# Android Malware Detection Using Multi-Domain Feature Analysis And Deep Learning Models

**Sharon R[1],Janapriya S[2], Nishmitha R[3], Prof.Mrs.Ramya R[4]**
[1, 2, 3, 4] Dept of Artificial Intelligence and Data Science
[1, 2, 3, 4] Misrimal Navajee Munoth Jain Engineering College, Chennai, Tamilnadu – 600097.

***Abstract-*** *We propose a hybrid and intelligent Android malware detection framework that integrates multi-domain feature extraction with deep learning models to improve mobile application security. The system consists of two key modules: the APK Security Analyzer and the App Behavior Analyzer. The APK Security Analyzer applies both static and dynamic analysis using tools such as Androguard, Android Emulator, ADB, and Monkey to extract permissions, API call sequences, and network traffic features. These are processed using a tri-model architecture—Multi-Layer Perceptron (MLP), Long Short-Term Memory (LSTM), and LightGBM—within a Streamlitinterface.The App Behavior Analyzer enhances static CSV-based analysis by automating preprocessing with StandardScaler and applying a sliding window for time-series classification using an LSTM model. It supports real-time predictions and provides interactive visualizations to aid user understanding of results.Experimental results show high accuracy: MLP and LSTM models each achieved 96%, while LightGBM reached 88%. Precision, recall, and F1-scores confirm system robustness, with MLP scoring 96%, 93%, and 94%, respectively. Visualizations such as radar charts and progress bars clearly communicate app risk levels and behaviorpatterns.These findings establish the framework as a robust, scalable, and interpretable solution for real-time Android malware detection, advancing mobile cybersecurity tools.*

***Keywords***- Android malware detection, machine learning, static and dynamic analysis, Deep learning, behavioral analysis.

## I. INTRODUCTION

The rapid adoption of Android devices has revolutionized mobile computing, yet this growth has also led to an alarming increase in Android malware threats. Malicious applications can compromise user privacy, steal sensitive data, and disrupt device operations, emphasizing the need for effective and scalable malware detection methods.

Traditional malware detection techniques, such as signature-based and heuristic approaches, have shown limitations in handling the evolving landscape of Android threats. Signature-based methods are reactive and fail to identify new or polymorphic malware variants, while heuristic approaches often generate high false positive rates due to their reliance on predefined rules. Furthermore, these methods usually analyze limited feature sets, restricting their ability to capture complex application behaviors comprehensively.

To address these challenges, this research proposes a hybrid Android malware detection framework leveraging multi-domain feature extraction and advanced machine learning models. By analyzing permissions, API call sequences, and network traffic data, the system constructs a comprehensive behavioral profile of each app. The framework employs specialized models—a Multi-Layer Perceptron (MLP) for permissions, Long Short-

Term Memory (LSTM) networks for API calls, and LightGBM for network traffic—to accurately classify applications as benign or malicious.Additionally, the system includes an App Behavior Analyzer module that was initially designed for static review of CSV data reflecting application behavior patterns. While the original design employed an LSTM model to classify apps based on this data, it lacked dynamic data interpretation, meaningful result explanations, and intuitive visualization tools. Addressing these shortcomings, the module was enhanced to provide a more interactive, insightful, and user-friendly analysis framework, significantly improving malware classification reliability and user experience.

## II. EXISTING SYSTEM

Android malware detection systems mainly rely on traditional static techniques like signature-based and heuristic analysis. These methods compare applications against known malware patterns or predefined rules, which makes them effective for detecting existing threats but ineffective against new, unknown, or polymorphic malware.

Heuristic analysis flags suspicious behavior such as unusual permission requests but often results in high false positives.Some approaches employ machine learning models—like Decision Trees, SVMs, and Random Forests—based on limited features such as permissions and API calls. While these improve classification, outdated datasets and narrow feature sets hinder their ability to adapt to evolving threats. Deep learning models like CNNs and RNNs offer better pattern recognition but still struggle without real-time behavioral insight.Hybrid systems have emerged, combining static and dynamic analysis for improved detection, using runtime monitoring to identify malicious patterns. Despite this, limitations persist in scalability, automation, and real-time performance. Manual feature engineering and delayed detection reduce effectiveness in fast-changing threat environments.

Overall, these systems face challenges such as limited detection for unknown malware, false positives, lack of real-time analysis, and poor adaptability—highlighting the need for more dynamic, scalable, and automated detection frameworks.

