS.

Feature	Wireshark	Zeek	Suricata	NetworkMiner	NetForensics
PCAP/PCAPNG input	Yes	Yes	Yes	Yes	Yes
Packet-level inspection	Strong	Medium	Medium	Medium	Medium
Automated alerting	Manual/filter-based	Log/script-based	Strong IDS rules	Limited	Rule-based forensic alerts
Beginner-friendly workflow	Low	Medium	Medium	Medium	High
Web dashboard	No native	No native	External tools needed	No	Yes
PCAP-to-report workflow	No	Partial	Partial	Partial	Yes
JSON export	Partial/external	Yes	Yes	Limited	Yes
CSV export	Partial	Yes	Yes	Partial	Yes
PDF forensic report	No	No	No	Limited	Yes
Explainable evidence statements	Manual	Log-based	Rule-based	Artifact-based	Alert evidence + recommendations
Intended use	Expert packet analysis	Network security monitoring	IDS/IPS	Artifact extraction	Academic and first-stage DFIR triage

D. Research Gap and Novelty

The research gap is the absence of a simple, explainable, PCAP/PCAPNG-to-report workflow that combines web upload, automated feature extraction, rule-based forensic detection, correlation, dashboard interpretation, and report export for academic and first-stage triage use. The novelty of this project is the implementation integration: it connects raw packet evidence to readable dashboard evidence and exportable investigation artifacts. This allows one to link the generated alerts back to real-world packet activity and observable behaviors.

E. Practical Novelty and Scientific Contribution

In contrast to other forensic tools available on the market which mostly offer packet inspection, traffic logging or separate detection approaches, NetForensics proposes a complete investigation workflow from raw packets to explainable reports. This innovation is not about offering a new detection technique but about integrating evidence acquisition, features extraction, behavioral analysis based on rules, alert correlation, dashboard presentation, traceability and reporting in one reproducible workflow.

The proposed architecture was developed in order to bridge the gap between low-level packet evidence and investigation-oriented forensic analysis. Unlike existing solutions where

TABLE IV. SCIENTIFIC CONTRIBUTION COMPARED WITH EXISTING APPROACHES.

Contribution Aspect	Existing Approaches	NetForensics
Packet analysis	Individual inspection tools	Integrated investigation workflow
Evidence correlation	Manual	Automatic correlation engine
Investigation reporting	External or manual	Automatic JSON, CSV, and PDF reports
Explainability	Limited	Explicit evidence-based explanations
Workflow integration	Multiple disconnected tools	End-to-end forensic pipeline
Educational usability	Low	High

analysts need to correlate output results from different tools, NetForensics preserves consistent evidence during the entire investigation process and ensures transparent detection logic and explainable results applicable in academic labs, digital forensics training, and first stage incident response.

Integration offered in the proposed solution contributes to simplification of the investigation process, workflow consistency, forensic reporting reproducibility and easy system extension due to its modularity. Table IV details the scientific contribution compared with existing approaches.

VI. SYSTEM ARCHITECTURE AND IMPLEMENTATION

The architectural process of six phases is described in Fig. 2. The design of the project is deliberately made modular. Each of the phases in the architecture is implemented as a separate Python module or as a Python template in the project directory, not a code snippet in the text.

A. Phase 1: PCAP/PCAPNG Parsing

The upload method takes a file that has the extension `.pcap` or `.pcapng`. The program creates a session ID and uploads the file to the `uploads/` folder. In turn, the parser loads packets using Scapy. The parser parses each IP packet by extracting its timestamp, source IP address, destination IP address, protocol used, packet size, source port, destination port, and TCP flags if present. The packets which are not IP packets may be skipped or marked as unsupported packets based on the settings of the parser. The parser also builds flow-like groups through communication tuples.

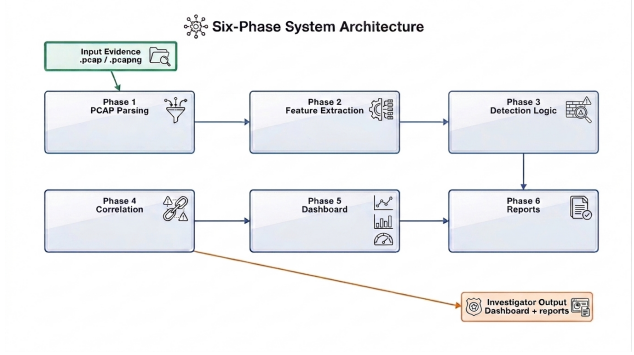

Fig. 2. NetForensics six phase architecture. The process starts with packet upload, followed by feature extraction, detection of alerts, correlation, visualization, and finally report exporting.

