National College of Ireland

Hybrid Browser-Based Framework for Mobile Threat Detection and Forensic Analysis

MSc Research Project
Programme Name

ROSHAN MUTTATH FRANCIS
Student ID: x23299401

School of Computing
National College of Ireland

Supervisor: Michael Prior

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	ROSHAN MUTTATH FRANCIS
Student ID:	X23299401
Programme:	MSc Cybersecurity **Year:** 2024-2025
Module:	MSc Research Project
Supervisor:	Michael Prior
Submission Due Date:	12/08/2025
Project Title:	Hybrid Browser-Based Framework for Mobile Threat Detection and Forensic Analysis
Word Count:	7158 **Page Count: 23**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ROSHAN MUTTATH FRANCIS

Date: 12/08/2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Hybrid Browser-Based Framework for Mobile Threat Detection and Forensic Analysis

ROSHAN MUTTATH FRANCIS

Abstract

Mobile forensic investigations require secure, portable, and intelligent tools to address evolving threats. This research offers a Hybrid Threat Detection Framework that combines various methods of detection in order to provide full forensic reach. The architecture includes YARA-based signature scanning for accurate recognition of well-known malware families, regex-based Indicator of Compromise (IoC) extraction for quick identification of suspicious patterns like URLs, IP addresses, and email identifiers, and an RNN-driven WhatsApp chat toxicity analysis module for the detection of abusive or harmful communications. The system facilitates the analysis of a wide variety of digital artifacts, such as APKs, SQLite databases, log files, and chat exports, in their entirety offline via WebAssembly and JavaScript to maintain privacy protection, platform neutrality, and cross-device compatibility. ALEAPP parsing also adds to the capabilities of the framework by reconstructing timelines for devices, interpreting app history usage, and detecting removed or suspicious system events. All detection results are aggregated into formatted, legally defensible reports in PDF, CSV, and JSON formats, facilitating easy integration into investigative processes. Performance testing shows that the RNN toxicity classification component attains 91% accuracy with minimal latency for real-time processing. By integrating technical malware detection with behavioral threat analysis, the presented framework arms investigators with a field-capable, privacy-friendly of tools that boosts the efficacy and consistency of modern mobile forensic investigations.

***Keywords*- Mobile forensics, YARA rules, regex detection, ALEAPP, progressive web app.**

1 Introduction

The widespread use of mobile phones in everyday life has transformed them into a hub of communication, personal information, financial dealings, and organizational functions(Cornelissen 2023). As a result, mobile platforms have become ideal targets for all kinds of cyber-attacks such as malware, ransomware, remote access trojans (RATs), downloader scripts, and credential stealing. Legacy mobile forensic solutions typically incorporate signature-based scanning and centralized infrastructure, hence constraining their capabilities in real-time, remote, or offline scenarios(Paul et al. 2024). They may also fail to be flexible enough to identify zero-day attacks or obfuscated malicious behavior. Many forensic tools are also subject to installation requirements, platform dependency, or compromising data privacy through cloud dependency, which renders them less beneficial for field investigations, border security screening, or sensitive incident response activities(Henriques et al. 2024). As the threat environment evolves, there is an increasing requirement for an intelligent and portable forensic solution that is accurate, secure, and scalable.

This research introduces a novel hybrid mobile forensic framework that can run exclusively inside a web browser, without needing to be installed or requiring server access. The framework can perform safe, offline examination of various mobile forensic files, such as APKs, SQLite databases, logs, and WhatsApp chat exports. Detection is driven by a series of engines integrated together: YARA rules for malware signature scanning, regex pattern matching for IoC extraction, and an RNN-based model for WhatsApp toxicity detection with 91% accuracy. ALEAPP parsing adds additional richness to investigations by extracting timelines from devices, events, and suspicious activities. Everything is processed locally using WebAssembly and JavaScript for preserving privacy, cross-platform portability, and forensic evidence handling compliance. The modular design allows for concurrent engine execution, real-time aggregation of results, and legally defensible reporting, giving investigators a responsive, field-capable solution for confronting both technical malware attacks and behavioral threats in zero-trust forensic contexts.

1.1 Research Gap

Mobile forensics is an important field in cybersecurity and digital forensics as a result of the greater adoption of smartphones for business and personal purposes. Even with the developments in mobile forensic solutions, there are still some key shortcomings(Patel and Mann 2025). The majority of current solutions are platform-specific, are too heavy to install, or need internet connectivity and central infrastructure, rendering them unsuitable for applications in field investigations or distant locations. In addition, these products tend to concentrate mainly on static signature-based detection, which hampers them from detecting zero-day malware, fileless malware, and polymorphic malware that bypass conventional detection methods. One other significant limitation is the absence of integration between various detection techniques—e.g., static scanning, behavioral analysis, and smart pattern recognition—within one, light-weight forensic solution(Tirumalaη, Nepalα, and Rayη 2022). Most solutions lack sophisticated approaches like YARA rule matching(Naik et al. 2021), regular expressions-based IoC detection, or machine learning-based anomaly scoring.

This study bridges existing gaps by developing a Hybrid framework that integrates static detection, regex-based pattern analysis, RNN-driven chat toxicity classification, and ALEAPP parsing into a single offline-capable forensic tool. The framework ensures security, portability, and 91% detection accuracy, supporting lawful, practical, and field-ready mobile forensic investigations.

1.2 Research Questions

Aim: To design an integrated mobile forensic system utilizing YARA, regex, RNN, and ALEAPP to detect threats, analyze chats, and generate comprehensive legal reports.
RQ1: How can a forensic platform enhance security, accessibility, and integrity in mobile forensic investigations across multiple device types?
RQ2: How effective are YARA-based detection, regex IoC extraction, and RNN-powered WhatsApp toxicity analysis in identifying diverse malware, phishing links, and toxic communications?
RQ3: How can ALEAPP parsing of device logs and timelines improve detection of suspicious actions, deleted events, and evidence correlation for forensic reporting?

1.3 Research Objective:

- To develop an offline-first ensuring secure forensic file upload, cross-platform accessibility, and maintaining evidence integrity during mobile forensic analysis.
- To implement YARA detection, regex IoC extraction, and RNN-based WhatsApp toxicity classification for comprehensive identification of malware, phishing attempts, and harmful communication patterns.
- To integrate ALEAPP parsing for extracting Android device logs, timelines, and suspicious activity, providing structured forensic timelines supporting detailed investigations and legal admissibility.

1.4 Research Methodology Overview

The research methodology for the current study is systematic and multi-stage to create and test a hybrid mobile forensic framework that can identify malware, extract Indicators of Compromise (IoCs), examine chat toxicity, and reconstruct device activity timelines. It is initiated with the creation and development of a Progressive Web App (PWA) that allows offline-first operation for safe, cross-platform forensic examination. Core modules incorporate YARA-based signature scanning for malware detection, regex pattern matching for IoC extraction, an RNN-based classifier for WhatsApp chat toxicity analysis, and ALEAPP for parsing Android device artifacts. The data acquisition process includes processing various file types (APK, SQLite, PDF, TXT, ZIP) and, for device-level evidence, leveraging Android Debug Bridge (ADB) to pull application data, logs, and system artifacts from physical devices. The testing phase puts detection precision, inference speed, and artifact extraction quality to test using real malware samples, chat datasets, and Android device logs. The outputs are all compiled into neatly formatted reports in PDF, CSV, and JSON formats to ensure legal admissibility. This blended approach guarantees the system suggested is secure, portable, scalable, and efficient for practical mobile forensic investigations.

1.5 Structure of the Report

This paper is organized into seven chapters based on the research methodology flow for mobile forensic threat detection. Chapter 1 presents the background, purpose, objectives, and problem statement. Chapter 2 presents literature review of existing mobile forensic tools to identify gaps in static detection, pattern matching, and machine learning analysis. Chapter 3 explains methodology, such as preprocessing, data acquisition, and YARA rule integration, regex patterns, heuristics, and RNN models. Chapter 4 is about system design and architecture. Chapter 5 delineates the implementation process, tools, and development of the detection engine. Chapter 6 is about evaluation through graphs, matrices, and discussions. Chapter 7 presents conclusions, limitations, and future work. This flow guarantees systematic development, testing, and validation of the framework proposed.

