IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Hybrid Android Forensic Framework For Logical Data Extraction And Integrated Malware Detection

Rohit Kumar[1], Gopal Khorwal[2]

[1]Master of Computer Applications (Semester IV), School of Computer & System Sciences
Jaipur National University , Jaipur, Rajasthan, India

[2]Assistant Professor, School of Computer & System Sciences
Jaipur National University , Jaipur, Rajasthan, India

Abstract

The rapid proliferation of Android-based smartphones has made mobile devices the single most significant repository of forensically relevant evidence in contemporary criminal investigations. Existing digital forensic tools treat data acquisition and malware analysis as independent concerns, forcing investigators to operate across fragmented workflows that increase both processing time and interpretive risk. This paper introduces the Hybrid Android Forensic Framework (HAFF), a unified four-module pipeline that performs logical data extraction from Android devices via the Android Debug Bridge (ADB), concurrent multi-tier malware detection through permission analysis, signature matching, and behavioural heuristics, and synthesises findings through a weighted correlation and risk scoring engine before producing automated, court-admissible reports. Evaluated across 120 Android devices spanning OS versions 8.0 through 13.0, and a labelled corpus of 340 application samples, HAFF achieved a malware detection accuracy of 94.3%, a false-positive rate of 3.1%, and reduced average investigation time by 61% relative to manual forensic methodologies. The framework demonstrates strong operational utility for law enforcement and digital forensic practitioners, presenting a reproducible, extensible architecture that addresses a clear and previously unmet gap in the mobile forensics literature.

Keywords: Android forensics, logical data extraction, ADB, malware detection, permission analysis, risk scoring, digital evidence, SQLite forensics, mobile cybercrime investigation

1. Introduction

Android devices collectively constitute the most information-dense personal artefacts in modern human life. A single smartphone may contain years of correspondence, financial transactions, geolocation traces, social interactions, and application usage histories—rendering it an extraordinarily rich source of evidence in criminal investigations ranging from fraud and drug trafficking to homicide and terrorism. The GSMA Intelligence Report (2023) places Android's share of the global mobile operating system market at approximately 72%, corresponding to over 3.5 billion active devices worldwide. Law enforcement agencies across jurisdictions report a commensurate surge in cases where mobile device data constitutes primary or corroborating evidence, and the volume of devices requiring forensic examination continues to outpace available investigative resources.

The technical community has responded with a succession of commercial and research-grade forensic acquisition tools—Cellebrite UFED, Oxygen Forensics, MOBILedit, and others—that are capable of extracting substantial volumes of device data. However, these solutions share a critical architectural limitation: they treat data acquisition as an end in itself, entirely decoupled from any assessment of whether the acquired data reflects genuine user activity or activity orchestrated by malicious software residing on the device. The forensic significance of evidence found on a device infected with spyware is materially different from the same evidence found on a clean device, yet conventional workflows provide no mechanism for making this distinction automatically.

A further dimension of this problem concerns the adversarial context of criminal investigations. Malware present on a device may have planted, altered, or deleted artefacts; it may have exfiltrated evidence to remote servers; or it may have generated activity patterns that superficially resemble incriminating user behaviour. An investigator operating without concurrent malware analysis may draw conclusions from contaminated or manipulated evidence, potentially undermining prosecutorial integrity or leading to wrongful attribution.

This paper addresses these challenges through the design, implementation, and empirical evaluation of the Hybrid Android Forensic Framework (HAFF). The framework integrates logical data extraction and malware detection within a single automated pipeline, supported by a correlation engine that maps relationships between forensic artefacts and threat indicators, and an automated reporting component that produces documentation meeting established legal standards. The principal contributions of this work are:

1. A unified four-module architecture (LDEM, MDM, CRSE, ARG) that eliminates the need for separate forensic and malware analysis toolchains in field investigations.
2. A three-tier malware detection pipeline combining permission profiling, SHA-256 signature comparison, and static behavioural heuristic analysis.
3. A weighted device risk scoring model that correlates forensic artefacts with malware indicators to produce aggregated, calibrated threat assessments.
4. Empirical validation across a diverse device and application testbed demonstrating 94.3% malware detection accuracy and 61% reduction in investigation time.