B. Phase 2: Feature Extraction

Feature extraction converts the packet record information into smaller pieces of information called forensic features. Total number of packets, total number of flows, time of recording, average number of packets per second, average number of bytes per second, protocol information, top source IP addresses, top destination IP addresses, top destination ports, abnormal destination ports, and packets per second buckets are some of the forensic features generated by this module. Source profiles and destination profiles including packet count, byte count, unique peers, diversity of destination ports, SYN count, ACK count, first seen time, and last seen time have also been included among the generated forensic features.

C. Phase 3: Detection Logic

The detection layer uses transparent rules rather than hidden model decisions. Fig. 3 shows the mapping between detector families and evidence.

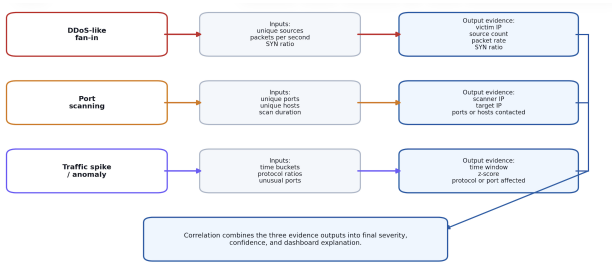

Fig. 3. Detection logic and evidence mapping. Each detector converts packet-derived behavior into an alert that can be verified by an investigator.

The DDoS detector checks for many sources targeting one destination, high packet rates, and SYN-heavy behavior. The port-scan detector checks for vertical scans, horizontal scans, and block scans by measuring destination-port diversity, destination-host diversity, packet count, and scan duration. The traffic-spike detector evaluates packet-rate windows using z-score logic and also flags protocol dominance and unusual destination-port activity. Each detector returns a structured result containing whether it was triggered, alert count, severity, confidence, evidence, and recommendations.

The suggested rule-based detectors use configurable threshold values for separating suspicious activities from typical network operations. However, instead of choosing the threshold values randomly, the authors calibrated those values empirically by implementing various forensic scenarios that included both regular and malicious traffic behavior. While creating the detectors, the threshold values have been iteratively refined and tested until the consistency of detecting the required behavior was ensured without any unnecessary alerts on normal traffic flow.

Each detector was implemented many times under various traffic loads and modes of communication. The chosen threshold values are the minimal values that ensure the required behavior is detected without any false alarms in all tested scenarios. Table V presents the empirical threshold selection for rule-based detectors.

TABLE V. EMPIRICAL THRESHOLD SELECTION FOR RULE-BASED DETECTORS.

Detector	Primary Threshold	Selection Rationale
DDoS-like Fan-in	Minimum number of unique source hosts	Selected to distinguish concentrated fan-in traffic from normal client communication.
SYN Flood	High SYN packet ratio	Chosen to identify abnormal connection establishment attempts while reducing false alerts.
Port Scan	Number of unique destination ports	Calibrated to separate legitimate service access from systematic scanning behavior.
Traffic Spike	Packet-rate z-score	Selected to detect statistically significant traffic bursts above normal variation.
Unusual Ports	Destination-port frequency	Chosen to identify rarely used services that may indicate suspicious communication.

To assess the strength of the chosen thresholds, the detection algorithms were run on the forensics scenarios involving varying volumes of traffic, varying lengths of time for communication, and varying combinations of regular and suspicious behavior. While the present testing has been carried out based on artificial traffic and not real-world operational data, the thresholds have proven to be stable throughout the testing process and have managed to recognize the desired behavioral patterns.

D. Phase 4: Correlation

Individual alerts are useful, but a forensic dashboard should provide a final interpretation. The correlation module combines alert families, severity values, and evidence density into a final classification, severity, confidence score, and narrative explanation. For example, a single unusual port may produce a low-risk anomaly, while simultaneous fan-in traffic and a vertical scan may produce a high-severity suspicious classification. The correlation output appears at the top of the dashboard and in the PDF report.