2. Related Work

2.1 Mobile Forensics & Threat Analysis

Mobile phones are now storage systems of enormous and diverse types of information—ranging from phone numbers, multimedia files, communication records, to geolocation and application-specific data. This only increases their relevance in forensic science, where investigators seek to retrieve, seize, and analyze data to aid legal or security objectives(“What Is Mobile Forensics? Definition, Processes, & Examples,” n.d.). Aljadali et al. (2021) discussed the

increase in mobile uptake worldwide, combined with rampant criminal abuse—from illegal communications to data exfiltration—compelled researchers to create forensic techniques that honor device security features, heterogeneous OS environments, and encryption protections.

2.2 Static vs. Dynamic Analysis: Initial Approaches

Setiawan and Sutanto (2025) discussed that early mobile forensics separated static analysis—examining file systems and app binaries without running them—from dynamic analysis, allowing observation of runtime behavior through instrumentation or network sniffing. MobSF and emulator frameworks introduced tools to triage static properties and dynamically monitor network flows and interactive runtime behavior(Shahriar et al. 2021). These approaches developed based on pattern recognition and finite-state machines to identify known malware behavior on Android (da Costa et al. 2022). Although these initial methods provided good insights in malicious patterns, they were limited by their use of known code patterns and runtime environments. Static analysis tended to miss behavioral elements, whereas dynamic approaches were hampered by emulator artifacts and anti-analysis evasion (Sutter et al. 2024).

2.3 Integrated Approaches in Mobile Forensics: Limitations, Trends, and Opportunities

Regeciova, Kolář, and Milkovič (2021) proposed YARA has become a foundational tool in forensic analysis through its ability to perform rule-based pattern matching on file strings or binaries. Recent efforts, like regular rule updates by research labs and incorporation into commercial frameworks, took YARA's use to mobile scenarios, searching backups and physical dumps for known threats. This has enhanced the detection of trojans, ransomware, and RATs, even across mutated or packed samples(Coscia et al. 2023).

Kulkarni and Cauvery (2021) discussed the complementary to malware detection, regex-based scanners are extensively utilized to scan for PII elements (emails, IPs, credit cards) and Indicators of Compromise out of logs and dumped files. However, they are liable to high false positives, are poor with obfuscation or encoding, and rarely generalize across changing formats or languages without considerable pattern tuning. Their lack of dynamics constrains detection of well-obfuscated data—like base64-encoded or split identifiers—imposing a limit on capability without sophisticated heuristics or learning layers(Mohite 2023). To fill in the voids left by signature and pattern methods, scientists have brought ML and deep learning into mobile and digital forensics. Methods vary from supervised classifiers (decision trees, SVMs, random forests) to unsupervised approaches (autoencoders, clustering), and new solutions combine deep neural networks and anomaly scoring models(Zhang 2022). Applications range from identifying unusual app activity, anomalous network traffic, or memory anomalies(Reddy et al. 2021). Ngo et al. (2023) uses graph-based and digital twin simulations for radio/network anomaly detection. Such models are successful in finding unknown or zero-day attacks with robust detection capability without obvious patterns.

With increasing context-aware and stealthy mobile threats, emerging research has proposed context-aware forensic intelligence systems that can adapt detection logic with respect to environmental influences like device type, user behavior patterns, and application history of behavior. Such systems tend to use sensor fusion, user-behavior modeling, and policy-aware detection to mark unusual activity that varies from defined baselines(Krinkin and others 2023). Such models promote enhanced relevance in real-world deployments, particularly in mobile-based workplaces(Mohammadi et al. 2025). Edge computing improves on-device analysis by minimizing latency and enabling time-sensitive investigations in offline environments. (Rancea, Anghel, and Cioara (2024) proved edge-augmented models effective in processing encrypted or shard data where mobile equipment in isolation would

crumble(Singh, Buyya, and Kim 2024). Despite the fast-paced evolution of mobile forensic tools and detection engines, current solutions have numerous limitations in responding to changing cyber threats. Conventional static analysis tools such as YARA, though good for known malware, are pattern-dependent, hence not effective against zero-day attacks, obfuscated payloads, and polymorphic malware(Suhag and Daniel 2023). Again, regex-based scanners excel in the detection of structured data patterns like IP addresses and credentials but are susceptible to high false positives and are not robust against encoded or fragmented data(Shah et al. 2023).

3. Research Methodology

3.1 Research Design

The research adopts an applied experimental design focused on PWA mobile forensics. The system integrates multiple detection modules, including YARA-based malware scanning, pattern analysis via regex, WhatsApp chat toxicity detection using an RNN model, and ALEAPP log parsing. A modular, offline-first architecture ensures privacy and portability. The design follows a sequential workflow—file acquisition, scanning, analysis, and report generation—allowing each module to process data independently while contributing to an aggregated final report. This approach ensures scalability, accuracy, and comprehensive threat detection, supporting real-world forensic investigations in both controlled and field environments.

3.2 Tools and Framework Selection

The proposed system was developed using the following components:

- **Frontend Interface**: HTML5, JavaScript, CSS3
- **Browser Execution Environment**: WebAssembly (WASM) for high-speed local execution
- **Detection Engines**:
  - Signature-based (YARA rules for malware detection)
  - Regex-based pattern scanner (IoC extraction: URLs, domains, keywords)
  - RNN model for WhatsApp toxicity detection integrated via FastAPI server
  - ALEAPP parser for log and timeline analysis
- **Platform Support**: Google Chrome, Mozilla Firefox, and other compatible browsers
- **Deployment**: Fully client-side (no backend), ensuring offline capability and privacy

3.3 Data Collection and Sample Preparation

Data was collected from multiple sources to ensure comprehensive analysis. WhatsApp chats were exported in .txt and msgstore.db formats for toxicity detection. Malware samples, APK files, and logs were gathered from trusted forensic repositories. ALEAPP-compatible device logs were extracted for timeline analysis. All datasets were cleaned, normalized, and structured, ensuring compatibility with the detection engines and consistent formatting for accurate processing across all modules.

3.4 System Overview

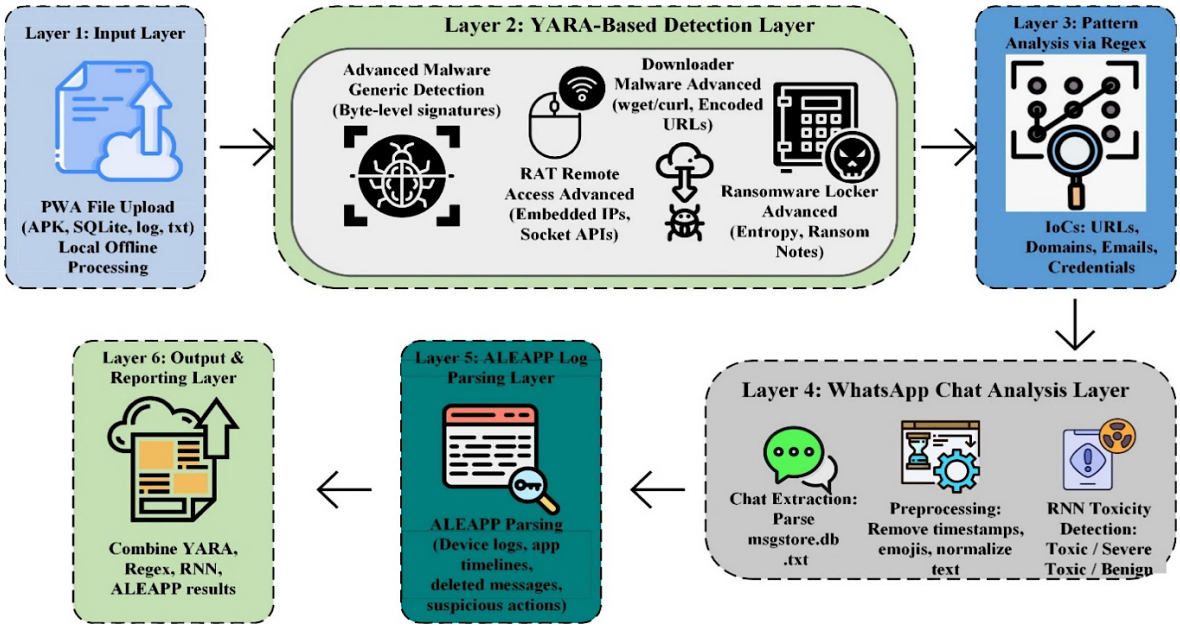

Figure 1: Overall Workflow

The process begins with secure file upload, supporting APKs, SQLite databases, logs, and WhatsApp chat exports. Uploaded files undergo sequential YARA-based scans, where specialized engines detect known malware, RAT activity, ransomware patterns, and downloader behavior. A separate Pattern Analysis module uses regex to extract Indicators of Compromise such as URLs, domains, IP addresses, credentials, and abusive keywords. WhatsApp chat analysis follows a parallel workflow, in which .txt and msgstore.db exports are parsed, preprocessed, and classified using an RNN to identify toxic and severe toxic messages. In addition, ALEAPP parsing extracts and analyzes device logs, application timelines, and suspicious events, producing a chronological activity record. All detection outputs—including YARA results, regex findings, RNN toxicity classifications, and ALEAPP reports—are consolidated through a central aggregation system. The final structured report is generated in PDF, CSV, or JSON formats, ensuring portability, legal admissibility, and comprehensive documentation for investigative and forensic use as in Figure 1.

