iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** V **Month of publication:** May 2026

DOI: https://doi.org/10.22214/ijraset.2026.81736

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue V May 2026- Available at www.ijraset.com

Android Malware Detection Using Machine Learning: A Hybrid Approach with Twitter-Based Hash Updates and Deep Learning Classification

M Tanmaya[1], K Sowmya[2], I Lavanaya[3]
Department of Computer Applications, Aditya University, Surampalem, Andhra Pradesh, India 533 437

Abstract: ***Smart phones play a pivotal role in human life, underscoring the critical importance of security and privacy. This is particularly true for the Android operating system, which dominates the smartphone market with a 70.97% share, making it a prime target for malware developers. This paper introduces a multilayer hybrid method for Android OS malware detection. Our approach uniquely integrates real-time data extraction from Twitter and machine learning techniques. The Twitter API is used to update a malware hash database (MD5, SHA-1, SHA-256) every 48 hours, capturing the latest malware signatures. Additionally, machine learning models — specifically Random Forest (RF) and Long Short-Term Memory (LSTM) — are used to analyze application permissions, achieving 88% accuracy in malware detection. A Java-based prototype anti-malware application demonstrates the practical effectiveness of the proposed system.***

Keywords: Android Malware Detection, Machine Learning, Random Forest, LSTM, Twitter API, Hash Database, Mobile Security, Permission Analysis.

I. INTRODUCTION

In recent times, smartphones have become indispensable in work, education, social interactions, and banking. However, their widespread use, especially on the Android OS, makes them susceptible to cyber threats including data leaks, phishing attacks, malware infections, and identity theft. Mobile attacks in Europe surged by 500% in the initial months of 2022 [1]. With Android holding a dominant market share of 70.97%, securing Android operating systems from malware is imperative.

Android banking trojans such as SharkBot, discovered in October 2021, highlight the severity of the threat. SharkBot targets banking apps and cryptocurrency exchanges using the Automatic Transfer System (ATS) approach to initiate fraudulent money transfers while bypassing multi-factor authentication [2].

This paper proposes a unified hybrid system comprising two complementary components: (1) hash-based malware detection with automated Twitter API updates, and (2) machine learning-based application permission classification using Random Forest and LSTM models. Together, these methods provide a comprehensive and adaptive defense against Android malware threats.

II. PROBLEM STATEMENT

The process of detecting Android malware presents several challenges that impact both the efficiency and accuracy of mobile security systems. One of the primary issues is the heavy reliance on static, manually maintained malware signature databases. These databases become outdated rapidly as new malware variants are released daily, leaving devices unprotected during the gap between update cycles.

Another significant problem is that most existing detection systems are single-layered, relying either on hash-based signature matching or on permission analysis alone. Hash-based systems fail entirely against zero-day and polymorphic malware that do not match any known signature. Permission-based systems, while more generalized, suffer from high false positive rates when applied without additional context, potentially flagging legitimate applications as malicious.

The increasing sophistication of Android malware further complicates detection. Techniques such as code obfuscation, dynamic code loading, and permission mimicry allow malware developers to bypass conventional detection mechanisms. Banking trojans such as SharkBot demonstrate that even multi-factor authentication can be circumvented by modern Android malware, highlighting the inadequacy of traditional defenses.

Additionally, most research-based malware detection models are not implemented as real-time, user-accessible applications. They remain confined to experimental environments with no practical deployment pathway. The lack of user-friendly interfaces further restricts usability, as non-technical users cannot interact with or benefit from these systems.

These challenges collectively highlight the need for an intelligent, adaptive, and deployable hybrid system capable of detecting both known and unknown Android malware in real time, while remaining accessible to end users without requiring specialized technical expertise.