E. Phase 5: Dashboard Visualization

The visualization layer prepares chart-ready data. The dashboard includes a navigation bar with sections for scenario, metrics, charts, explanation, alerts, and reports. This was added to make the full scenario visible to the user: the page does not only show graphs; it explains how the user moved from PCAP upload to dashboard and report outputs. The metrics are also accompanied by definitions on the dashboard so that the user knows what each of the metrics is all about.

F. Phase 6: Report Generation

The file export formats for the reports include JSON, CSV, and PDF. JSON format retains all aspects of the case structure object, which includes metadata, features, detection results, correlation, thresholds, and report paths. CSV provides an alert-level table that can be opened in spreadsheet software. PDF provides an investigator-readable report with executive summary, traffic evidence, alert details, evaluation context, thresholds, and recommendations. Table VI presents the clarification of implementation modules.

G. Evidence Integrity and Report Traceability

In order to increase the forensic reliability of the workflow process, NetForensics stores metadata related to integrity and traceability for each PCAP/PCAPNG upload. After each upload, NetForensics creates a session ID and computes an SHA-256 hash from the original file before performing the analysis. The original file serves as the source evidence, while the packet record extraction and feature calculation happen separately from it within the analysis pipeline. These values can be stored along with other information about the upload process inside the JSON report, CSV alert table, and PDF report to ensure that everything produced in NetForensics came from the same evidence. Additional information includes analysis timestamp, filename, detection threshold values, and file locations where the output was saved. Table VII presents the integrity and traceability elements implemented in NetForensics to ensure consistency between uploaded evidence and generated forensic reports.

VII. DASHBOARD RESULTS VIEW

Fig. 4 explains the dashboard result view. The correlation module produces the top classification. The metric cards show packet-derived and alert-derived values. *Total packets* is the number of parsed packets. *Total flows* is the number of grouped communication records. *Unique sources* is the count of distinct source IPs. *Threat severity* is the final label assigned by correlation. *Detections* is the number of alert records created by the detector modules. *Confidence* is a normalized score showing how strongly the evidence supports the final classification.

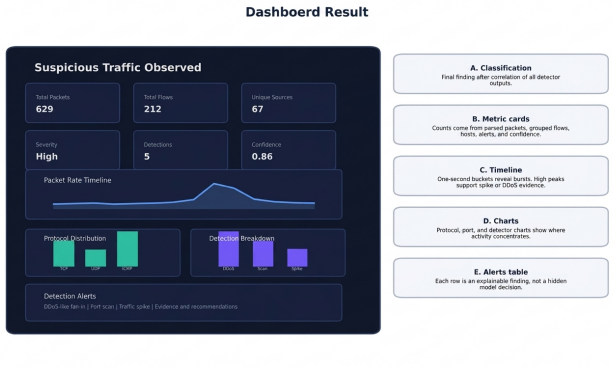

Fig. 4. Dashboard results view. The numbers are not manually entered; they are computed from parsing, feature extraction, detection, and correlation outputs.

VIII. COMPLETE SCENARIO AND EXAMPLE EVIDENCE

The scenario implemented in the package begins with a sample PCAP generator. The generated capture contains normal web and DNS traffic, a vertical scan from one source to many destination ports, DDoS-like SYN-heavy fan-in traffic from many sources to one destination, and unusual UDP destination-port activity. This is not used to claim benchmark accuracy. This is used to show how all the components in the process work together.

Fig. 5 illustrates the whole process, starting with uploading the file and ending with producing reports. The process starts with uploading the capture using the web page, processing takes place on the server using six steps, then the evidence gets presented using the dashboard, and finally the user downloads reports in JSON, CSV, and PDF formats.