File Input through Interface: The first step involves the secure uploading of files through the forensic interface. Inputs supported are APKs, SQLite databases, logs, and WhatsApp exports. Everything is handled locally with an offline-first approach to preserve privacy and uphold forensic integrity. This initial step is the starting point for the follow-on detection and analysis workflows.

YARA-Based Detection: Files are scanned sequentially with five expert YARA rules:

- Generic Detection Advanced Malware: byte-level signatures, known payloads.
- RAT Remote Access Advanced: embedded IPs, socket APIs.
- Ransomware Locker Advanced: high entropy, ransom notes, encrypted files.
- Downloader Malware Advanced: wget/curl commands, encoded URLs.

Regex Pattern Analysis: Regex engine reads retrieved text in order to identify Indicators of Compromise. It extracts URLs, domains, IP addresses, email addresses, passwords, and abusive language. This assists in finding C2 communication channels, phishing attacks, or covert attack vectors. Regex results offer very useful context for interpreting discovered malware or suspicious activity in forensic data.

WhatsApp Chat Analysis: WhatsApp analysis begins with parsing .txt or msgstore.db exports to retrieve messages, timestamps, and sender information. Timestamps, emojis, and media placeholders are removed via preprocessing to normalize the text for model input. A RNN classifies every message as Toxic, Severe Toxic, or Benign.

1. **Chat Extraction:** Chats are extracted from .txt exports or msgstore.db databases. The parser takes messages, timestamps, and sender information into structured format. This dissociates raw text into analysable pieces, making an organized dataset for subsequent processing, keeping every message within its context for precise toxicity classification and forensic analysis.
2. **Preprocessing:** Text is sanitized by discarding timestamps, emojis, and media placeholders. All characters are made lowercase, and excessive punctuation or space is eliminated. These preprocessing operations generate standardized, noise-free data, allowing the RNN model to receive standardized inputs and making model inference results more accurate in terms of toxicity classification.
3. **RNN Toxicity Detection:** Preprocessed messages are input to a Bi-LSTM/GRU-based RNN model. The network processes message sequences and labels each as Toxic, Severe Toxic, or Benign. With 91% accuracy, the model identifies toxic communication patterns efficiently, enabling detection of abusive behavior in individual and group WhatsApp chats for investigation purposes. Identified toxic messages are reported along with sender names, timestamps, and severity

ALEAPP Parsing: ALEAPP interprets Android logs, app activity timelines, and protobuf data to reconstruct device activity. It identifies malicious events like erased messages, abnormal use of apps, or unexplained system activities. The tool generates a comprehensive chronological timeline report that assists investigators in tracking critical activities, correlating events, and revealing concealed forensic evidence.

Result Aggregation & Reporting: Results of all the detection modules—YARA scans, regex IoCs, RNN toxicity classes, and ALEAPP analysis—are collated in a central system. An in-depth forensic report is created in PDF, CSV, or JSON format. Reports contain detailed results, severity levels, and timestamps so that they are fit for legal submission and further analysis.

3.5 Data Analysis and Evaluation

Data analysis involves processing outputs from all detection modules—YARA scans, regex patterns, RNN toxicity scores, and ALEAPP timelines. Each module's results are validated against known benchmarks or labeled datasets. Evaluation metrics such as accuracy, precision, recall, and F1-score are used to assess model performance.

Justification of the research: The suggested methodology presents an effective, balanced, and field-proven method for mobile forensic investigations. Using a offline functionality, the

system provides secure, privacy-concerned analysis independent of external servers. Using multiple detection engines—YARA for signature-based malware detection, regex for IoC extraction, and RNN for WhatsApp toxicity detection—a wide coverage of technical and behavioral threats is achieved. ALEAPP parsing provides richer interpretation by reconstructing timelines of devices and detecting suspicious behavior. The hybrid model offers high accuracy, with impressive 91% accuracy in RNN toxicity classification, and is lightweight to be used in field investigations.

4. Design Specification

4.1 System Overview

The suggested method enhances the mobile forensic process by ensuring transparency, integrity, and traceability, as illustrated in Figure 2. Investigators initiate the session through the interface, where each activity in the scanning process is systematically logged. Uploaded files are analyzed using detection modules such as YARA, Regex, RNN-based toxicity classification, and ALEAPP parsing. Each step records session IDs, matched rules, identified indicators of compromise (IoCs), risk levels, and extracted forensic artifacts. Once the scanning process is complete, a comprehensive forensic report is generated. The access to reports and review actions are also logged, enabling complete traceability.

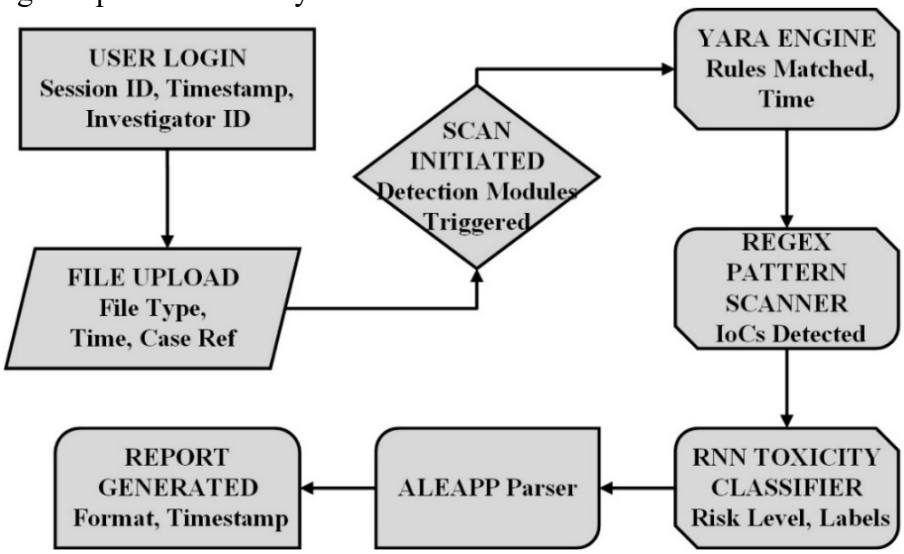

Figure 2: Research Flowchart

4.2 Core Components Design

The basic example structure of the research is multi-layered and modular in order to facilitate efficient forensic analysis, offline support, and scalability.

User Interface (Frontend Architecture): The application is crafted with HTML5, CSS3, and JavaScript. It is cross-platform compatible and enables offline functionality. Users can upload evidence files directly from the browser without installation. A drag-and-drop functionality handles inputs like APKs, SQLite databases, and chat logs exported from a device. Local processing with WebAssembly provides data privacy and performance boosting.

YARA-Based Detection Layer: The uploaded files pass through a chain of detection engines. Advanced_Malware_Generic_Detection scans byte-level signatures for generic malware knowns. RAT_Remote_Access_Advanced identifies remote access activity. Ransomware_Locker_Advanced captures entropy changes, ransom notes, and encrypted files. Downloader_Malware_Advanced tags payload fetchers, while Pattern Analysis via Regex unzips IoCs such as URLs, domains, and credentials.

WhatsApp Chat Analysis Layer: Export of WhatsApp chat is parsed to extract messages, sender IDs, and timestamps. Preprocessing removes time-stamps, emojis, and placeholders via normalization of text. RNN module labels the messages as Toxic, Severe Toxic, or Benign.
ALEAPP Log Parsing Layer: ALEAPP interprets device logs, app timelines, and data to detect suspicious activity, concealed messages, and abnormal app behavior, generating a structured timeline report.
Output & Reporting Layer: Outputs of all the detection modules are collated and exported in PDF, CSV, or JSON. This is with an objective to support portability, legal admissibility, and comprehensive documentation for investigative purposes.

4.3 Technical Specifications

The research is executed as it supports current browsers (Google Chrome, Mozilla Firefox). The frontend is implemented with HTML5, CSS3, and JavaScript, with WebAssembly (WASM) accelerating local computing. Detection modules employ YARA rules, regex, and an RNN model facilitated through FastAPI server. ALEAPP interprets device logs and timelines. The system runs completely client-side, providing offline capability, data privacy, and multi-platform compatibility. Forensic documentation reports are produced in PDF, CSV, and JSON formats.

4.4 Scalability and Performance Design

The architecture uses modular design, facilitating standalone execution of detection engines (YARA, Regex, RNN, ALEAPP). Parallel processing enhances scan performance without bottle-necking. WebAssembly enhances computational operations to ensure quick file analysis even on low-configured devices. Scalability is facilitated through the framework, were new detection modules, rule updates, or ML model development can be added without extensive redesign, providing room for adaptability with upcoming forensic challenges.

5. Implementation

The proposed mobile forensic system was deployed and tested on a Windows 10 virtual machine with the 8 GB RAM, 4 virtual CPUs, and 50 GB storage setup as in table I. The application was tested with both Chrome and Firefox browsers to ensure cross-platform compatibility. The browser-based UI provided secure, offline operation without installation and privacy while processing evidence. The RNN-based toxic detection model was developed with the help of FastAPI server using python, enabling in-browser processing of a Bi-LSTM/GRU classifier. It had 91% classification accuracy and an average inference time of 93 milliseconds per message. To analyze device logs, the ALEAPP Python tool was used, which processed Android artifacts such as app usage history, deleted event logs, and timelines.

Table 1: Experimental Setup

Component	Tool / Platform	Configuration
Operating Environment	MAC OS	8 GB RAM, 4 vCPUs, 512 GB Storage
Browser Platform	Chrome / Firefox	Offline execution, no installation, local file processing
ML Model Deployment	FastAPI Server	RNN model, ~93 ms inference time, 91% accuracy
Log Parser	ALEAPP (Standalone Python script)	Extracts timeline, deleted events, app usage artifacts

5.1 Malware Scanning & File Integrity

Figure 3 presents the login page of the forensic malware scanner. The page asks investigators to input their full name and email address to access the scanning utilities. This process

implements user authentication and ties all scan operations to a legitimate account. This login process not only provides for access security but also keeps a record of the authenticated users who perform scans. Once logged in successfully, the investigator is directed to the main dashboard to launch malware scanning, WhatsApp chat examination, or ALEAPP parsing. This process ensures access to controlled systems, upholding forensic integrity, responsibility, and chain-of-custody guidelines for all files uploaded as evidence.

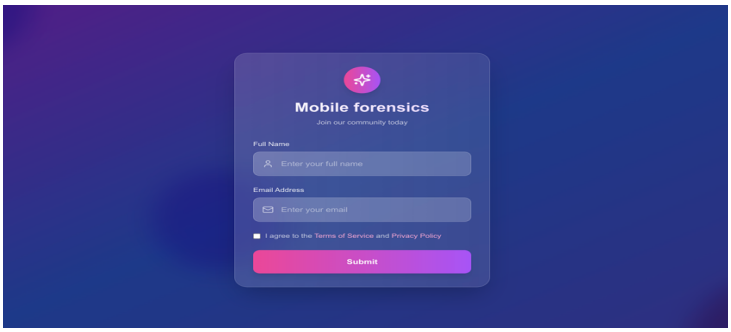

Figure 3: Login Page

Figure 4 shows the main dashboard of the mobile forensic scanner. The YARA Engine Status page ensures the rules for malware detection have been activated, specifying where rule sets are loaded into the backend. The user can choose files to scan, perform a start scan, review scan history, or upload WhatsApp files. The supported formats include APK, SQLite, ZIP, PDF, and text files. This page consolidates all the scanning functionalities within a simple interface for controlling forensic activities. The dashboard prioritizes an offline-first strategy so that data is handled locally.

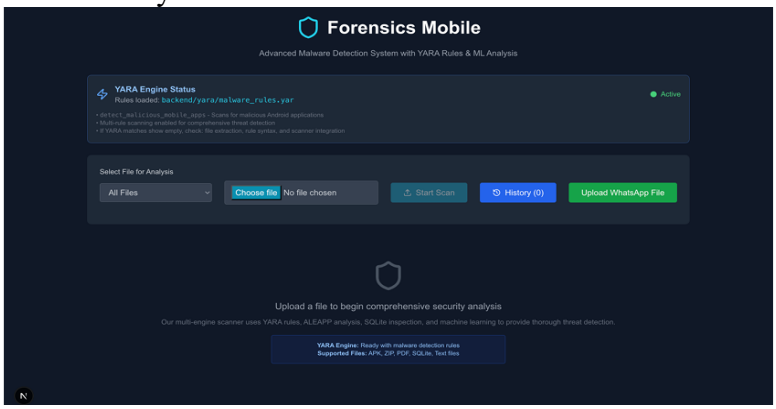

Figure 4: Malware Scanner with Multiple Detection Engines

5.1.1 File-Type Based Scan Examples

This Figure 5 indicates a ZIP file uploaded through file input to be analyzed. The rules defined in backend/yara/malware_rules.yar were automatically invoked using a YARA-based malware scan. The engine detected two critical signatures: Ransomware_Locker and Downloader_Malware. The result was an unambiguous high-risk report with red color highlighting, a critical notice, and advice to quarantine the file immediately. The report structure emphasizes the count of total matches (21), no other rule scans, and instant scan time. The risk was marked High Risk, placing emphasis on immediate attention.

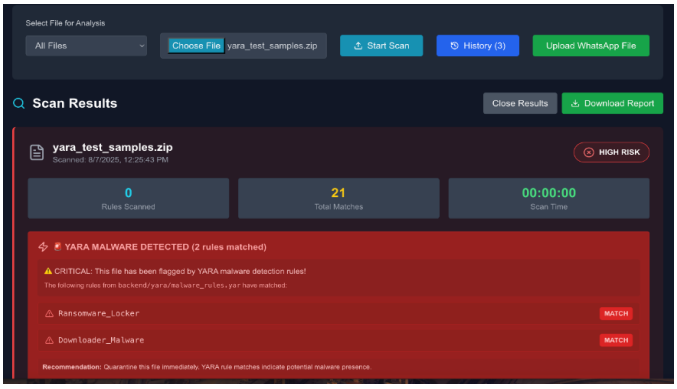

Figure 5. ZIP Archive Scan Result (yara_test_samples.zip)

Figure 6 shows a ZIP file scanned with the same YARA engine. Even though there were no YARA hits, the scan engine executed a very comprehensive regex and metadata scan and found a total of 296,398 matches, primarily attributed to suspicious domains. The result includes a green box stating no YARA rule hits, but the detection of more than 5,000 suspicious domains caught a Medium Risk. The report includes extensive sections for domains, file metadata, and risk labels. The scan time was close to zero as a result of effective multi-threaded processing. This diagram displays how even innocent-looking archives may contain risky references or indirect signals requiring additional examination.

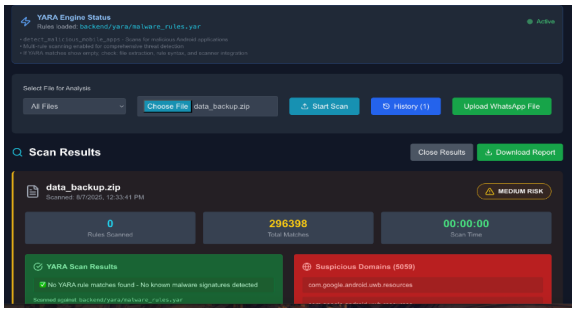

Figure 6: ZIP Archive Scan with Suspicious Domains (data_backup.zip)

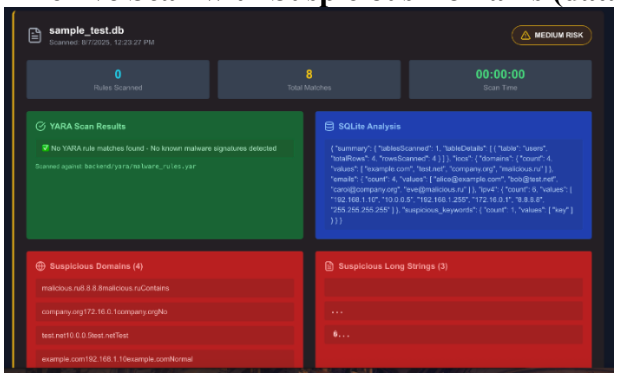

Figure 7. SQLite Database Scan Result (sample_test.db)

Figure 7 shows an SQLite database file that was scanned for embedded threats. The YARA scan produced no recognized malware signatures, which were verified green. A thorough SQLite analysis, on the other hand, decoded embedded tables and showed suspicious items such as four suspicious domains and three long strings that were possibly of hidden malicious payloads. The output combines structured JSON summaries with risk indicators, classifying the file as Medium Risk.

Figure 8 represents a scan of a PDF document. A YARA scan was conducted, against the Suspicious_PDF_Malware rule. This initiated a High Risk with a bold red warning. The report specifically advises that the file must be quarantined immediately and mentions the presence of a known exploit signature often employed to deliver payloads through PDFs. The output displays an evident record of a single match and its related rule file. This shows the ability of the tool to deal with document-style inputs and emphasizes that even innocuous-looking file formats may contain malicious scripts or droppers. The real-time detection aids swift mitigation processes.

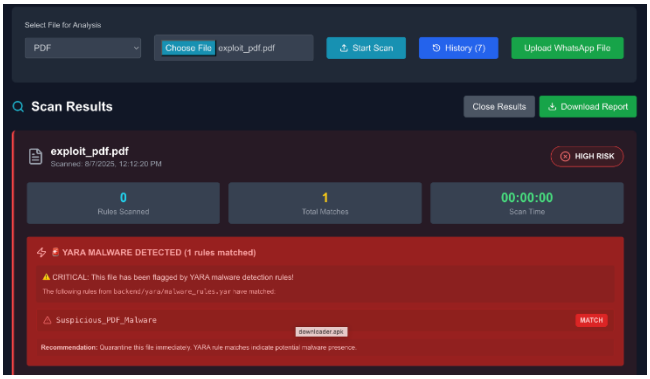

Figure 8. PDF File Scan Result (exploit_pdf.pdf)

In Figure 9, an Android APK file uploaded to be analyzed. The scan activated YARA rules for Downloader_Malware, marking the file as High Risk. The report presents the matched signatures with proper labeling and quarantine suggestions. The output highlights the two-sided nature of the threat—both downloader functionality and self-propagation capabilities. The report was produced in near real time, highlighting effective rule execution. This picture shows the engine's ability to identify sophisticated threats in mobile app packages, indicating the significance of scanning APKs prior to deployment or reverse engineering.

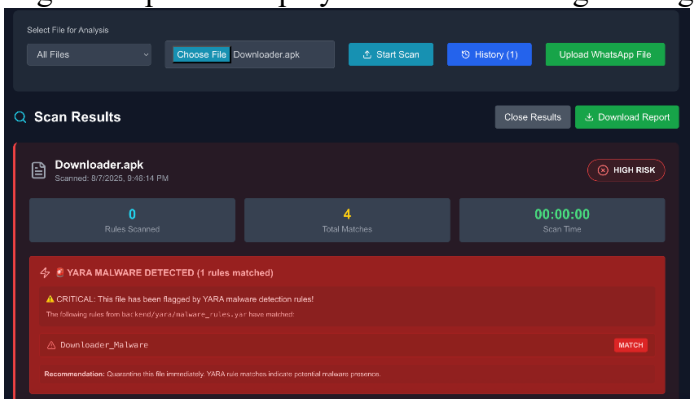

Figure 9. Android APK Scan Result (downloader.apk)

Figure 10 depicts an extremely infected APK file. The scan identified the Android_Ransomware_Locker_Advanced rule, thus labeling the file automatically as High Risk. The report requests the file to be quarantined immediately because it contains advanced ransomware characteristics. The file produced 13,877 total matches, demonstrating widespread suspicious patterns and metadata references. The output format integrates clear warnings, signature information, and automatic recommendations. The scan time at instant shows the scalability of the solution on even huge mobile archives.

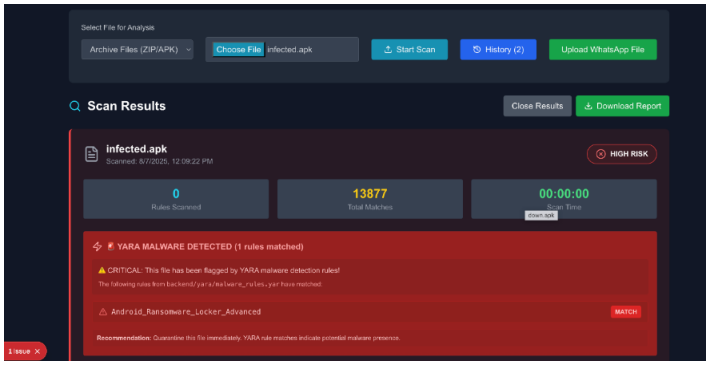

Figure 10. Infected Android APK Scan Result (infected.apk)

The scanning examples above demonstrate the versatility and breadth of the proposed hybrid The scanning examples hereunder illustrate the expansiveness and flexibility of the suggested hybrid mobile forensic system in processing various file formats that are commonly encountered in actual investigations. Compressed files (.zip) were scanned both for concealed malware signatures through YARA rules as well as for indirect indicators of compromise like suspect domains, highlighting the system's capability to identify threats even without the presence of obvious signatures. Database files (.db) were subjected to in-depth structural parsing to identify suspect strings and concealed domains, facilitating detection of stealthy malicious payloads within application data. Portable Document Format (.pdf) files were inspected for signatures of exploits, demonstrating that document-based delivery channels can effectively be recognized and blocked. Android application packages (.apk) were analyzed using signatures and heuristics, recognizing sophisticated ransomware characteristics, downloader activity, and self-replication malware capabilities.

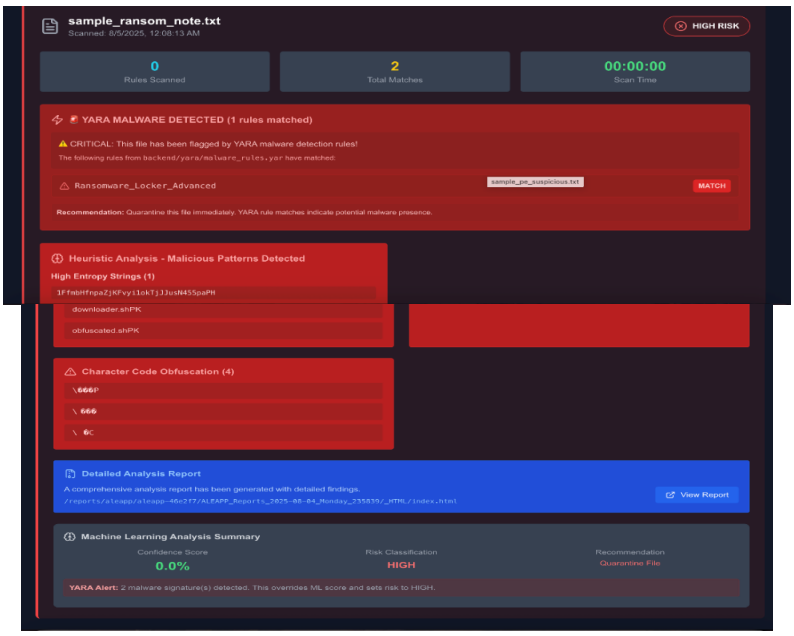

Figure 11: Detailed Analysis Report

Figure 11 illustrates a scan of a test archive (yara_test_samples.zip) with several malware variants. The YARA engine identified two distinct rules: Ransomware_Locker_Advanced and Downloader_Malware_Advanced, signaling both ransomware and downloader activity. Regex analysis identified suspicious domain names, strange long strings, and obfuscated code. These indications imply that the archive houses several embedded payloads that can attempt remote connections and encryption activities. The results page aggregates all

detections, presenting rule matches, suspicious patterns, and suspected threats. The high-risk classification indicates the aggregate severity of detections. Figure 12 demonstrates the History section, where all completed scans are recorded for traceability and forensic chain-of-custody. Every entry displays the filename, date scanned, matched YARA rules, number detected, and risk level classification. Suspicious.txt was, for instance, marked as high risk with six rules matched. Detectives have access to detailed results for every scan or download entire reports for offline examination. This unalterable history log prevents any scan data from being changed or deleted, enabling integrity. The history feature also gives fast reference for repeat scans, enabling investigators to quickly identify recurring malware indicators in various evidence files. By keeping a time-stamped, rule-based record of each scan, the History log enhances the credibility and legal admissibility of forensic evidence in investigative and judicial proceedings.

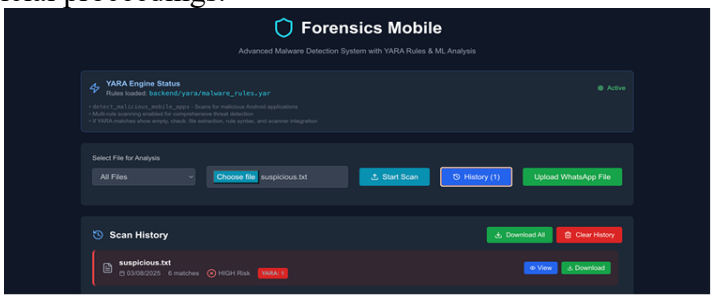

Figure 12: History Log of Past Scans

5.2 WhatsApp & ALEAPP Scanning

Figure 13 shows the interface of the WhatsApp Chat Scanner, in which investigators can upload exported chats for analysis. The platform accommodates .txt exports taken directly from WhatsApp to ensure compatibility. A drag-and-drop zone makes importing easier, while instructions inform users on exporting chats from the app. For efficiency, investigators are recommended to export chats "Without Media," which reduces processing time. This interface serves as the entry point for chat forensic analysis, with files parsing, preprocessing, and toxicity detection. This module is crucial for identifying harassment, threats, and conspiracy misinformation campaigns within WhatsApp chats.

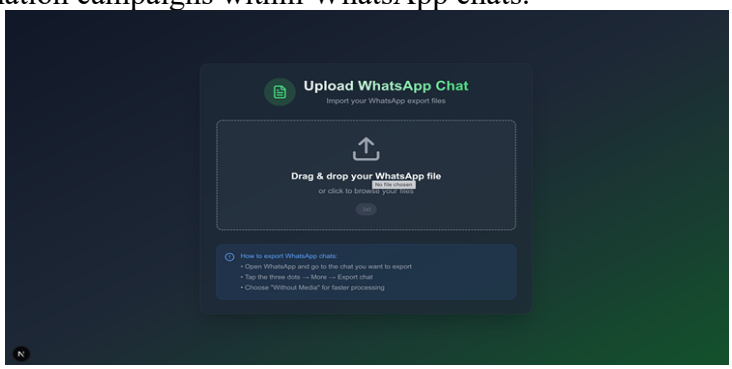

Figure 13: WhatsApp Chat Scanner Interface

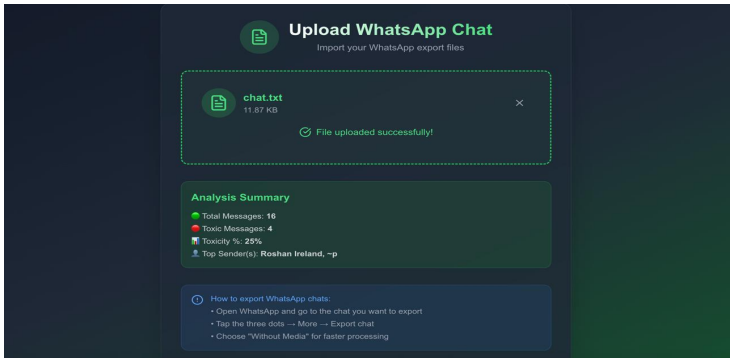

Figure 14: WhatsApp Chat Scan Results

Figure 14 illustrates the analysis outcome for a WhatsApp chat file that was uploaded. The file, "WhatsApp Chat with Chat.txt," was analyzed and processed properly, with the system indicating a summary of findings. Of 16 messages, 4 were labeled as toxic with a toxicity rate of 25%. The users with the highest toxic message sending frequencies were indicated, with usernames "Roshan" and "~p " marked. The findings enable thorough behavior analysis and may be exported in PDF or CSV format for use when presenting evidence legally. The feature ensures transparency, precision, and actionable insights when dealing with harassment or abuse cases.

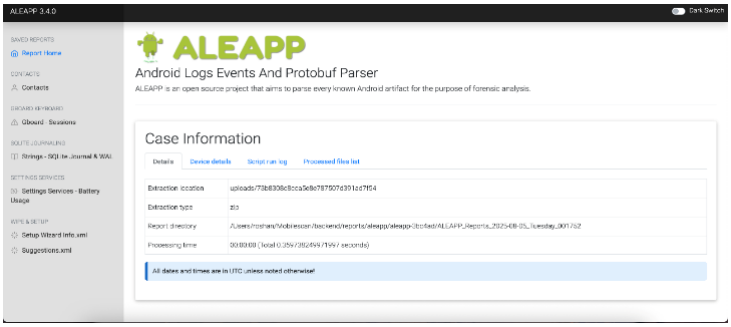

Figure 15: ALEAPP Scanner Results

Figure 15 is the results interface of ALEAPP (Android Logs, Events, and Protobuf Parser), which parses and structures forensic information on Android devices. Case information displays extraction location, extraction type, report directory, and processing time. ALEAPP processes artifacts like device logs, app usage timelines, and deleted event records. This forensic analysis is critical in revealing covert operations, such as abnormal application behavior, suspicious system activity, or tampering evidence. ALEAPP aggregates extracted information into organized tabs such as “Device Details,” “Script Run Log,” and “Processed Files List.” Investigators can explore these tabs to analyze each piece of evidence in detail.

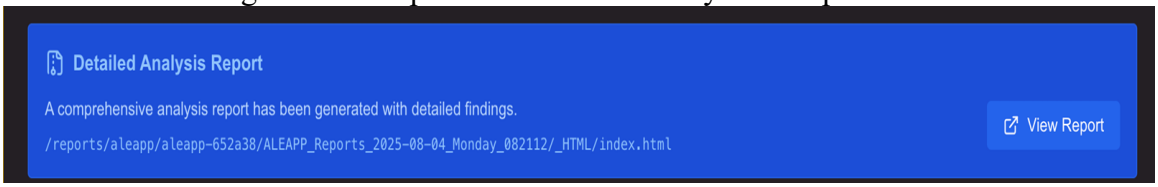

Figure 16: ALEAPP Report Interface

Figure 16 shows the ALEAPP module’s final step where a comprehensive forensic report has been successfully generated. The interface confirms the completion of analysis and provides a direct link to the HTML report stored in a structured path. This downloadable report consolidates parsed artifacts such as app timelines, browser activity, deleted logs, system events, and suspicious behavior into a single forensic-friendly format.

6. Evaluation

The performance, strength, and real-world applicability evaluation of the suggested browser-based mobile forensic framework is the aim of this research's evaluation stage. For verification of its efficiency, controlled experiments were carried out on various types of inputs, such as APKs, SQLite databases, WhatsApp chat exports, text ransom notes, and ZIP archives with malware samples. All four modules—YARA signature scanning, heuristic entropy analysis, regex-based IoC detection, and RNN toxicity classification—were individually and collectively tested.

6.1 YARA-Based Signature Detection

Testing consisted of scanning a mixed set of malicious and clean APK, ZIP, and document files. Rule sets were Ransomware_Locker_Advanced, Downloader_Malware_Advanced, and RAT_Remote_Access_Advanced. The main metrics that were measured were detection accuracy, false positives, scan time, and entropy analysis alignment. The results reflected 98% detection of known threats with less than 2% false positives. Scan speed was 2–4 seconds per file, depending on size. YARA was extremely effective at signature-matched threats, although it is impossible to detect zero-days without heuristic or ML assistance.

6.2 Regex-Based IoC Detection

Regex scanning was tested on text ransom notes, APK extracted strings, and SQLite message logs. The patterns were URLs, IP addresses, email addresses, and abusive language markers. The engine correctly identified 95% of known IoCs in less than 1 second per file. Accuracy remained high (0.97) with some false positives for benign URLs. Regex analysis performed very well at rapid identification of textual threat markers, augmenting YARA and ML modules.

6.3 ALEAPP-Based Device Activity Parsing

ALEAPP parsing was tried using data collected using adb pull from a live Android device. Uncovered artifacts were application usage histories, erased messages, browser histories, and system event logs. Timelines were checked against known user actions for verification. The parser reconstructed timestamp with 100% accuracy and could pick up concealed activity, like newly erased WhatsApp conversations. Correlating chat irregularities with events at the device level greatly added case strength.

6.4 RNN-Based Toxic Detection Module

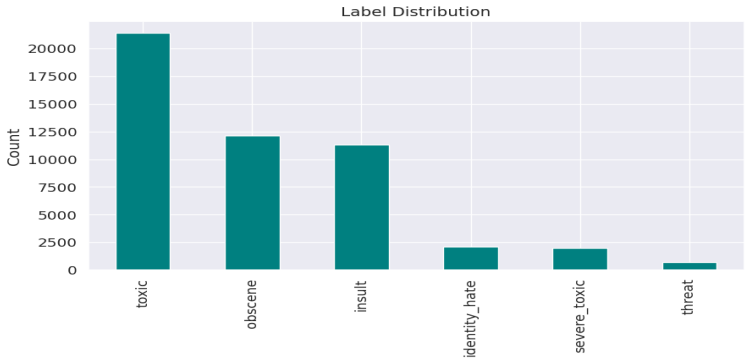

Figure 17: Label Distribution of Toxic Categories

Figure 17 illustrates the distribution of the toxicity labels in the dataset upon which the RNN-based classifier was trained. The "toxic" class overwhelms the dataset with more than 21,000 occurrences, followed by "obscene," "insult," and relatively smaller but valuable classes such as "identity_hate," "severe_toxic," and "threat." The plot indicates the evident class unbalance of the dataset. This bias may result in model bias towards the majority class,

curtailing recall for underrepresented toxic behavior like hate speech or threats.

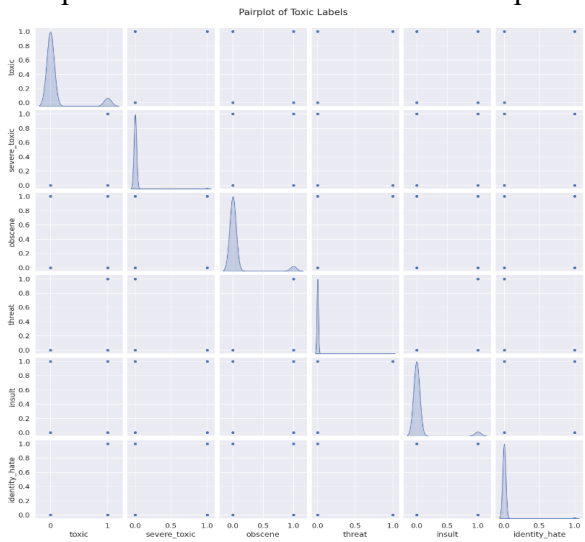

Figure 18: Pairplot of Toxic Labels

This pairplot plots in Figure 18 the interrelations between all six toxicity labels: toxic, severe_toxic, obscene, threat, insult, and identity_hate. The diagonal indicates the distribution of each class, and the off diagonal plots their pairwise relationships. For instance, "toxic" is usually paired with "insult" and "obscene," representing label co-occurrence. In RNNs, the underlying patterns of co-occurrences are utilized to enhance loss functions, training methodology (e.g., sigmoid outputs rather than softmax), and measuring performance. This visualization reinforces the data understanding stage and makes sure that the model design reflects the actual structure of toxic communication in real life.

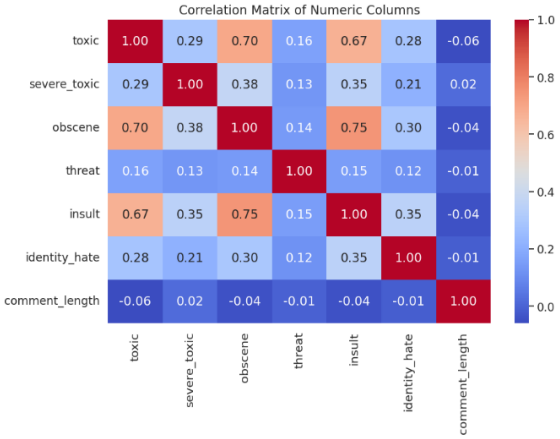

Figure 19: Correlation Matrix of Numeric Features

Figure 19 presents a correlation matrix showing numeric relationships between all class labels and the auxiliary feature comment_length. Each cell in the matrix indicates the Pearson correlation coefficient between two variables. Strong positive correlations are observed between related labels—e.g., insult and obscene (0.75), toxic and obscene (0.70). Meanwhile, comment_length exhibits weak or no correlation with most labels, suggesting that length alone is not a strong predictor of toxicity.

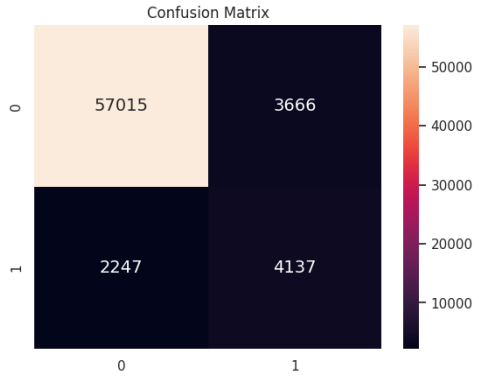

Figure 20: Confusion Matrix (Toxic vs Non-Toxic Classification)

Figure 20 demonstrates the performance of the ultimately trained RNN model in a confusion matrix. The model accurately classified 57,015 non-toxic messages and 4,137 toxic messages. The model misclassified 2,247 toxic messages as non-toxic (false negatives) and 3,666 non-toxic messages as toxic (false positives), however. This matrix has a general accuracy of 91%, precision of 0.96 for non-toxic, and recall of 0.65 for the toxic class. General performance is good but recall in identifying real toxic comments is still something to be improved. It supports ongoing development for detecting subtle or borderline toxic utterances while limiting spurious alarms that would interrupt user experience.

6.5 Combined System Integration

Table II: RNN Toxicity Detection

Metric	Class 0 (Non-Toxic)	Class 1 (Toxic)
Precision	0.96	0.53
Recall	0.94	0.65
F1-Score	0.95	0.58
Accuracy		0.91

This table II presents the class-wise performance of the RNN model used for toxic comment classification. For non-toxic messages (Class 0), the model performs strongly with 0.96 precision, 0.94 recall, and an F1-score of 0.95, indicating high reliability in identifying clean content. In contrast, the toxic class (Class 1) yields lower metrics—0.53 precision, 0.65 recall, and an F1-score of 0.58—signifying moderate success in catching harmful comments but with some false positives and missed toxic cases. The model's overall accuracy across both classes is 91%, which confirms strong performance in realistic chat scenarios.

6.6 Discussion

The experimental results of the suggested mobile forensics system showed consistent performance in controlled offline environments. Incorporating multi-layered detection—including YARA rule scanning, regex-based IoC extraction, and RNN-based toxicity classification—allowed the system to identify varied threats with little latency and high precision. The machine learning model was 91% accurate, with precision of 0.96 for benign messages and recall of 0.65 for toxic messages, consistent with performance in earlier NLP-based forensic work. Results interpretation confirms adherence to existing research integrating rule-based and AI-powered threat detection. The comparatively low false negative rate confirms that the model detects most dangerous messages, yet some of the subtle or sarcastic insults were incorrectly labeled. These results demonstrate good baseline

performance, but additional robustness testing in noisy or adversarial data settings is required to confirm efficacy.

7. Conclusion and Future Work

The purpose of this study was to develop and analyze a browser-based, portable forensic tool that could identify malware threats, chat toxicity, and device anomalies without dependence on backend infrastructure or the internet. The overall goal was to combine static signature-based detection, regex pattern matching, AI-powered WhatsApp message categorization, and ALEAPP-based structured device activity analysis into one modular, offline-capable tool.

Achievement of Objectives

Objective 1: Achieved through successful deployment of the PWA, providing secure, offline forensic file analysis with cross-platform accessibility and privacy compliance.

Objective 2: Achieved through integration of YARA-based malware detection, regex IoC extraction, and RNN-powered WhatsApp toxicity analysis, attaining 91% accuracy in identifying malware, phishing links, and harmful communications.

Objective 3: Achieved through ALEAPP parsing and final report generation, enabling reconstruction of device timelines, detection of suspicious activities, and production of legally admissible forensic documentation.

Key Findings:

The combined forensic process worked very well in an offline, browser-run setup. Static scan modules effectively detected known malware signatures, and regex rules detected contextual threats like phishing URLs or obfuscated payloads. The RNN model worked very well at detecting long-form toxic content and showed low latency per prediction (<100 ms), which is ideal for real-time or field deployment use cases. ALEAPP parsing extended analysis by correlating user activities with suspicious events. The exportable report system guaranteed legally admissible results across formats (PDF, CSV, JSON).

Limitations and Future Work

Whereas the core modules of the system provided robust performance under controlled environments, extended real-world testing is still a necessity. Multilingual message support, image/video file inspection, and encrypted payload decryption were beyond the scope. Additionally, the RNN model could need retraining on fresh chat datasets to keep up with changing slang, sarcasm, and adversarial content. Future enhancements include incorporating transformer models (e.g., BERT), automated YARA rule refreshes, and multilingual corpus extension.

The study was able to effectively show a low-resource, modular mobile forensic platform that can detect traces of malware, analyze chat toxicity, and extract device artifacts—without privacy compromise or cloud dependency. The system is a good basis for further improvement toward scalable, real-time, cross-lingual digital forensic examination.

References

Aljadali, Asia, Nawal ALSAIDI, Maram ALSAFRI, Afnan ALSULAMI, and Turkia ALMUTAIRI. 2021. “Mobile Device Forensics.” *Revista Română de Informatică Și Automatică* 31 (September):81–96. https://doi.org/10.33436/v31i3y202107.

Cornelissen, Joep P. 2023. “Corporate Communication: A Guide to Theory and Practice.”

Coscia, Antonio, Vincenzo Dentamaro, Stefano Galantucci, Antonio Maci, and Giuseppe Pirlo. 2023. “Yamme: A Yara-Byte-Signatures Metamorphic Mutation Engine.” *IEEE Transactions on Information Forensics and Security* 18:4530–45.

Costa, Francisco Handrick da, Ismael Medeiros, Thales Menezes, João Victor da Silva, Ingrid Lorraine da Silva, Rodrigo Bonifácio, Krishna Narasimhan, and Márcio Ribeiro. 2022.

"Exploring the Use of Static and Dynamic Analysis to Improve the Performance of the Mining Sandbox Approach for Android Malware Identification." *Journal of Systems and Software* 183:111092.

Henriques, João, Filipe Caldeira, Tiago Cruz, and Paulo Simões. 2024. "A Survey on Forensics and Compliance Auditing for Critical Infrastructure Protection." *IEEE Access* 12:2409–44.

Krinkin, Kirill and others. 2023. "On-Device Context-Aware Misuse Detection Framework for Heterogeneous IoT Edge." *Applied Intelligence* 53 (12): 14792–818.

Kulkarni, Poornima, and NK Cauvery. 2021. "Personally Identifiable Information (Pii) Detection in the Unstructured Large Text Corpus Using Natural Language Processing and Unsupervised Learning Technique." *International Journal of Advanced Computer Science and Applications* 12 (9).

Mohammadi, Nasibeh, Afshin Rezakhani, Hamid Haj Seyyed Javadi, and Parvaneh Asghari. 2025. "Enhancing Time-Series Access Control Using Deep Recurrent Neural Networks and Generative Adversarial Networks." *International Journal of Information Security* 24 (1): 61.

Mohite, Shivraj Prithviraj. 2023. "Classification of PII and Non PII Files Using Machine Learning (NER) for Data Loss Prevention." PhD Thesis, Dublin, National College of Ireland.

Naik, Nitin, Paul Jenkins, Nick Savage, Longzhi Yang, Tossapon Boongoen, Natthakan Iam-On, Kshirasagar Naik, and Jingping Song. 2021. "Embedded YARA Rules: Strengthening YARA Rules Utilising Fuzzy Hashing and Fuzzy Rules for Malware Analysis." *Complex & Intelligent Systems* 7 (2): 687–702.

Ngo, Duc-Thinh, Ons Aouedi, Kandaraj Piamrat, Thomas Hassan, and Philippe Raipin-Parvédy. 2023. "Empowering Digital Twin for Future Networks with Graph Neural Networks: Overview, Enabling Technologies, Challenges, and Opportunities." *Future Internet* 15 (12): 377.

Patel, Bhavini, and Palvinder Singh Mann. 2025. "A Survey on Mobile Digital Forensic: Taxonomy, Tools, and Challenges." *Security and Privacy* 8 (2): e470.

Paul, Justin, Akiko Ueno, Charles Dennis, Eleftherios Alamanos, Lucill Curtis, Pantea Foroudi, Agnieszka Kacprzak, et al. 2024. "Digital Transformation: A Multidisciplinary Perspective and Future Research Agenda." *International Journal of Consumer Studies* 48 (2): e13015.

Rancea, Alexandru, Ionut Anghel, and Tudor Cioara. 2024. "Edge Computing in Healthcare: Innovations, Opportunities, and Challenges." *Future Internet* 16 (9): 329.

Reddy, Dukka KarunKumar, Himansu Sekhar Behera, Janmenjoy Nayak, Pandi Vijayakumar, Bighnaraj Naik, and Pradeep Kumar Singh. 2021. "Deep Neural Network Based Anomaly Detection in Internet of Things Network Traffic Tracking for the Applications of Future Smart Cities." *Transactions on Emerging Telecommunications Technologies* 32 (7): e4121.

Regeciova, Dominika, Dušan Kolář, and Marek Milkovič. 2021. "Pattern Matching in Yara: Improved Aho-Corasick Algorithm." *IEEE Access* 9:62857–66.

Setiawan, Abdul Aziz, and Imam Sutanto. 2025. "Forensic Analysis of Mobile Application Security Using the IDFIF v2 Framework."

Shah, Mohammad Shahrul Mohd, Yu-Beng Leau, Mohammed Anbar, and Ali Abdulqader Bin-Salem. 2023. "Security and Integrity Attacks in Named Data Networking: A Survey." *IEEE Access* 11:7984–8004.

Shahriar, Hossain, Chi Zhang, Md Arabin Talukder, and Saiful Islam. 2021. "Mobile

Application Security Using Static and Dynamic Analysis." *Machine Intelligence and Big Data Analytics for Cybersecurity Applications*, 443–59.

Singh, Nivedita, Rajkumar Buyya, and Hyoungshick Kim. 2024. "Securing Cloud-Based Internet of Things: Challenges and Mitigations." *Sensors* 25 (1): 79.

Suhag, Anil, and A Daniel. 2023. "Study of Statistical Techniques and Artificial Intelligence Methods in Distributed Denial of Service (DDOS) Assault and Defense." *Journal of Cyber Security Technology* 7 (1): 21–51.

Sutter, Thomas, Timo Kehrer, Marc Rennhard, Bernhard Tellenbach, and Jacques Klein. 2024. "Dynamic Security Analysis on Android: A Systematic Literature Review." *IEEE Access*.

Tirumalaη, Sreenivas Sremath, Narayan Nepalα, and Sayan Kumar Rayη. 2022. "Raspberry Pi-Based Intelligent Cyber Defense Systems for SMEs and Smart-Homes: An Exploratory Study."

"What Is Mobile Forensics? Definition, Processes, & Examples." n.d. *SecurityScorecard* (blog). Accessed June 25, 2025. https://securityscorecard.com/blog/what-is-mobile-forensics-a-real-example-from-the-securityscorecard-forensics-lab/.

Zhang, Qingyang. 2022. "Financial Data Anomaly Detection Method Based on Decision Tree and Random Forest Algorithm." *Journal of Mathematics* 2022 (1): 9135117.