III. GAP ANALYSIS

An extensive review of existing literature and systems reveals several critical gaps that limit the effectiveness of current Android malware detection solutions:

1) Single-Layer Detection: Most existing systems rely solely on either hash-based or permission-based detection, but not both. This means they either fail to detect novel malware (hash-only systems) or produce higher false positives (permission-only systems). No single-method system provides complete coverage against the full spectrum of Android malware threats.
2) Static Databases: The majority of hash-based detection systems use manually maintained malware signature databases. These databases are rarely updated in real time, leaving devices exposed to newly discovered malware between update cycles. The absence of an automated, continuous update pipeline is a significant operational weakness.
3) Absence of Real-Time Deployment: Many research works achieve high accuracy in controlled settings but are not extended into deployable, real-world applications. Models remain confined to experimental environments, limiting their practical utility for end users and security administrators.
4) Susceptibility to Adversarial Attacks: Current ML models, including Random Forest and LSTM, remain vulnerable to adversarial techniques such as code obfuscation, repackaging, and permission mimicry, where malware authors disguise malicious apps to resemble benign ones. Few systems incorporate adversarial robustness mechanisms.
5) Scalability and Performance: Dynamic analysis methods, while behaviorally rich, are computationally expensive and difficult to scale to large application repositories or real-time mobile environments. Static analysis alone is insufficient to capture runtime malware behaviors.
6) Lack of User-Friendly Interfaces: Most existing solutions require technical expertise to operate, limiting their accessibility to non-expert users. There is a clear need for intuitive interfaces that allow users to interact with detection systems without specialized knowledge.

These gaps collectively underscore the need for a hybrid, adaptive, and deployable system that combines automated database updates, ML-based classification, and a practical user interface — precisely what the proposed system addresses.

IV. LITERATURE SURVEY

The domain of Android malware detection is rapidly evolving. Researchers have employed static, dynamic, and hybrid analysis methodologies to improve the precision and efficiency of malware detection.

Chen et al. [3] introduced a graph attention network-based method using a class-set call graph (CSCG) to extract structural and semantic features, achieving accuracies between 97.28% and 99.54%. Alamro et al. [4] developed an Automated Android Malware Detection using Optimal Ensemble Learning (AAMD-OELAC) combining LS-SVM, KELM, and RRVFLN, tuned using the hunter-prey optimization (HPO) algorithm. Zhu et al. [5] proposed a multi-head squeeze-and-excitation residual network for static feature extraction, achieving 96.48% accuracy on a 3,187-sample dataset.

Mothanna et al. [6] demonstrated that RNN-based LSTM models achieved state-of-the-art accuracy of 98.58%, surpassing traditional algorithms.

XOmar et al. [7] used GRU architecture, achieving 98.2% accuracy in malware classification. Jaemin et al. [8] applied RF and LSTM on grayscale data-section images, reaching 98.02% accuracy while reducing data volume by 17.5%. Sun et al. [10] introduced a Frequency Differential Selection (FDS) algorithm, achieving 99% accuracy and 98% F1-score using the Omni Droid dataset.

Table I summarizes the key works reviewed in this survey:

TABLE I. Literature Survey Summary

S.No	Author & Work	Year	Method	Contribution	Results
1	Chen et al. [3]	2024	Graph Attention + ML	Class-set call graph; multi-modal	97.28–99.54% accuracy
2	Alamro et al. [4]	2023	Ensemble: LS-SVM, KELM, RRVFLN	HPO tuned; AAMD-OELAC	Superior accuracy & efficiency
3	Zhu et al. [5]	2023	Multi-head squeeze-and-excitation	Static: perms, API, hardware	96.48% accuracy
4	Mothanna et al. [6]	2021	RNN + RF + LSTM	State-of-the-art deep models	98.58% accuracy
5	Omar et al. [7]	2021	GRU + RF + LSTM	Comparative analysis	98.2% accuracy
6	Jaemin et al. [8]	2021	RF + LSTM on grayscale images	Reduced data by 17.5%	98.02% accuracy
7	Sun et al. [10]	2022	FDS + Weight Measurement	Omni Droid dataset	99% accuracy, 98% F1

V. SYSTEM ANALYSIS

A. *Existing Systems*

Current Android malware detection systems primarily use static, dynamic, and hybrid analysis. Static methods analyze code and metadata without execution; dynamic methods monitor runtime behavior; hybrid approaches combine both. While achieving high accuracy (up to 99%), these systems suffer from notable limitations:

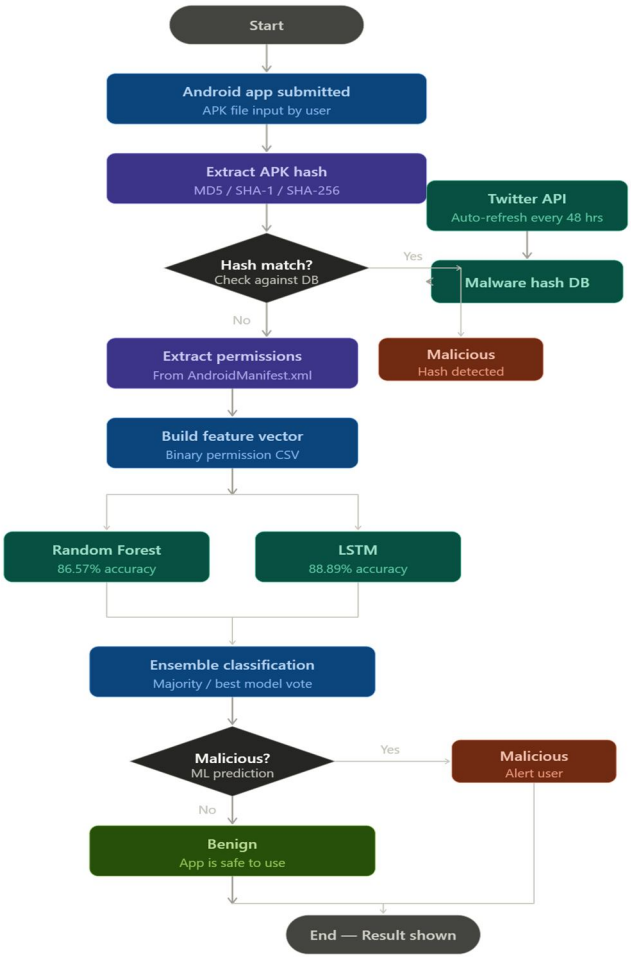
- High false positive rates from over-reliance on permission and API-call patterns.
- Scalability issues: dynamic analysis is resource-intensive at large scale.
- Susceptibility to adversarial attacks where malware obfuscates code to evade detection.
- Absence of automated database update mechanisms, making systems outdated against new threats.

B. *Proposed System*

The proposed system addresses these gaps through a dual-component hybrid architecture:

Component 1 — Hash-Based Detection with Automated Updates: Malware hashes (MD5, SHA-1, SHA-256) are compared against a Firebase database. The Twitter API automatically refreshes the database every 48 hours with the latest malware signatures extracted from security-related tweets.

Component 2 — ML-Based Permission Classification: Android application permissions extracted from AndroidManifest.xml files are analyzed using Random Forest and LSTM classifiers to identify malicious applications. A Java-based prototype Anti-Malware Application demonstrates the system's practical viability.



VI. SYSTEM DESIGN

A. *System Architecture*

The system architecture integrates two main modules: (1) a Twitter Hash Update mechanism and (2) a Machine Learning Analysis module. The Twitter API collects data, extracts malware hashes, and performs hash comparison against a Malware Hash Database. The ML module extracts permission features, processes them through Random Forest and LSTM models, and classifies applications as Malicious or Benign. Both modules feed into a unified Java-based Anti-Malware Application.

B. *UML Representations*

The system is modeled using Unified Modeling Language (UML) diagrams including Use Case, Sequence, Activity, Class, Deployment, and Flowchart diagrams. The Use Case diagram captures interactions between User, Admin, and the Android Malware Prediction subsystem. The Sequence diagram illustrates the message flow across Login, Register, Model Training, Prediction, and Data Visualization stages. The Deployment diagram shows a three-tier architecture: Database Server (MySQL), Web Server (Django), and User PC connected via port 8000.

VII. IMPLEMENTATION

A. *Dataset*

The models were trained on a dataset of 837 samples: 288 benign Android applications sourced from Google Play and 549 malicious samples from the CICInvesAndMal2019 dataset, GitHub repositories, and VirusShare. This dataset covers a diverse range of malware families to ensure robust model generalization.

B. Feature Extraction

Android APK files were reverse-engineered using apktool to access AndroidManifest.xml files. A custom Python script extracted permission features (e.g., INTERNET, CAMERA, ACCESS_FINE_LOCATION, READ_CONTACTS) and compiled them into a structured CSV file for binary classification. Permissions represent a critical security signal, as they define what resources an application can access.