The output of JSON is helpful when the subsequent tool/script requires the entire object of results. The output of CSV comes in handy when the investigator wishes to filter the alert records based on severity, source IP, destination IP,

TABLE VI. CLARIFICATION OF IMPLEMENTATION MODULES

Module/file	Where it is written	Clarification
`app.py`	Project root	Flask code containing the route definitions for the uploads, the load process, the analysis, dashboard display, API results, and downloading the report.
`pcap_parser.py`	`modules/parsing/`	Parser code source. It parses the PCAP/PCAPNG file format and converts the packets into structured data in terms of packet and flow data records.
`feature_extractor.py`	`modules/features/`	Features source code. It calculates the features used by detectors such as packet, host, port, protocol, and timeline stats.
`ddos_detector.py`	`modules/detection/`	Fan-in, high-rate, and SYN-heaviness detector code.
`portscan_detector.py`	`modules/detection/`	Vertical, horizontal, and block scanning detectors code.
`traffic_spike` detector	`modules/detection/`	Z-score peaks, protocol dominance, and weird destination ports detectors code.
`correlator.py`	`modules/correlation/`	Aggregates all the detectors' findings to classify the attack, severity, confidence level, and explain the findings.
`dashboard.html`	`templates/`	The browser template that shows metric cards, charts, guide scenarios, explanations, alerts, and links to reports.
`report_generator` v2	`modules/`	Reports the code source. It outputs in JSON, CSV, and PDF the findings of the investigation.

TABLE VII. FORENSIC INTEGRITY AND TRACEABILITY ELEMENTS

Forensic Integrity Element	Required Implementation
Unique session ID	Generated at upload and attached to all outputs.
Original evidence hash	SHA-256 calculated before analysis.
Evidence preservation	Original PCAP/PCAPNG file not modified.
Report traceability	JSON, CSV, and PDF reports include the same session ID.
Output consistency	Alerts in CSV must match alerts in JSON/PDF reports.
Threshold recording	Detector thresholds stored in report metadata.
Time recording	Upload time, analysis start time, and report generation time recorded.
Tool reproducibility	Python, Scapy, and Flask versions documented.
Report hash	Optional SHA-256 hash for the generated PDF report.

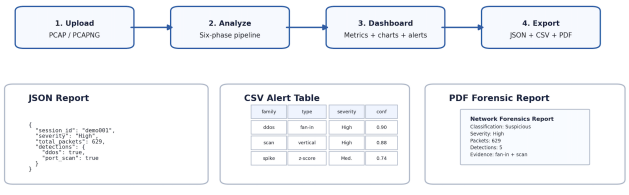

Fig. 5. Complete example scenario. The panels show the workflow from upload to analysis and the three report outputs: JSON for structured evidence, CSV for alert rows, and PDF for readable investigation reporting.

or detector family. The PDF output is useful for submission, review, or first-stage incident documentation. Together,

TABLE VIII. EVALUATION CRITERIA USED FOR THE PROTOTYPE

Criterion	How it is checked
Parsing completeness	The uploaded PCAP/PCAPNG must be read, packet records must be created, and flow-like groupings must be formed.
Feature correctness	Packet count, flow count, source count, destination count, protocol distribution, destination ports, and packet-rate timeline must appear consistently in dashboard and reports.
Detector correctness	Known synthetic behaviors must trigger the expected detector families: fan-in traffic for DDoS-like detection, many ports for vertical scan detection, and high packet buckets for spike detection.
Explainability	Each alert will need an evidence statement and a recommendation such that it can be verified manually.
Dashboard interpretability	The dashboard will need to display classification, severity, confidence, metric cards, charts, explanations, and alerts without having any overlapping text.
Report generation	There will need to be JSON, CSV, and PDF reports available for download from the dashboard.
Modularity	Each step should be kept separate so that in the future, ML models, Zeek logs, and stream-based approaches can be added.

these formats cover machine-readable, spreadsheet-readable, and human-readable reporting needs.

IX. EVALUATION APPROACH

The evaluation criteria are functional forensic criteria. The system is evaluated by checking whether the implementation performs the intended workflow correctly on a known capture scenario. The criteria are shown in Table VIII.

X. SCALABILITY AND COMPUTATIONAL ANALYSIS

The present implementation handles uploaded PCAP/PCAPNG files serially, in such a way that every

TABLE IX. SCALABILITY CHARACTERISTICS OF THE PROPOSED WORKFLOW

Workflow Stage	Processing Behavior	Primary Computational Cost
Packet Parsing	Sequential packet processing	Reading uploaded packet captures
Feature Extraction	Feature computation from parsed packets	Traffic statistics generation
Rule-Based Detection	Rule evaluation over extracted features	Behavioral analysis
Alert Correlation	Alert aggregation	Evidence correlation
Dashboard Generation	Summary visualization	Rendering processed results
Report Generation	Export processing	JSON, CSV, and PDF generation

analysis proceeds with the full six-stage forensic process before handling the next request. This decision was deliberately made in order to facilitate management of the evidence, analysis reproduction, and deterministic report generation.