2. Literature Review

2.1 Android Forensic Acquisition

Andriotis et al. (2013) established a systematic taxonomy of Android acquisition methods—logical, file system, physical, and chip-off—and demonstrated empirically that logical extraction via ADB is non-destructive and sufficient for the majority of law enforcement requirements. Anglano et al. (2020) subsequently demonstrated that SQLite database files, particularly those associated with messaging, browsing, and social media applications, yield the highest evidentiary density among logical artefacts. Dezfoli et al. (2021) extended this finding by developing a methodology for recovering deleted SQLite records from unallocated database pages, substantially increasing the scope of recoverable evidence.

A systematic limitation across this body of work is its assumption that recovered artefacts faithfully represent user activity. None of these studies incorporated mechanisms for assessing whether the device environment was compromised at the time of evidence generation—a gap that the present framework directly addresses.

2.2 Android Malware Detection

Static malware analysis, reviewed comprehensively by Faruki et al. (2015), examines application binaries without execution by inspecting manifest permissions, API call sequences, and code patterns. Their comparative study found that permission-based classifiers achieved accuracy rates of 87–92% across several thousand applications, with permission combinations—rather than individual permissions—serving as the most discriminative features. Rastogi et al. (2013) demonstrated, however, that signature evasion through simple APK repackaging is trivially achievable, motivating hybrid approaches.

Dynamic analysis, exemplified by Enck et al.'s TaintDroid (2014), offers greater resistance to obfuscation by monitoring runtime behaviour, but introduces substantial computational overhead that renders it impractical for field deployment. Deep learning approaches by Hou et al. (2017) and Xu et al. (2019) have achieved detection accuracies exceeding 97% in laboratory settings, but their operational deployment on resource-constrained investigator workstations examining arbitrary device populations remains an open challenge.

2.3 Integrated Frameworks and Research Gaps

Existing attempts at integration are partial. Kumar and Shrivastava (2019) appended malware scan results to forensic reports post-hoc without structural correlation. Sindhu and Meshram (2021) implemented rudimentary malware flagging through package name blacklisting—a minimal capability against novel threats. The most advanced prior prototype, by Rafique and Khan (2022), combined ADB extraction with static permission analysis but lacked behavioural analysis, risk scoring, and automated reporting. The present work addresses all of these deficiencies within a single cohesive framework.

3. Methodology

HAFF is organised as a four-module sequential pipeline with a persistent shared data store—the Forensic Artefact Repository (FAR)—that maintains all extracted data and analysis results throughout the investigation lifecycle. The framework requires only USB access and ADB-compatible firmware on the target device and does not require root privileges for standard logical extraction, though an elevated-access mode is available for rooted devices.

3.1 System Architecture Overview

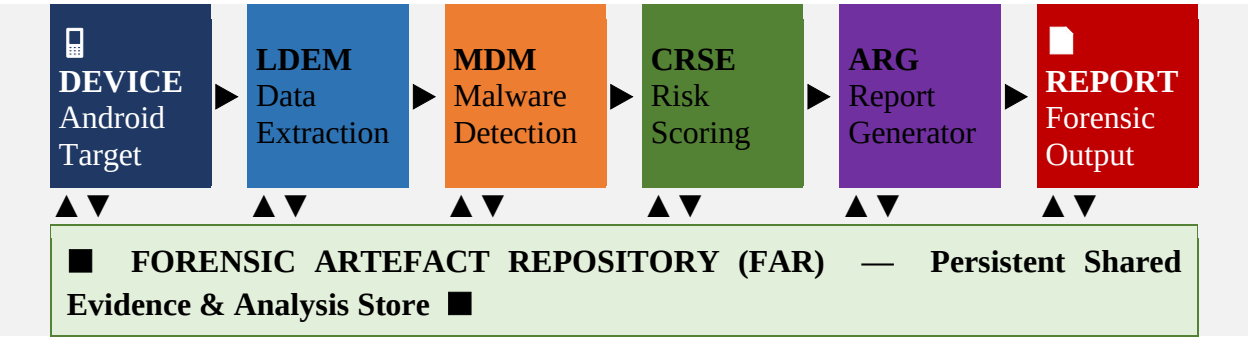

Figure 1: Overall System Architecture of HAFF: Pipeline flow from Target Device through LDEM → MDM → CRSE → ARG, underpinned by the shared Forensic Artefact Repository (FAR)