C. Model Training

Two machine learning models were developed and evaluated:

Random Forest: An ensemble of decision trees trained on the extracted permission features. The model is robust to overfitting and capable of handling high-dimensional feature spaces. It uses majority voting across 100 estimators for final classification.

Long Short-Term Memory (LSTM): A recurrent deep learning model that captures sequential dependencies in permission patterns. The model uses gated memory cells to retain relevant information and is particularly effective at distinguishing complex malware permission patterns from benign ones.

Both models were implemented using Python 3.10, scikit-learn, and TensorFlow/Keras, with an 80:20 train-test split (random_state=42).

D. System Requirements

Hardware: Intel i3 5th Gen processor, 4 GB RAM, 500 GB storage. Software: Windows 10/11, Python 3.10, Visual Studio Code, Django framework, MySQL with WAMP/XAMPP. Front-end: HTML5, CSS3, JavaScript.

VIII. RESULTS AND DISCUSSION

Table II presents the performance metrics for both models on the test dataset. The LSTM model outperforms Random Forest across all metrics, demonstrating superior ability to capture sequential patterns in Android application permissions.

TABLE II. Model Performance Comparison

Metric	Random Forest	LSTM	Best Model
Accuracy	86.57%	88.89%	LSTM
Precision	67.94%	69.20%	LSTM
Recall	92.77%	94.58%	LSTM
F1 Score	90.06%	91.01%	LSTM

The LSTM model achieves an overall accuracy of 88.89%, with a high recall of 94.58%, indicating that the system rarely misses actual malware instances — a critical requirement for security applications. The Random Forest model achieves 86.57% accuracy with a strong F1-score of 90.06%, making it a computationally lighter alternative suitable for resource-constrained environments.

The hash-based component with 48-hour Twitter API updates ensures that the system remains current with emerging malware signatures, reducing reliance on static databases. The combined hybrid system achieves lower false negative rates compared to either component in isolation, validating the advantages of the multi-layer approach.