While the system was not intended to be used as a high-throughput forensic system, the modularity of its design allows for scaling in the future without any need for changing the overall forensic process itself. This is because the processes of packet parsing, feature extraction, detection, correlation, visualization, and reporting are all separate modules that could be optimized independently in the future (see Table IX).

Processing is mainly determined by packet analysis and feature extraction since these processes take place right at the level of packet capture. Other components deal with forensic feature summary and alerts generated, hence processing overhead becomes less.

A. Memory Consumption

The memory consumption of the existing architecture depends mostly on the size of the uploaded packet capture file as well as on the number of extracted forensics that are kept while analyzing the packet capture data. Because of the sequential nature of the packet processing, the more data is parsed, the higher memory consumption will be experienced, but not the duration of the analysis.

B. Concurrent User Considerations

At present, the implemented system processes investigations that have been uploaded independently and is primarily meant for use in educational labs and first-level forensics. Thus, the developed system is not optimized for usage by many concurrent users. However, the design of the analysis modules as well as their stateless nature combined with the way evidence is processed during a session allows for deploying the system asynchronously in the future without changing the workflow.

In summary, although the current system favors reproducibility, explainability, and modularity more than throughput, the proposed architecture creates a solid platform for scalable development in the future through parallel packet processing, chunked parsing, streaming, and cloud support.

XI. DISCUSSION

The project results show that a web-based forensic workflow can make packet-capture analysis more understandable for novice users. Instead of presenting raw packet rows, it analyzes traffic through timeline of packet rate, protocol ratio, source, destination ports, and alert explanations. This is very useful in academic research since it makes it easy for users to relate traffic behavior to forensic results.

The calibration of thresholds via empirical data enhances the clarity of the framework as well. As opposed to statistics and machine learning algorithms where it is hard to figure out why an event belongs to one or another category, all detectors in NetForensics raise alerts based on explicitly formulated behavioral parameters.

The dashboard metrics also provide analytical meaning beyond simple visualization. Packet count and packet-rate timelines help investigators identify abnormal traffic bursts and temporal anomalies. Flow count and unique-source metrics help estimate communication diversity and possible fan-in behavior associated with DDoS-like activity. Protocol distribution assists in identifying dominant protocols or unusual protocol usage patterns, while destination-port frequency analysis helps reveal scanning behavior and suspicious service targeting. Severity and confidence values quantify the extent to which the collected evidence matches the implemented forensic rules. Together, these metrics help users move from raw packet observations to interpretable forensic indicators that support preliminary incident assessment.

This study has also shown the benefit of using modular design. For instance, the parser, feature extractor, detectors, correlation, dashboard and reporting modules have been designed separately. What this does is allow new detectors to be introduced without any changes to the upload or reporting modules. New machine learning module can be included later on after feature extraction without changing the dashboard and reporting modules.

Scientific contribution of NetForensics must hence be considered as a systems integration contribution instead of a new approach towards packet detection algorithm. In doing so, the architecture shows how forensic technologies that have been developed independently can be integrated in order to build an investigative process. The resulting system integration helps in making forensic investigation easier, without compromising on transparency and extensibility, thus adding value for the teaching of digital forensics and preliminary investigation.

Another important result is the reporting layer. JSON, CSV, and PDF exports give the investigator different ways to use the same evidence. JSON supports reproducibility and integration. CSV supports tabular review and filtering. PDF supports human-readable reporting. This aligns with incident response needs because analysts often need both machine-readable records and concise summaries for supervisors or coursework submissions.

A. Architectural Modularity and Extensibility

Modular design of NetForensics is deliberate and is used to divide the process of forensic analysis into separate functions. Every module has a particular function to perform and interacts with the next modules by means of structured data and not by implementing a certain functionality directly. This way, any change in a single module will not affect the other modules, as long as interfaces stay the same (see Table X).

TABLE X. MODULE INDEPENDENCE WITHIN THE NETFORENSICS ARCHITECTURE.