## III. LITERATURESURVEY

Mohammed Maray et al. (2024) [1] introduced the Intelligent Pattern Recognition using Equilibrium Optimizer with Deep Learning (IPR-EODL) for Android malware detection. Their approach combines data preprocessing, a Channel Attention LSTM model, and hyperparameter tuning via the Equilibrium Optimization algorithm. Evaluated on standard datasets, the model achieved a high accuracy of 99.18%, demonstrating the effectiveness of deep learning techniques in improving detection accuracy against evolving malware threats.

Omar N. Elayan et al. (2021) [2] proposed a deep learning-based detection method using Gated Recurrent Units (GRU) with static feature extraction from API calls and permissions. Compared to traditional classifiers such as SVM, KNN, and Random Forest, the GRU model outperformed all, reaching 98.2% accuracy on the CICAndMal2017 dataset. This work highlights the importance of deep learning and careful feature selection for robust Android malware identification.

Zhenlong Yuan et al. (2016) [3] developed DroidDetector, integrating static and dynamic analysis with a Deep Belief Network (DBN) to classify Android malware. Using a dataset of over 20,000 apps, their method combined permissions and API calls with runtime behaviors captured via DroidBox. The DBN achieved a detection accuracy of 96.76%, outperforming classical machine learning methods, and underscored the value of multi-source feature fusion for effective malware classification.

MuhammadUmar Rashid et al. (2025) [4] presented a hybrid deep learning framework analyzing permissions, intents, and API calls through Deep Neural Networks. Evaluated on a large dataset spanning 15 malware families, the model improved detection accuracy to 98.2%, surpassing previous state-of-the-art methods. Their work emphasizes the significance of multi-dimensional feature analysis and model interpretability in addressing scalability and obfuscation challenges in malware detection.

Meghna Dhalaria et al. (2020) [5] proposed a hybrid detection system combining static and dynamic analyses for malware classification. They introduced two public datasets for binary and multiclass tasks and utilized various machine learning algorithms with feature selection for improved performance. Their hybrid approach achieved 98.53% accuracy for binary classification and 90.10% for multiclass, demonstrating the benefits of integrating behavioral and structural app features for enhanced malware detection.

## IV. PROPOSEDSYSTEM

The proposed system is a hybrid Android malware detection framework designed to overcome the liitations of conventional methods by integrating multi-domain feature extraction with advanced machine learning techniques. It harnesses the power of deep learning and machine learning algorithms to enhance detection accuracy, efficiency, and scalability in identifying malicious applications.

Initially, the system accepts input files in two formats: APK files or CSV behavioral logs. For APK files, the system performs feature extraction across four critical domains: permissions (using the Drebin dataset), API call sequences (extracted via Cuckoo Sandbox), network traffic (analyzed with CIC-InvesAndMal2019 dataset), and activity logs. If CSV files containing app behavior logs are uploaded, the system bypasses static feature extraction and directly applies sequential analysis to detect anomalies.

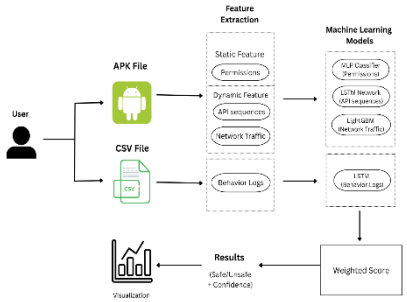


**Figure1:SystemArchitecture**

The extracted features are processed using specialized models tailored for each domain. Permissions are analyzed using a Multi-Layer Perceptron (MLP), API sequences are evaluated through Long Short-Term Memory (LSTM) networks, and network traffic is classified using LightGBM. This modular approach ensures high precision in detecting various aspects of malicious behavior.

To complement static analysis, the system incorporates real-time monitoring capabilities. During application execution, dynamic behavioral analysis, anomaly detection, and context-aware detection techniques are applied to capture malicious activities as they unfold, thereby offering comprehensive malware evaluation.

Users upload APK or CSV files, after which the system performs automated feature extraction, model training, and classification. The results are presented in a user-friendly format, empowering users with actionable insights to make informed decisions regarding app safety.

Overall, this proposed system delivers a robust, automated, and scalable solution that protects Android devices from sophisticated malware threats while enhancing user security and trust in mobile applications.

## V. IMPLEMENTATION

This project comprises two main modules that collectively form an integrated system for Android malware detection:

**APK Security Analyzer Module**
**App Behavior Analyzer Module**

The entire system is developed using Python 3.9 and deployed through the Streamlit framework, which provides a lightweight and interactive web interface for real-time analysis. The system architecture incorporates both static and dynamic analysis techniques, driven by multiple specialized machine learning models to ensure comprehensive risk evaluation.

### 1. APK Security Analyzer Module

This module serves as the foundation for static and simulated dynamic analysis of APK files. It is engineered to extract critical feature sets from uploaded applications and predict risk levels using a hybrid of models trained on diverse data domains.

### 1.1 Input Processing

Upon receiving an APK file through the file upload interface, the system utilizes the Androguard library to extract basic metadata such as the application name, version, and package ID.

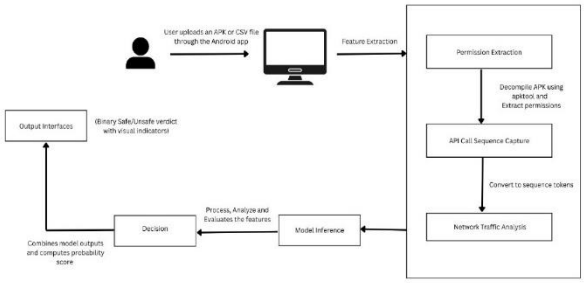


**Figure2:Flow Diagram**

Following this, it initiates multi-domain feature extraction:

**Permissions** are parsed from the manifest and vectorized using a predefined schema.
**API call sequences** are identified through static analysis and transformed into token sequences.
**Network traffic behavior** is simulated using mock data in the absence of actual PCAP files.

These extracted features are subsequently converted into structured vectors, optimized for model consumption.

### 1.2 Initial Implementation

Initially, the module displayed individual model outputs from the MLP, LSTM, and LightGBM models directly on the interface. While technically accurate, this raw output format overwhelmed users with excessive data such as feature vectors and model confidence scores. Risk levels were calculated via a weighted average but presented with minimal explanation.

### 1.3 Challenges Identified

The early interface suffered from poor visualization

and interpretability. Misaligned charts, verbose probability details, and redundant technical information rendered the system less intuitive. Users found it difficult to interpret results due to inconsistent visual cues and the absence of meaningful summaries.

**1.4 Optimized Solution**

A complete redesign was undertaken to enhance user experience. This included:
A **radar chart** to compare predictions from each model.
A **horizontal bar graph** showing the influence of each model in the risk calculation (Permissions: 0.6, API: 0.2, Traffic: 0.2).
A **permissions bar chart** highlighting suspicious permissions in red.
A **pastel-colored progress bar** to intuitively indicate overall risk level (Low, Medium, High).
Concise, aligned text explanations adjacent to each visualization.
These enhancements preserved the original logic while vastly improving user comprehension and visual appeal.

**1.5 Visualization Features**

The Visualization Module integrates multiple elements into a unified dashboard:

Circular radar graphs for model comparison
Linear weight distribution bars
Permission charts with color-coded risk indicators

A progress bar summarizing the application's final risk status
All charts are uniformly styled and vertically aligned, ensuring a clean, cohesive layout that supports rapid risk assessment.

**1.6 Module Outcome**

The final APK Security Analyzer provides users with an intuitive, real-time risk assessment interface. Upon analysis, it delivers a structured output containing metadata, risk scores, and visual analytics. Suspicious permissions are flagged clearly, and users are offered a downloadable JSON report for offline review or compliance needs.

**2. App Behavior Analyzer Module**

The App Behavior Analyzer is responsible for analyzingbehavioral patterns from CSV logs, focusing on dynamic data obtained from runtime execution or monitoring tools.

**2.1 Input Processing**

Initially, users had to manually preprocess CSV files before uploading. The uploaded dataset was then analyzed using an LSTM model designed to classify app behavior as benign or malicious. However, this static pipeline lacked intuitive visualization and dynamic interpretability.

**2.2 Initial Design**

The first iteration involved simple CSV ingestion and LSTM-based prediction. It offered minimal interaction, no dynamic visualization, and required users to format data precisely, increasing the chance of input errors.

**2.3 Issues Encountered**

Several critical shortcomings were identified:
Lack of user-friendly interface
Manual preprocessing requirements
Frequent errors due to improper CSV formatting
Inconsistent prediction reliability due to input sensitivity

**2.4 Optimized Solution**

To resolve these limitations, the following enhancements were introduced:

**Automated preprocessing** using a pre-trained StandardScaler.pkl to normalize input features.
A **sliding window mechanism** to convert flat data into structured sequences suitable for the LSTM Model.
A **Streamlit-based interactive interface** for file uploads, visual feedback, and real-time behavior trend visualization.
These changes significantly improved model reliability, interpretability, and user experience.

**2.5 Module Outcome**

The improved App Behavior Analyzer now offers seamless, accurate classification of behavioral data. It generates real-time predictions, supports structured sequence analysis, and displays results using clean, interpretable charts. This makes it highly effective for analysts and developers evaluating app behavior patterns for security anomalies.

**VI. RESULTANALYSIS**

To assess the effectiveness of the proposed Android malware detection system, various machine learning models were evaluated using core classification metrics: accuracy, precision, recall, and F1-score. These metrics provide a

comprehensive understanding of each model's predictive capability and help identify strengths and limitations.

Firstly, precision was calculated to determine the proportion of correctly identified malware instances among all predicted malware. High precision signifies a lower rate of false positives, which is crucial in security-sensitive applications where incorrect classification could trigger unnecessary warnings. Similarly, recall was used to evaluate the ability of the model to detect all actual malware samples in the dataset. A high recall score indicates the model successfully captured most of the malicious activities. Accuracy, the most intuitive metric, measured the overall correctness of predictions. However, in imbalanced datasets like malware detection, where benign apps may outnumber malicious ones, accuracy alone may not provide sufficient insight. Therefore, the F1-score, the harmonic mean of precision and recall, was also considered to balance both metrics and offer a clearer picture of performance under unequal class distributions.

| Model | Accuracy (%) | Precision (%) | Recall (%) | F1-Score (%) |
|---|---|---|---|---|
| Traditional Model (J48) | 69.10 | 68.0 | 70.0 | 69.0 |
| Traditional Model (Random Forest) | 97.80 | 97.5 | 97.0 | 97.2 |
| Traditional Model (Decision Tree) | 94.60 | 94.0 | 95.0 | 94.5 |
| Multi-Layer Perceptron (MLP) | 96 | 96 | 93 | 95 |
| Long Short-Term Memory (LSTM) | 99 | 92 | 88 | 90 |
| Light GBM | 88 | 88 | 87 | 87 |

**Table1: Model Comparison**

The proposed models included Multi-Layer Perceptron (MLP), Long Short-Term Memory (LSTM), and LightGBM. Among these, the LSTM model exhibited the highest accuracy at 99.0%, showcasing its superior learning ability over time-sequenced data such as API call patterns. The MLP model demonstrated the highest precision and recall scores at 96.0% and 93.0%, respectively, resulting in a strong F1-score of 95.0%, indicating its balanced performance. LightGBM, while efficient and faster in execution, achieved comparatively lower metrics across the board, with an F1-score of 87.0%, suggesting that deeper learning-based models are more effective in this domain.

In addition to evaluating advanced models, a comparison was made with traditional classifiers such as J48, Random Forest, and Decision Tree. Random Forest outperformed other traditional approaches with a high accuracy of 97.8% and an F1-score of 97.2%, demonstrating its robustness and reliability. The Decision Tree also performed well with an accuracy of 94.6%. On the other hand, the J48 model had the lowest overall performance, achieving an accuracy of 69.1% and an F1-score of 69.0%. When comparing these with the proposed deep learning models, it became evident that MLP and LSTM delivered superior results, surpassing most traditional classifiers in every evaluation aspect. These findings affirm the advantages of adopting advanced deep learning techniques for effective and reliable Android malware detection.

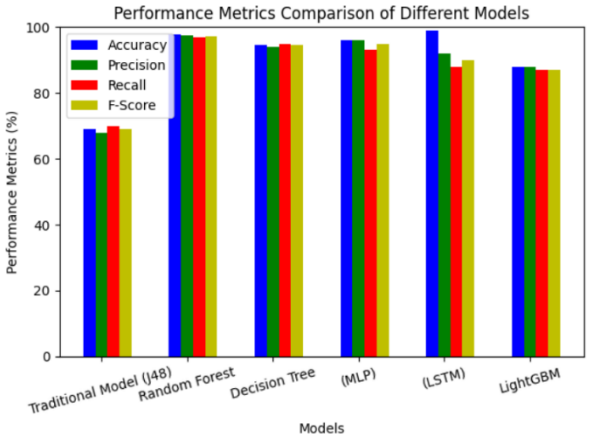


**Figure3:Graph Analysis**

## VII. CONCLUSIONAND FUTURE ENCHANCEMENT

### A. CONCLUSION

The proposed Hybrid Android Malware Detection Framework presents a comprehensive and flexible solution to counter the growing sophistication of mobile threats. By enabling the upload of both APK files and CSV files containing app behavior logs, the system supports multi-format inputs for broader analysis. It leverages a combination of advanced machine learning models, including Multi-Layer Perceptron (MLP), Long Short-Term Memory (LSTM), and LightGBM, to detect malicious applications with high precision and accuracy. The use of these models ensures that the framework not only identifies threats effectively but also reduces false positives, enhancing user trust and reliability.

The system's real-time monitoring feature further strengthens its detection capability by dynamically tracking malicious behaviors during app execution. The integration of behavioral analysis, anomaly detection, and context-aware intelligence adds multiple layers of scrutiny, significantly improving detection rates. Moreover, the user-friendly interface simplifies complex results into actionable insights,

empowering users to take prompt decisions regarding app safety. The proposed framework meets its objective of enhancing Android device security, while its modular architecture allows for easy scalability and adaptability to future advancements, making it a robust and forward-looking solution for mobile security.

B. FUTUREENHANCEMENT

Future enhancements aim to broaden the system's applicability, improve performance, and increase its responsiveness to evolving malware techniques. One major direction is multi-platform support, where the framework can be extended to detect threats not only on Android but also across web browsers and other mobile operating systems such as iOS and HarmonyOS. This cross-platform intelligence will allow the creation of a unified threat detection ecosystem.

To enhance real-time protection, future iterations may incorporate continuous user behavioral monitoring and automated threat mitigation features, such as isolating suspicious apps or blocking malicious actions without manual intervention. The development of a lightweight SDK will also enable third-party applications to embed this malware detection capability directly into their platforms, while push notification features can alert users immediately when suspicious activities are detected.

Cloud-based enhancements will include distributed analysis using services like AWS or Google Cloud to handle large-scale datasets, and a collaborative threat intelligence network where users can contribute and share malware samples for collective learning. Machine learning improvements will focus on using transfer learning to adapt rapidly to new malware families with minimal retraining, along with explainable AI features to help users understand the basis of detection results.

Furthermore, integrating real-time malware feeds from global sources such as VirusTotal or MITRE ATT&CK can keep the system updated against emerging threats. Automated malware reporting mechanisms can contribute to building a global database of new threats. Finally, advanced threat simulation tools and a secure sandbox environment will allow safe testing of unknown apps and help train detection models under realistic attack scenarios. These enhancements will not only future-proof the framework but also provide a more proactive, scalable, and user-friendly security solution.

## REFERENCES

[1] Intelligent Pattern Recognition Using Equilibrium Optimizer With Deep Learning Model for Android Malware Detection, MOHAMMED MARAY, MASHAEL MAASHI, HAYA MESFER ALSHAHRANI, SUMAYH S. ALJAMEEL, SITELBANAT ABDELBAGI, AND AHMED S. SALAMA — IEEE, 2024

[2] Android Malware Detection Using Deep Learning, Omar N. Elayan, Ahmad M. Mustafa — Science Direct, March 23–26, 2021

[3] DroidDetector:Android Malware Characterization and Detection Using Deep Learning, Zhenlong Yuan, Yongqiang Lu, and Yibo Xue — IEEE, February 2016

[4] Hybrid Android Malware Detection and Classification Using Deep Neural Networks, Muhammad Umar Rashid, Shahnawaz Qureshi, Abdullah Abid, Saad Said Alqahtany, Ali Alqazzaz, Mahmood ul Hassan, Mana Saleh Al Reshan, Asadullah Shaikh — Springer, 25 February 2025

[5] A Hybrid Approach for Android Malware Detection and Family Classification, Meghna Dhalaria, Ekta Gandotra —IJIMAI, 1 September 2020