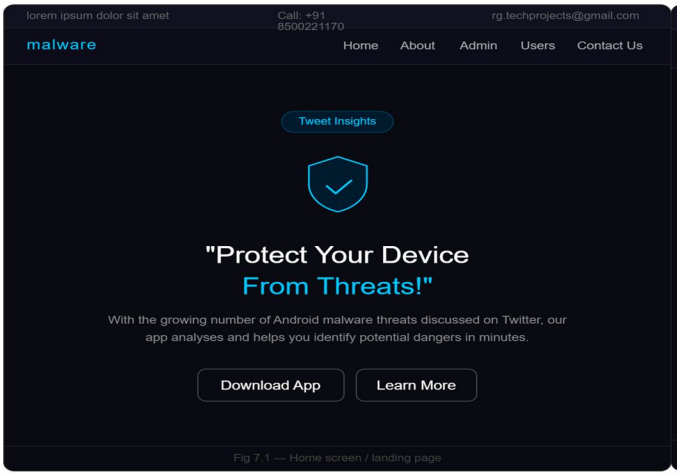


Fig 7.1 — Home screen / landing page

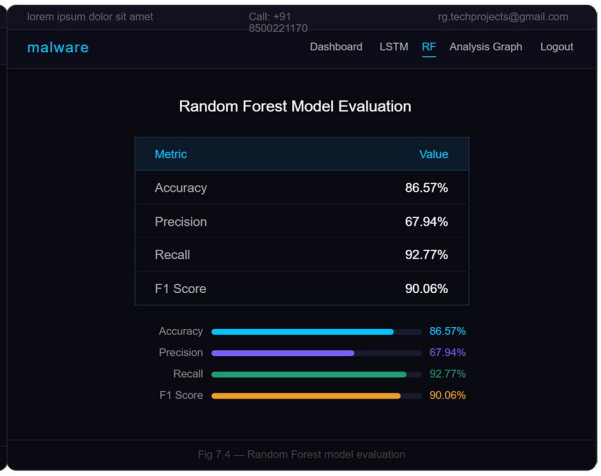


Fig 7.4 — Random Forest model evaluation

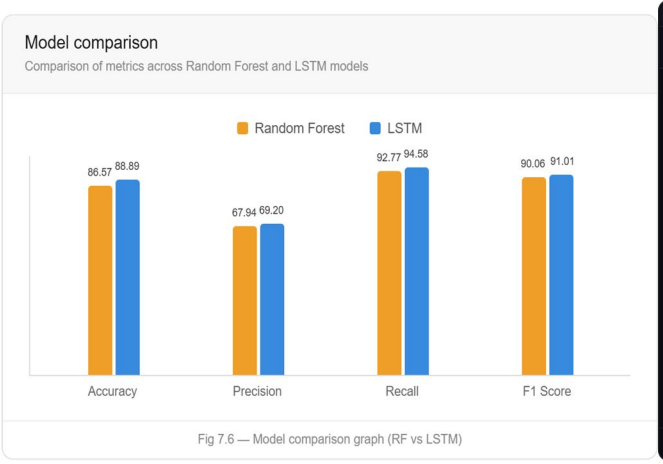


Fig 7.6 — Model comparison graph (RF vs LSTM)

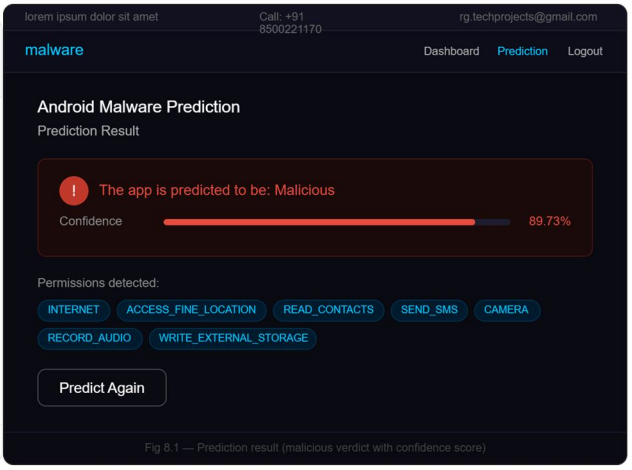


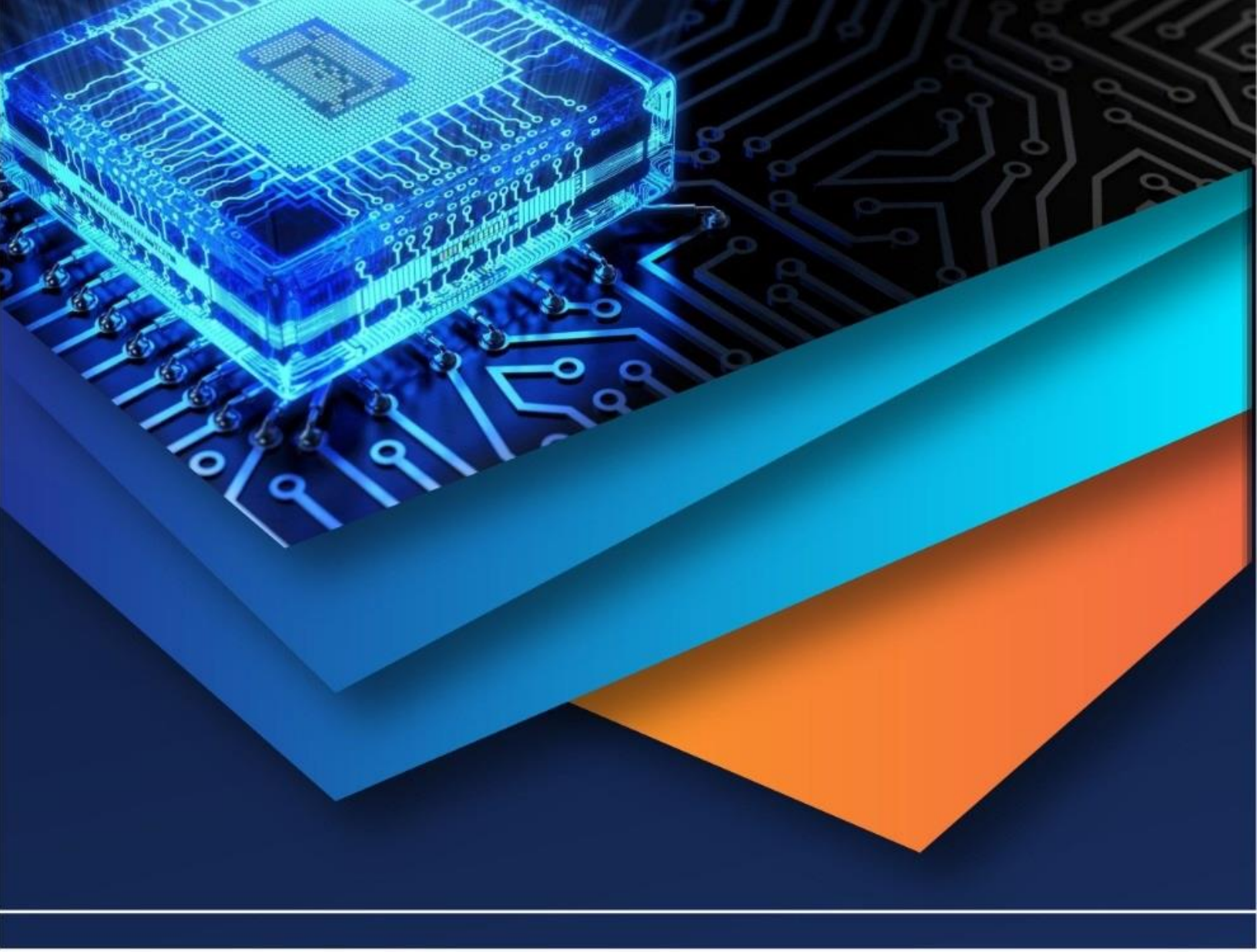
Fig 8.1 — Prediction result (malicious verdict with confidence score)