Module	Primary Responsibility	Dependency on Other Modules
Packet Parser	Reads PCAP/PCAPNG files	Independent
Feature Extraction	Computes forensic features	Depends only on parsed packets
Rule-Based Detection	Evaluates suspicious behavior	Uses extracted features only
Correlation Engine	Aggregates detector outputs	Uses alert results only
Dashboard	Visualizes investigation results	Uses summarized analysis output
Report Generator	Exports JSON, CSV, and PDF reports	Uses finalized forensic results

TABLE XI. TYPICAL EXTENSION SCENARIOS

Extension	Modified Module	Impact on Remaining Workflow
New detector	Detection module	No modification required
Additional report format	Report generator	No modification required
New dashboard visualization	Dashboard module	No modification required
Additional forensic feature	Feature extraction	Detection modules can optionally use it

It is much easier to evolve the system when the responsibilities are divided. The addition of any new functionality such as a new type of detection like detection of DNS tunneling or beaconing would only entail writing another module for detection but leaving the rest of the system untouched. Similarly, enhancement of the visual or reporting capabilities will be done separately. Table XI presents the typical extension scenarios.

While the present prototype does not support any plug-in architecture explicitly, the chosen architecture does follow a plug-in architecture approach through isolation of detector implementations from other parts of the process. This makes the maintenance task easy, and also supports future expansion of the system with additional features.

XII. LIMITATIONS

Even though NetForensics creates a full forensic procedure that can be explained easily, several weaknesses should be pointed out regarding the current version of the product.

First, the current implementation relies primarily on rule-based detection logic rather than adaptive machine-learning or behavioral models. While rule-based approaches improve explainability and reproducibility, they may not generalize well to highly dynamic or previously unseen attack patterns. Another limitation is that the current evaluation focused on functional validation using synthetic forensic scenarios rather than quantitative evaluation using independently labeled test data.

Secondly, scalability limitations may arise when processing very large packet captures because the current implementation performs file-based sequential analysis without distributed processing, database indexing, or stream-based optimization. Large-scale traffic analysis may therefore require substantial processing time and memory resources.

Finally, the system does not perform attacker attribution or advanced threat-intelligence correlation. The generated alerts indicate suspicious communication behavior and forensic indicators, but they do not prove attacker identity, intent, or organizational attribution. Therefore, NetForensics should be considered a first-stage forensic triage and educational investigation platform rather than a replacement for expert-led forensic analysis or enterprise-grade intrusion-detection systems.

TABLE XII. POTENTIAL OPERATIONAL LIMITATIONS OF THE PROPOSED DETECTION FRAMEWORK.

Network Condition	Potential Impact	Possible Future Enhancement
Encrypted traffic	Application-layer information becomes unavailable	Integrate flow-based behavioral analysis
Fragmented packets	Incomplete packet reconstruction may affect detection	Support fragment reassembly
Asymmetric routing	Partial communication flows may be observed	Multi-point traffic collection
Packet loss	Missing packets may reduce evidence completeness	Capture quality validation
NAT environments	Multiple hosts may appear under one public IP address	Session-aware host correlation
IPv6 networks	Current implementation focuses primarily on IPv4 traffic	Extend feature extraction to IPv6 headers

A. Operational Robustness Under Challenging Network Conditions

The rule-based detector algorithms were tested under typical forensic synthetic scenarios that emulate the desired behavior of the network. Nevertheless, in its current state, the implementation does not consider a number of possible network environments that may negatively impact the reliability of the detection process. They include encrypted traffic, fragmented IP packets, asymmetric routing, packet drop while capturing, Network Address Translation (NAT), and IPv6 traffic. As these environments may restrict the availability of the packet-level information, some of the behavioral markers used in the detector algorithms may become unobservable (see Table XII).

While these constraints do not impact the validity of the developed forensic methodology in the analyzed scenarios, they should be taken into account while implementing the described methodology in practice. Meeting these requirements is a key direction for future research and will lead to even better robustness of the described framework in various networks.

XIII. CONCLUSION AND FUTURE WORK

NetForensics implements a complete web-based PCAP/PCAPNG forensic workflow. It accepts packet-capture files, extracts packet and flow-style features, detects DDoS-like traffic, port scanning, traffic spikes, protocol dominance, and unusual ports, correlates the findings, displays a dashboard, and exports JSON, CSV, and PDF reports. One of the major findings of this study is that a beginner-friendly web interface can help in understanding packet evidence without obscuring the reasoning behind the generated alerts.

The evaluation results demonstrated that the implemented workflow was able to process uploaded PCAP/PCAPNG files successfully, generate reproducible forensic outputs, and detect the intended synthetic behaviors used in the experimental scenario. The dashboard, alert system, and exported JSON, CSV, and PDF reports remained consistent with the parsed evidence and detector outputs throughout the workflow.

The contribution of this study includes implementation modularity with six phases, explanation of the detection mechanism, dashboard visualization, evidence traceability, and exportable forensic reporting. The practical usefulness of this work lies in its application in laboratories, educational environments, and the initial stage of incident response, where users need an accessible explanation to begin deeper investigation and forensic analysis.

In addition, the modular design of the suggested architecture is a basis for sustainable future development in terms of adding more detectors, features and capabilities without breaking the compatibility with current forensic process.

A. Future Directions

Possible directions for future work will be broadening the scope to IoT networks, enterprise traffic, cloud traffic, and industrial control traffic. It will be useful to develop scalability through the use of streaming parsers, database indexing, and chunked processing. Real-time implementation may be achieved with live capture ingestions, Zeek logs integration, alert queuing, and continuous dashboards. Examine the performance of the proposed model in harsh operational scenarios that include encrypted network traffic, fragmentation of packets, loss of packets, asymmetric routing, use of NAT, and IPv6 networks. The extension of the algorithm to handle such environments is expected to increase its reliability and applicability in forensic analyses in the enterprise domain. Machine learning approaches like Random Forest, XGBoost, and deep learning models can be used, but care must be taken to implement dataset quality assurance, class imbalance techniques, cross-dataset validation, and XAI techniques recently proposed in the literature [17]–[20].

ACKNOWLEDGMENT

The authors acknowledge the support of the Deanship of Scientific Research, Vice Presidency for Graduate Studies and Scientific Research, King Faisal University, Saudi Arabia [Grant No. KFU263877].

REFERENCES

[1] L. F. Sikos, "Packet analysis for network forensics: A comprehensive survey," *Forensic Science International: Digital Investigation*, vol. 32, Art. no. 200892, 2020.

[2] A. Rosay, F. Guenane, K. Lounis, D. Idoughi, and J. Ben-Othman, "From CIC-IDS2017 to LYCOS-IDS2017: A corrected dataset for better performance," in *Proc. Int. Conf. Availability, Reliability and Security*, 2021.

[3] S. Rizvi, A. Kurtz, J. Pfeffer, and M. Rizvi, "Application of artificial intelligence to network forensics: A survey," *Forensic Science International: Digital Investigation*, vol. 41, Art. no. 301401, 2022.

[4] I. S. Alansari, A. Alghamdi, and R. Alshammari, "A detection and investigation model for the capture and analysis of network crimes," *Engineering, Technology & Applied Science Research*, vol. 13, no. 6, pp. 12238–12244, 2023.

[5] A. A. Ahmed *et al.*, "IoT forensics: Current perspectives and future directions," *Sensors*, vol. 24, no. 16, Art. no. 5210, 2024.

[6] H. Mahmood *et al.*, "Comparative study of IoT forensic frameworks," *Forensic Science International: Digital Investigation*, vol. 50, Art. no. 301788, 2024.

[7] M. M. Rahman *et al.*, "A survey on intrusion detection system in IoT networks," *Cyber Security and Applications*, vol. 3, Art. no. 100084, 2025.

[8] M. Gelgi, Y. Guan, and A. A. Ghorbani, "Systematic literature review of IoT botnet DDoS attacks and detection techniques," *Sensors*, vol. 24, no. 11, Art. no. 3571, 2024.

[9] W. Su *et al.*, "A comprehensive survey on DDoS detection, mitigation and defense strategies in software-defined networks," *IEEE Access*, vol. 12, pp. 88311–88345, 2024.

[10] E. Altulaihan, M. A. Almaiah, and A. Aljughaiman, "Anomaly detection IDS for detecting DoS attacks in IoT networks based on machine learning algorithms," *Sensors*, vol. 24, no. 2, Art. no. 713, 2024.

[11] S. More *et al.*, "Enhanced intrusion detection systems performance with machine learning and feature selection," *Algorithms*, vol. 17, no. 2, Art. no. 64, 2024.

[12] M. B. Musthafa *et al.*, "Optimizing IoT intrusion detection using balanced class distribution, feature selection, and ensemble machine learning techniques," *Sensors*, vol. 24, no. 13, Art. no. 4293, 2024.

[13] A. Pekar and T. Jozsa, "Evaluating ML-based anomaly detection across datasets of different integrity," *Computer Networks*, vol. 249, Art. no. 110617, 2024.

[14] I. Belalis *et al.*, "Evading detection during network reconnaissance," in *Proc. International Conference on Security and Cryptography*, pp. 337–344, 2023.

[15] H. Ghani *et al.*, "A deep learning approach for network intrusion detection using a small features vector," *Journal of Cybersecurity and Privacy*, vol. 3, no. 3, pp. 451–463, 2023.

[16] Y. N. Rao *et al.*, "An imbalanced generative adversarial network-based approach for network intrusion detection," *Sensors*, vol. 23, no. 1, Art. no. 550, 2023.

[17] D. Gaspar *et al.*, "Explainable AI for intrusion detection systems: LIME and SHAP applicability on multi-layer perceptron," *IEEE Access*, vol. 12, pp. 30164–30175, 2024.

[18] A. L. Samed *et al.*, "Explainable artificial intelligence models in intrusion detection systems: A review and future directions," *Engineering Applications of Artificial Intelligence*, vol. 144, Art. no. 110067, 2025.

[19] F. M. Anis, M. Alabdullatif, S. Aljbli, and M. Hammoudeh, "A survey on the applications of deep learning in network intrusion detection systems to enhance network security," *IEEE Access*, vol. 13, pp. 185357–185373, 2025.

[20] A. Hozouri, A. Mirzaei, and M. Effatparvar, "A comprehensive survey on intrusion detection systems with advances in machine learning, deep learning and emerging cybersecurity challenges," *Discover Artificial Intelligence*, vol. 5, Art. no. 314, 2025.

[21] M. B. Bankó *et al.*, "Advancements in machine learning-based intrusion detection for distributed denial-of-service attacks in Internet of Things environments," *Algorithms*, vol. 18, no. 4, Art. no. 209, 2025.

[22] N. Sarwar *et al.*, "Securing IoT networks: A machine learning approach for anomaly detection in IoT systems," *Scientific Reports*, vol. 15, Art. no. 33447, 2025.

[23] R. Almuhanna *et al.*, "A deep learning/machine learning approach for anomaly-based network intrusion detection," *Scientific Reports*, vol. 15, Art. no. 32018, 2025.

[24] M. A. Faizin *et al.*, "Optimizing feature selection method in intrusion detection systems using XGBoost on UNSW-NB15," *International Journal of Intelligent Engineering and Systems*, vol. 17, no. 3, pp. 18–31, 2024.

[25] E. M. Maseno *et al.*, "Performance evaluation of intrusion detection systems on the UNSW-NB15 dataset using feature selection," in *Proc. International Conference on Intelligent Computing and Networking*, pp. 119–126, 2024.

[26] W. H. M. Kurdi *et al.*, "Efficient two-stage intrusion detection system based on feature selection and machine learning," *International Journal of Intelligent Engineering and Systems*, vol. 18, no. 2, pp. 304–318, 2025.

[27] M. Al-Ajlan *et al.*, "A review of generative adversarial networks for intrusion detection systems," *Computer Modeling in Engineering & Sciences*, vol. 141, no. 2, pp. 1045–1072, 2024.

[28] Y. Yang *et al.*, "A conditional generative adversarial network approach to address data imbalance in network intrusion detection," *Scientific Reports*, vol. 15, Art. no. 7620, 2025.

[29] D. Pinto *et al.*, "A review on intrusion detection datasets: Tools, processes and perspectives," *Computer Networks*, vol. 258, Art. no. 111005, 2025.

[30] M. Cantone *et al.*, "Machine learning in network intrusion detection: A cross-dataset generalization study," *IEEE Access*, vol. 12, pp. 117934–117956, 2024.