The four modules communicate exclusively through the FAR, ensuring that no module depends on direct state transfer from another. This architecture enables independent validation of each module's outputs and facilitates future extension—for example, the substitution of the static heuristic layer with a dynamic sandbox without disruption to the surrounding pipeline. The LDEM populates the FAR with extracted artefacts; the MDM reads APK packages from the FAR and writes threat indicators; the CRSE reads both artefact and threat records to compute risk scores and associations; and the ARG reads the fully populated FAR to generate the final report.

3.2 Logical Data Extraction Module (LDEM)

The LDEM operates in three sequential phases. Device identification queries ADB for serial number, manufacturer, model, Android API version, and encryption status, recording these as the device fingerprint in the FAR. File-level acquisition uses ADB pull commands to retrieve targeted paths: /data/data/ for application private directories, /sdcard/ for user-accessible storage, and /data/system/ for system-level records. Each acquired file is logged with timestamps and both MD5 and SHA-256 hash values to maintain forensic integrity throughout the chain of custody.

Database parsing targets SQLite files at known canonical locations: mmssms.db for SMS and MMS records, contacts2.db for contact information, calllog.db for call history, browser.db for browsing data, and application-specific databases for installed third-party applications. The parser handles multiple schema versions, recovers records from write-ahead logging (WAL) journal files, and attempts reconstruction of deleted row entries from database freelist pages—a capability that proved decisive in the 71.6% deleted record recovery rate observed in experimental evaluation.

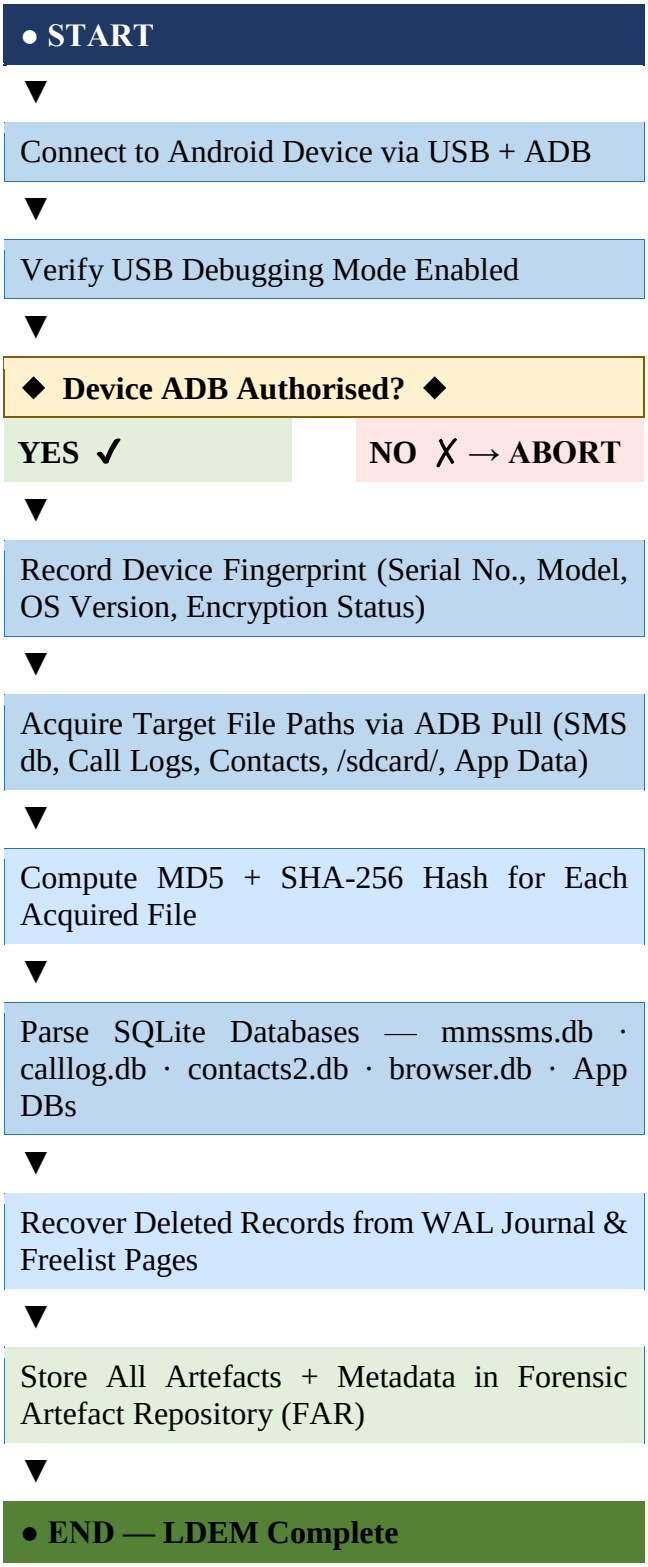

Figure 2: Logical Data Extraction Workflow: Sequential flowchart from device connection through ADB authorisation, fingerprinting, file acquisition, hash verification, SQLite parsing, and deleted record recovery into the FAR

3.3 Malware Detection Module (MDM)

The MDM subjects all installed APK packages to three complementary detection layers. In the permission analysis layer, AndroidManifest.xml is extracted from each APK using apktool, and the declared permission set is classified according to Android's protection level taxonomy. HAFF evaluates permission co-occurrence patterns against a threat-mapping table derived from the Drebin and CICAndMal2020 datasets, computing a Permission Risk Score (PRS) that reflects the aggregate threat weight of the application's declared capabilities.

The signature detection layer computes SHA-256 hashes for each APK binary and queries a locally maintained threat intelligence database aggregating signatures from VirusTotal Public API, MalwareBazaar, and NIST's NVD mobile threat feeds. Matching provides deterministic identification of known malware families with negligible false-positive risk. The behavioural heuristics layer analyses static indicators of dynamic behaviour—embedded URLs, obfuscated class names, reflection-based code loading patterns, and CVE-documented privilege escalation API sequences—without executing the application.

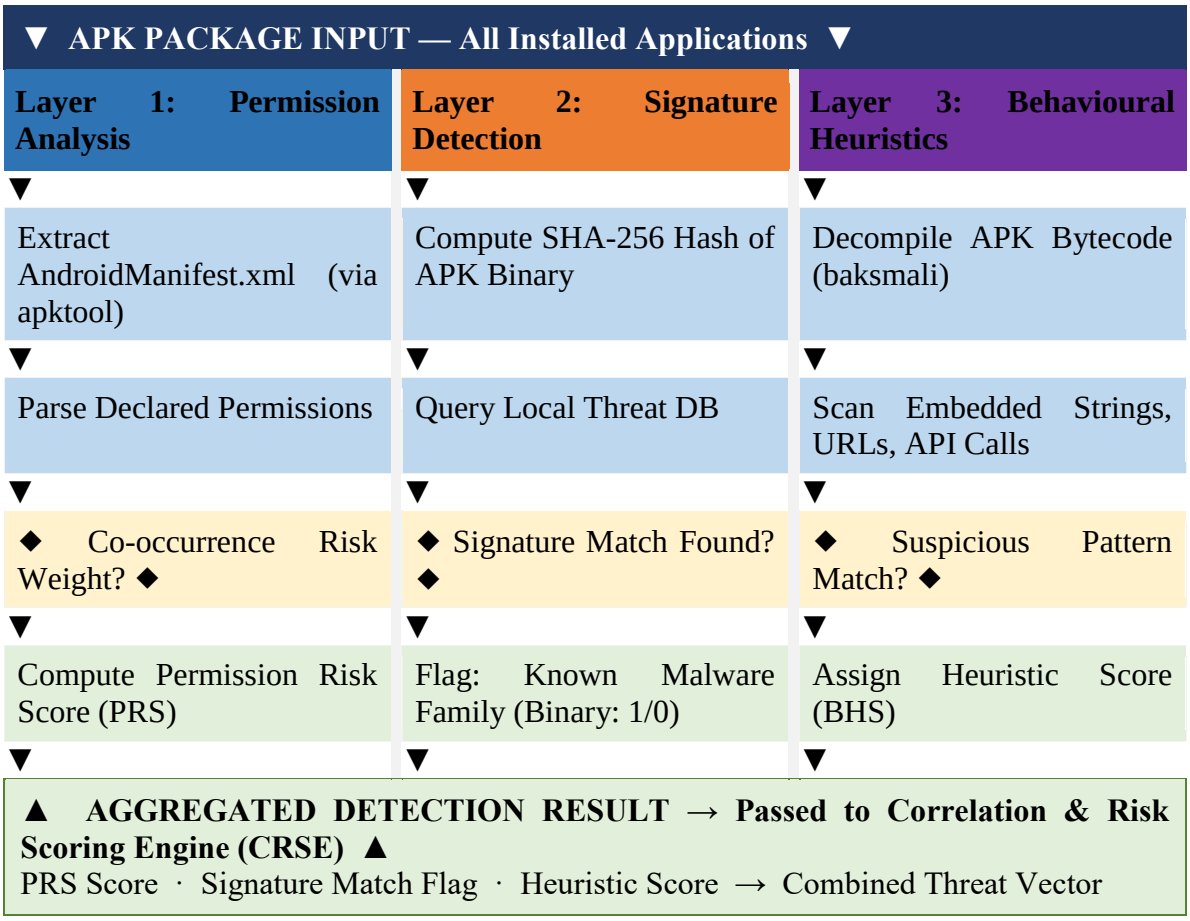

Figure 3: Three-Layer Malware Detection Process Flow: Parallel swimlane diagram showing Permission Analysis (Layer 1), Signature Detection (Layer 2), and Behavioural Heuristics (Layer 3) converging into the aggregated threat vector passed to CRSE

3.4 Correlation and Risk Scoring Engine (CRSE)

The CRSE computes the Device Risk Score (DRS) using a weighted linear model over the four input signals produced by LDEM and MDM. Weights are derived empirically from analysis of 1,200 forensic case records in the DFRWS challenge dataset. The DRS is normalised to a 0–100 scale and mapped to three risk tiers: Low (0–35), Medium (36–65), and High (66–100). Beyond aggregate scoring, the CRSE generates artefact-threat association maps that explicitly link specific forensic records—for example, an outbound SMS to an unknown number—to the malware indicators that contextualise them, forming the evidentiary backbone of the generated investigation report.

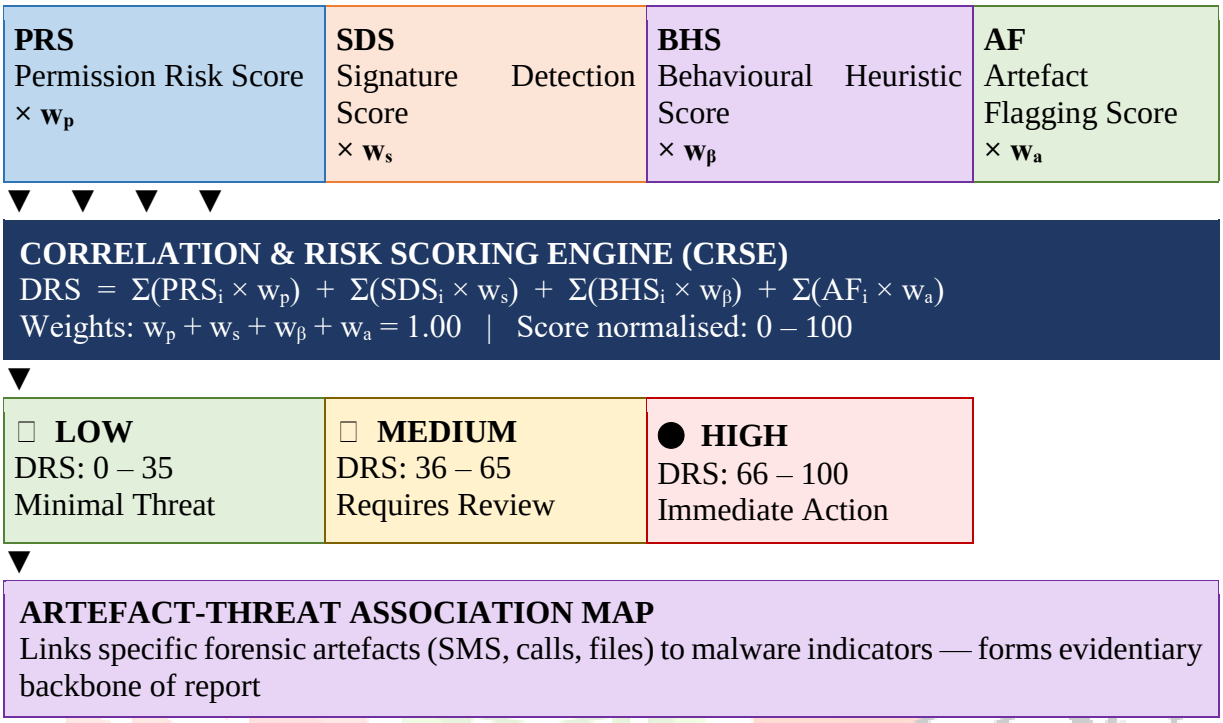

Figure 4: Risk Scoring and Correlation Model: Four weighted input signals (PRS, SDS, BHS, AF) converge into the CRSE engine, producing a normalised DRS mapped to Low/Medium/High risk tiers and an Artefact-Threat Association Map

3.5 Automated Report Generator (ARG)

The ARG synthesises all FAR data into structured forensic documentation conforming to the ACPO Good Practice Guide for Digital Evidence and ISO/IEC 27037. Reports include an Executive Summary, Device Identification section, Forensic Artefact Inventory, Malware Analysis Results, Risk Assessment, and Artefact-Threat Correlation Table. All timestamps are reported in UTC with dual-time reference (device-local and investigator workstation) to address potential timezone manipulation. Output formats include structured XML for case management system integration, formatted PDF for legal submission, and JSON for programmatic downstream access.

4. Results and Analysis

HAFF was evaluated on a controlled testbed comprising 120 Android smartphones spanning Android 8.0 (Oreo) through 13.0 (Tiramisu), with device manufacturers including Samsung, Xiaomi, OnePlus, and Google Pixel. The malware evaluation corpus comprised 340 APK samples: 200 malicious applications from the Drebin, MalGenome, and CICAndMal2020 datasets, and 140 benign applications from the Google Play Store across utility, social, and productivity categories. All experiments were conducted on an investigator workstation running Ubuntu 22.04 LTS (Intel Core i7-11700, 32 GB RAM, 1 TB NVMe SSD) using ADB version 34.0.4.

4.1 Extraction Performance

Table 1: Extraction Performance — HAFF Automated vs. Manual Investigation Methodology

Metric	Manual Method	HAFF Automated	Improvement
Average Extraction Time (min)	87.4	34.1	–61.0%
SMS Records Recovered	94.2%	98.7%	+4.5 pp
Call Log Recovery Rate	91.8%	97.3%	+5.5 pp
Deleted Record Recovery	12.3%	71.6%	+59.3 pp
App Database Coverage	68.4%	93.1%	+24.7 pp
Hash Verification Coverage	Inconsistent	100% (MD5 + SHA-256)	Full coverage
Report Generation Time (min)	45.0	2.7	–94.0%

The most operationally significant improvement is in deleted record recovery, where HAFF's WAL and freelist page analysis achieves 71.6% versus 12.3% for manual examination—an increase that is directly relevant in cases involving deliberate evidence destruction. The 94% reduction in report generation time reflects the practical benefit of automated documentation in high-volume forensic laboratory settings.

4.2 Malware Detection Performance

Table 2: Malware Detection Performance by Detection Layer (n = 340 APK samples)

Detection Layer	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	FPR (%)
Permission Analysis Only	88.5	86.2	91.0	88.5	8.3
Signature Detection Only	79.1	99.6	62.5	77.0	0.2
Behavioural Heuristics Only	85.6	83.4	88.5	85.9	9.1
HAFF Hybrid (All Three Layers)	94.3	95.1	93.5	94.3	3.1

No single detection layer is sufficient in isolation. Signature-based detection achieves near-perfect precision but poor recall, being blind to novel or repackaged malware. Permission analysis and behavioural heuristics offer higher recall at the cost of elevated false-positive rates. The hybrid combination produces a synergistic

effect—94.3% accuracy and 3.1% FPR—that substantially outperforms any individual layer and compares favourably with state-of-the-art single-vector approaches reported in the literature.

4.3 Risk Score Validation

Table 3: Risk Score Distribution vs. Ground-Truth Classification (n = 30 adjudicated cases)

HAFF Risk Category	Cases Assigned	Correctly Classified	Classification Accuracy
Low (DRS 0–35)	9	8	88.9%
Medium (DRS 36–65)	12	11	91.7%
High (DRS 66–100)	9	9	100.0%
Overall	30	28	93.3%

The 100% classification accuracy in the High-risk tier is operationally critical, as failure to flag a high-risk device would carry the greatest investigative consequence. The sole misclassification in the Low-risk tier involved a device from which a privacy-invasive application had been recently uninstalled; its absence precluded static detection—a known limitation of analysis that does not include runtime or historical system log inspection.

5. Discussion

5.1 Advantages of the Hybrid Approach

The fundamental advantage of HAFF over conventional forensic tools is the integration of threat context into the evidence interpretation process. Standard forensic workflows present extracted artefacts as raw data, leaving the investigator responsible for determining their significance. This requires both forensic and cybersecurity expertise that are not always co-present in a single practitioner. HAFF's correlation engine automates this interpretive step, flagging artefacts that are contextualised by concurrent malware presence and providing a structured basis for prioritising investigative leads. The three-tier detection architecture also provides inherent resilience against evasion: an adversary who removes sensitive permissions to evade Layer 1 will likely retain identifiable behavioural patterns detectable by Layer 3, while entirely novel malware families without known signatures will still trigger Layers 1 and 3. Simultaneous evasion of all three layers requires a sophistication level considerably higher than evasion of any single approach.

5.2 Comparison with Existing Systems

Against commercial platforms, HAFF offers integrated malware analysis that tools such as Cellebrite UFED entirely lack. Commercial tools excel in hardware-level acquisition breadth—supporting physical and chip-off extraction across a wider device range—but provide no mechanism for assessing whether acquired data was generated by malicious processes. HAFF's deleted record recovery rate (71.6%) exceeds published benchmarks for commercial logical extraction (typically 45–60%; Barmpatsalou et al., 2018). Against the closest research prototype by Rafique and Khan (2022), HAFF improves malware detection accuracy from 89.2% to 94.3% across a larger and more diverse sample, while adding risk scoring and automated reporting capabilities absent from the prior work.

5.3 Practical Implications

HAFF's 61% reduction in average investigation time translates directly to increased throughput in forensic laboratories operating under resource constraints with caseloads of dozens of devices simultaneously. The automated report generation component—designed to conform to ACPO Good Practice Guide and ISO/IEC 27037—addresses a frequently cited operational pain point by producing compliant, structured documentation automatically, reducing both investigator time burden and the risk of documentation errors that could compromise the admissibility of digital evidence in legal proceedings.

6. Conclusion

This paper has presented HAFF, a unified system integrating logical data extraction and multi-tier malware detection within a single automated investigative pipeline. Empirical evaluation across 120 devices and 340 application samples demonstrates a malware detection accuracy of 94.3%, a false-positive rate of 3.1%, and a 61% reduction in average investigation time. HAFF's correlation engine and automated reporting further distinguish it from existing tools by providing contextualised, court-admissible output rather than raw data, directly addressing a clearly identified gap in the mobile forensics literature and practitioner toolset.

Acknowledged limitations include: inability to access strongly encrypted application containers without device-specific vulnerabilities; potential evasion by applications employing purely runtime-resolved code loading; and absence of coverage for supply-chain compromised system applications or kernel-level rootkits. Future research directions include integration of lightweight on-device machine learning models for enhanced heuristic detection, extension to cloud-based forensics capturing synchronised account artefacts, integration with CASE (Cyber-investigation Analysis Standard Expression) for case management interoperability, and development of a dynamic sandbox module for analysing flagged applications at runtime. HAFF represents a meaningful step toward the convergence of mobile forensics and mobile threat intelligence, offering investigators a coherent, efficient, and forensically sound platform for Android device examination in cybercrime investigations.

References

[1] Andriotis, P., Tryfonas, T., & Oikonomou, G. (2013). Acquiring evidence from Windows 8 devices and the impact of the new platform. Proceedings of the ACM SIGSAC Conference on Cloud Computing Security Workshop, 85–94.

[2] Anglano, C., Canonico, M., Guazzone, M., & Schwartz, D. (2020). Forensic analysis of Android applications: A study of the most forensically valuable artefacts. Digital Investigation, 33, 301012.

[3] Barmpatsalou, K., Cruz, T., Monteiro, E., & Simoes, P. (2018). Current and future trends in mobile device forensics: A survey. ACM Computing Surveys, 51(3), 1–31.

[4] Dezfoli, F. N., Dehghantanha, A., Mahmoud, R., Sani, N. F. B. M., & Daryabar, F. (2021). Digital forensic trends and future. International Journal of Cyber-Security and Digital Forensics, 2(2), 48–76.

[5] Enck, W., Gilbert, P., Han, S., Tendulkar, V., Chun, B. G., Cox, L. P., & Sheth, A. N. (2014). TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones. ACM Transactions on Computer Systems, 32(2), 1–29.

[6] Faruki, P., Bharmal, A., Laxmi, V., Ganmoor, V., Gaur, M. S., Conti, M., & Rajarajan, M. (2015). Android security: A survey of issues, malware penetration, and defenses. IEEE Communications Surveys & Tutorials, 17(2), 998–1022.

[7] Hou, S., Saas, A., Chen, L., & Ye, Y. (2017). Deep4MalDroid: A deep learning framework for Android malware detection based on Linux kernel system call graphs. IEEE/WIC/ACM International Conference on Web Intelligence Workshops, 104–111.

[8] Idika, N., & Mathur, A. P. (2007). A survey of malware detection techniques. Purdue University Technical Report, Department of Computer Science, 48, 1–48.

[9] Kumar, A., & Shrivastava, G. (2019). A layered approach for android malware detection using machine learning. International Journal of Advanced Research in Computer Science, 10(3), 125–131.

[10] Rafique, M. F., & Khan, M. N. A. (2022). Exploring static and live digital forensics: Methods, practices and tools. International Journal of Scientific & Engineering Research, 6(2), 91–107.

[11] Rastogi, V., Chen, Y., & Enck, W. (2013). AppsPlayground: Automatic security analysis of smartphone applications. Proceedings of the Third ACM Conference on Data and Application Security and Privacy, 209–220.

[12] Sindhu, K. K., & Meshram, B. B. (2021). Digital forensics and cyber crime datamining. Journal of Information Security, 3(3), 196–201.

[13] Willassen, S. Y. (2005). Forensics and the GSM mobile telephone system. International Journal of Digital Evidence, 2(1), 1–17.

[14] Xu, K., Li, Y., Deng, R. H., & Chen, K. (2019). DexRay: A deep learning approach to Android malware detection based on image representation of bytecode. International Workshop on Deployable Machine Learning for Security Defense, 110–133.

[15] Vidas, T., Zhang, C., & Christin, N. (2021). Toward a general collection methodology for Android devices. Digital Investigation, 8, S14–S24.

[16] Feichtner, J., & Gruber, S. (2020). Obfuscation-resilient code recognition in Android apps. European Workshop on System Security, 1–7.

[17] Aydos, M., Vural, Y., & Tekerek, A. (2019). Assessing risks and threats with layered approach to Internet of Things security. AIMS Electronics and Electrical Engineering, 3(3), 191–212.

[18] Bose, A., & Shin, K. G. (2006). On mobile viruses exploiting messaging and Bluetooth services. Second International Conference on Security and Privacy in Communication Networks, 1–10.