IX. CONCLUSION AND FUTURE WORK

This paper presented a novel hybrid Android malware detection system combining automated hash-database updates via the Twitter API with machine learning-based permission classification using Random Forest and LSTM models. Trained on 837 samples (288 benign, 549 malicious), the system achieved 88% classification accuracy. The dual-component architecture reduces both false positives and false negatives while remaining adaptive to the evolving malware landscape. Future work will explore: (1) integration of dynamic behavioral features (API call sequences, network traffic patterns) alongside static permission analysis; (2) adversarial robustness training to counter obfuscated malware; (3) expansion of the Twitter-based update pipeline to include other threat intelligence sources such as VirusTotal and MISP feeds; and (4) deployment on cloud platforms for scalable, real-time protection at the enterprise level.

REFERENCES

[1] A. Mcneil and W. S. Jones, "Mobile Malware Surging in Europe: A Look at the Biggest Threats," Proofpoint, 2022. [Online]. Available: https://www.proofpoint.com
[2] CYBLE, "New SharkBot Variant Discovered," 2022. [Online]. Available: https://blog.cyble.com
[3] J. Chen et al., "Android Malware Detection Method Based on Graph Attention Networks and Deep Fusion of Multimodal Features," Expert Systems with Applications, 2024, Elsevier.
[4] H. Alamro et al., "Android Malware Detection Using Optimal Ensemble Learning Approach for Cybersecurity," IEEE Access, 2023.
[5] H. Zhu et al., "Android Malware Detection Based on Multi-head Squeeze-and-Excitation Residual," Expert Systems with Applications, vol. 212, pp. 118705, 2023.
[6] A. Mothanna et al., "Machine Learning Models for Android Malware Detection," Procedia Computer Science, vol. 184, pp. 841–846, 2021.
[7] O. N. Elayan and A. M. Mustafa, "Android Malware Detection Using Deep Learning," Procedia Computer Science, vol. 184, pp. 847–852, 2021.
[8] J. Jung et al., "Efficient Malware Detection Using Grayscale Image Representation," pp. 153, 2021.
[9] A. S. Shatnawi et al., "Android Malware Detection Using Static and Dynamic Analysis," Wireless Communications and Mobile Computing, Hindawi, 2022.
[10] J. Sun et al., "Frequency Differential Selection Algorithm with Weight Measurement for Android Malware Detection," 2022.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY