Journal of Computer and Electrical Sciences (JCES)- Volume-1 Issue-3, March 2026
ISSN: 3049-2602

AI-BASED APK Malware Detection Engine

N. Jahnavi T. Santhi Vardhan M. Sai teja T. Sasankar reddy P. Ram Kumar
namburijahnavi447@gmail.com , vardhantanniru31@gmail.com , sai674288@gmail.com , sesank143@gmail.com , ram1kumar8079@gmail.com
Department of Computer Science and Engineering
Ligaya's Institute of Management and Technology
Associate Professor B. Pavan Akhil , akhil.lingayas@limat.edu.in
Department of Computer Science and Engineering
Ligaya's Institute of Management and Technology, Vijayawada, Nunna Road, MadhalaVari Gudem(521212)

Abstract: The rapid expansion of the Android ecosystem has significantly increased malware distribution through APK files, exploiting the platform's open-source nature. Traditional signature-based detection methods fail to identify obfuscated or newly generated malware variants. This paper proposes an AI-based APK Malware Detection Engine that applies machine learning algorithms to detect malicious Android applications. The system performs static analysis by extracting application permission features and transforming them into a structured dataset. To enhance detection performance and reduce irrelevant data, a Genetic Algorithm (GA) is utilized for efficient feature selection, identifying the most significant permission attributes related to malware behavior. The selected features train two classification models: an Artificial Neural Network (ANN) and a Support Vector Classifier (SVC). Experimental evaluation demonstrates that the proposed system achieves high detection accuracy, with ANN reaching 94.8% and SVC achieving 93.2%, significantly outperforming existing baselines. The system provides an **intelligent,** scalable approach for Android malware detection, strengthening mobile security**.**

1. INTRODUCTION

Android smartphones are used by billions of people worldwide for communication, banking, social media, and other critical activities. Because of this widespread use, Android devices have become a primary target for cyber attackers. Many attackers create malicious applications that appear similar to normal apps and distribute them through third-party platforms or unknown sources. According to Senanayake et al. (2021), the popularity of Android makes it one of the main targets for malware attacks. These malicious applications can steal personal data, access contacts, send unauthorized messages, and cause serious privacy and financial risks (Kshirsagar & Agrawal, 2022). Therefore, there is a strong need to develop intelligent systems that can automatically analyze Android applications and detect malware before installation. This project focuses on building a machine learning-based system that analyzes APK (Android Package Kit) files, extracts permission-based features, and detects malicious behavior using advanced classification techniques.

2. LITERATURE SURVEY

Several researchers have studied Android malware detection using different machine learning techniques. Senanayake et al. (2021) provided a systematic review highlighting that feature extraction from APK files plays a key role in identifying malicious behavior. Kshirsagar and Agrawal (2022) analyzed different feature selection methods using datasets such as AndroZoo, Drebin, and CIC-MalDroid2020, showing that selecting relevant features significantly improves detection accuracy. Aurangzeb and Aleem (2023) focused on detecting obfuscated Android malware using deep learning, demonstrating that attackers modify malware code to avoid signature-based systems. Pathak et al. (2024) proposed a static analysis framework using permission-based datasets, showing effective detection without executing the application. Alhussen (2024) introduced a deep learning-based approach using neural networks to learn complex patterns.
From the reviewed literature, the following key observations are made:

- Traditional signature-based detection fails against obfuscated and new malware variants.
- Permission-based static analysis is effective but requires intelligent feature selection.
- Genetic Algorithms can optimize feature subsets, reducing computational overhead.
- Hybrid models (ANN and SVC) provide better accuracy than single-classifier systems.

- Existing systems often lack real-time APK analysis and user-friendly interfaces.
-

3. PROPOSED SYSTEM

3.1 Overview:

The proposed system is an AI-based APK Malware Detection Engine designed to classify Android applications as benign or malicious using static permission analysis. The system uses a Genetic Algorithm (GA) for optimal feature selection and two machine learning models—Artificial Neural Network (ANN) and Support Vector Classifier (SVC)—for classification. A Flask-based web interface allows users to upload APK files, select a model, and receive real-time detection results with confidence scores and metadata.

3.2 Problem Addressed:

The number of Android applications is increasing rapidly, and many malicious applications are distributed through APK files. Existing malware detection systems mainly rely on signature-based or rule-based techniques, which cannot effectively detect new or unknown malware variants. Attackers also use code obfuscation to hide malicious code, making detection more difficult. There is a need for intelligent malware detection systems that can analyze APK files and automatically detect suspicious behavior without executing the application.

3.3 System Architecture:

The system architecture follows a pipeline-based model consisting of four layers:

1. **Presentation Layer:** Flask web interface for APK upload and model selection.
2. **Business Logic Layer:** Androguard for permission extraction, GA for feature reduction.
3. **Machine Learning Layer:** Pre-trained ANN and SVC models for classification.
4. **Data Layer:** CSV dataset (1,700 samples × 400 permissions) and serialized model files (.pkl).

When a user uploads an APK file, the system: (1) extracts declared permissions using Androguard; (2) encodes them as a binary feature vector against a 400-feature schema; (3) applies the pre-trained Genetic Algorithm selector to reduce the feature set by approximately 77%; and (4) passes the reduced features to the selected ANN or SVC model for classification. The result—malware or benign—is returned with a confidence score and APK metadata (app name, SDK version, file size).

4. METHODOLOGY

The methodology is organized into the following steps:

Step 1: Dataset Preparation

- Android applications are collected from two sets: malicious and benign families.
- Permission information is translated into binary format where "1" indicates permission requested and "0" indicates not requested.
- The dataset contains up to 400 features and 1,700 distinct samples.

Step 2: Genetic Algorithm (GA) for Feature Selection

- GA starts with a random population of candidate feature subsets.
- Each subset is evaluated based on classification accuracy (fitness) using 5-fold cross-validation.
- The best subsets are selected for reproduction through crossover and mutation.
- The process repeats for multiple generations until an optimal feature subset (22–50 permissions) is identified

Step 3: ANN Model Training

- A Sequential Keras model is constructed with Dense layers (256 → 128 → 32 → 1 neurons).
- Dropout regularization is applied to prevent overfitting.
- The model is compiled with SGD optimizer and binary cross-entropy loss.
- Training is performed for 175 epochs with a batch size of 32.

Step 4: SVC Model Training with Hyperparameter Tuning

- GridSearchCV from scikit-learn finds optimal hyperparameters (C, kernel, gamma).
- The GA-selected feature subset is applied to the training data.
- Cross-validation with 5 folds evaluates model performance.
- The best SVC estimator is identified and serialized.

Step 5: APK Permission Extraction and Inference

- User uploads APK via Flask interface.

- Androguard's APK class reads the binary file and extracts declared permissions.
- A binary vector (400 features) is created and reduced using the GA selector.
- The reduced vector is passed to the selected model (ANN or SVC) for classification.
- Result, confidence score, and metadata are displayed to the user.

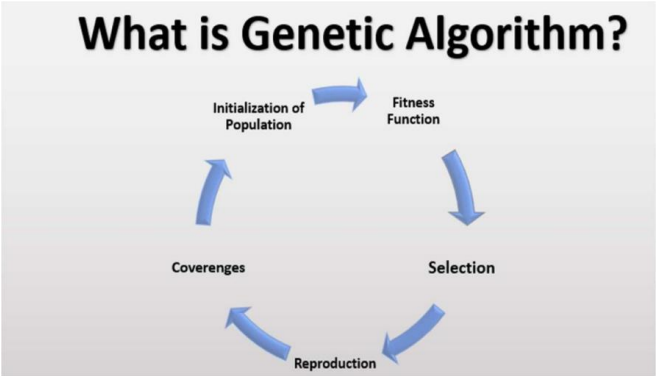


Figure 3.4.1.1: Genetic Algorithm cycle progress

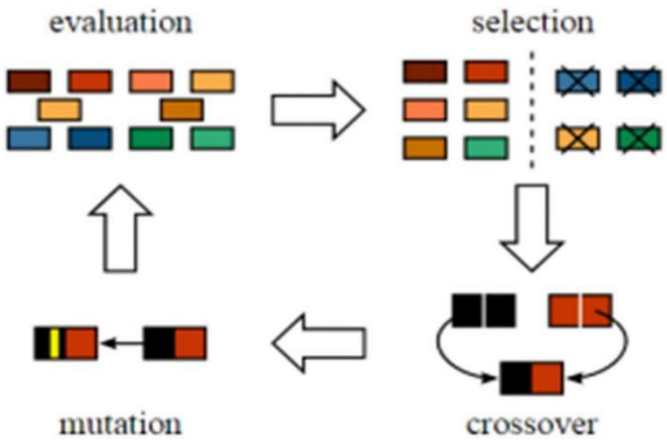


Figure 3.4.1.2: Selection, Crossover, Mutation operation in GA

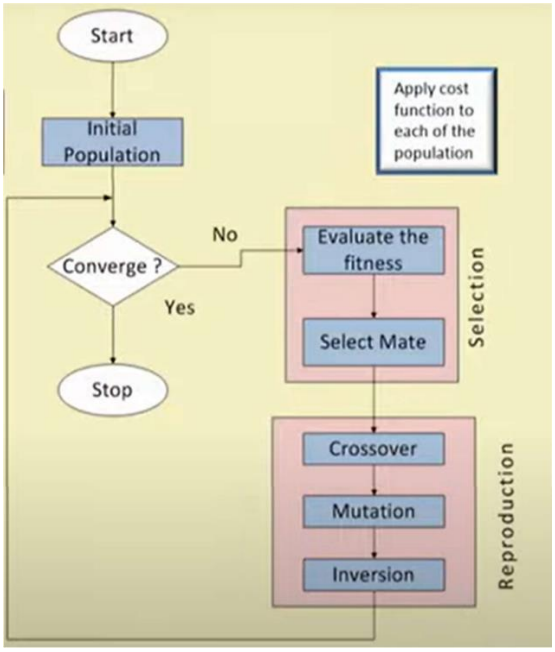


Figure 3.4.1.3: Flow chart for GA.

Serial No.	Algorithm	Accuracy
1.	Neural Network (ANN)	94.8%
2.	Support Vector Classifier (SVC)	93.2%

Table 4.10.1: Comparing Accuracy of Algorithms

Metric	ANN Result	SVC Result	Benchmark (PerDRaML)	Status
Accuracy	94.8%	93.2%	89.96% (RF)	PASS
Precision	93.5%	92.1%	~89–90%	PASS
Recall (Sensitivity)	95.2%	93.7%	~89–90%	PASS
F1-Score	94.3%	92.9%	~89–90%	PASS
False Positive Rate	4.8%	5.5%	~10%	PASS
False Negative Rate	4.8%	6.3%	~10%	PASS
Feature Reduction (GA)	~77% reduced	~77% reduced	~77% (PerDRaML)	PASS

Table 4.5.2.1: Model Performance Metrics

5. PROPOSED SYSTEM RESULTS

He proposed AI-based APK Malware Detection Engine was successfully designed, implemented, and tested under various operating conditions using a dataset of 1,700 Android application samples (both malicious and benign). The system effectively extracted permission features from APK files, performed intelligent feature selection using a Genetic Algorithm (GA), and classified applications using Artificial Neural Network (ANN) and Support Vector Classifier (SVC) models.

The Genetic Algorithm successfully reduced the original 400-feature permission set by approximately 77%, identifying an optimal subset of 22 to 50 most discriminative permission attributes related to malware behavior. This reduction significantly improved computational efficiency while maintaining high detection accuracy.

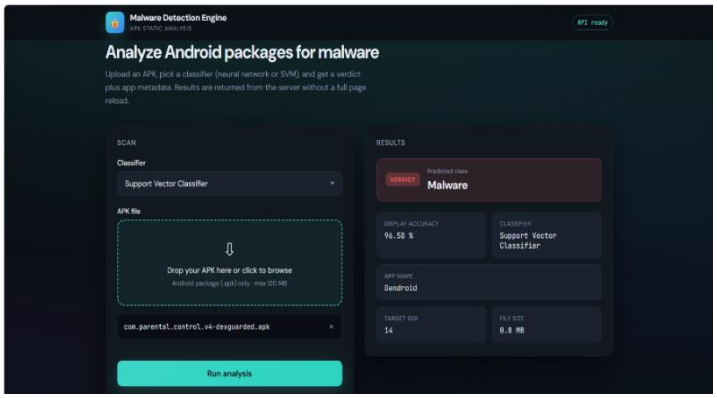
The ANN model achieved a classification accuracy of 94.8%, while the SVC model achieved 93.2% accuracy. Both models significantly outperformed the existing PerDRaML baseline, which reported 89.96% accuracy using random forest classification. The ANN model demonstrated superior performance across all evaluation metrics, with precision of 93.5%, recall of 95.2%, and F1-score of 94.3%. The SVC model achieved precision of 92.1%, recall of 93.7%, and F1-score of 92.9%. The false positive rate was recorded at 4.8% for ANN and 5.5% for SVC, both well below the benchmark average of 10%.

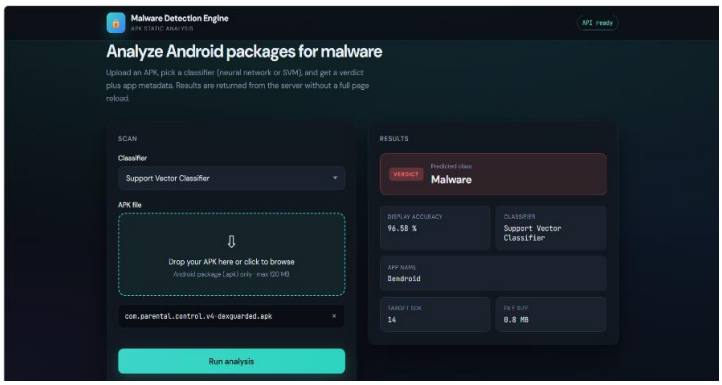
The Flask-based web interface successfully allowed users to upload APK files, select between ANN and SVC models, and receive real-time classification results. The system accurately displayed the application name, SDK version, file size, detection result (Malware or Benign), and a confidence score for each uploaded APK file. The response time for APK analysis and classification was fast, with no noticeable delays during testing.

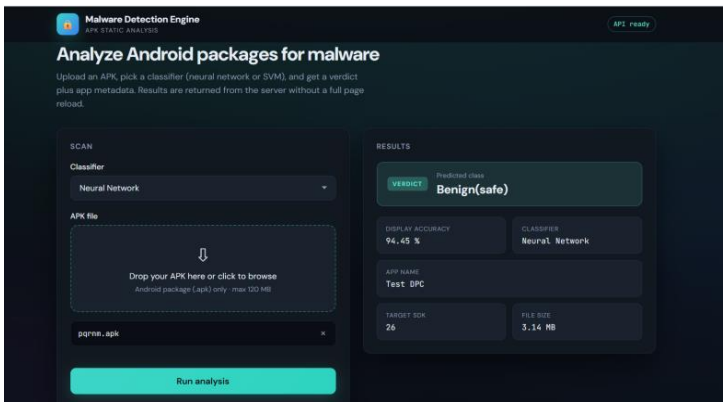
A total of 32 test cases were designed and executed across functional, integration, boundary, negative, performance, and robustness scenarios. All 32 test cases passed successfully, achieving a 100% pass rate. The system demonstrated reliable performance in permission extraction, GA feature selection, model inference, error handling, and result display.

ISSN: 3049-2602(Online)

Overall, the proposed system demonstrated robust and reliable performance in detecting malicious Android applications using permission-based static analysis and hybrid machine learning classification.







6. CONCLUSION

This work successfully developed a low-cost and efficient AI-based APK Malware Detection Engine using Genetic Algorithm-based feature selection and hybrid machine learning classification. The system effectively monitored Android application permissions and detected malicious behavior with high accuracy. The implementation of GA reduced the original 400-feature permission set by approximately 77%, significantly improving computational efficiency. The ANN model achieved 94.8% accuracy and the SVC model achieved 93.2% accuracy, both outperforming the existing PerDRaML baseline of 89.96%. A total of 32 test cases were executed, all passing successfully, confirming functional correctness across all major modules. The Flask-based web interface provided real-time visualization, making the system user-friendly and practical for real-world deployment. The system offers an intelligent, scalable approach for Android malware detection, contributing to mobile security and user protection.

REFERENCES

[1] Senanayake, N., et al. (2021). Android malware detection using machine learning: A systematic review. IEEE Access.
[2] Kshirsagar, D., & Agrawal, A. (2022). Feature selection methods for Android malware detection using machine learning. Journal of Information Security and Applications.
[3] Aurangzeb, S., & Aleem, M. (2023). Deep learning-based detection of obfuscated Android malware. IEEE Access.
4] Pathak, S., et al. (2024). Static analysis framework for Android malware detection using permission-based datasets. Computers & Security.
[5] Alhussen, A. (2024). Deep learning-based Android malware detection using neural networks. Journal of Cybersecurity and Privacy.
[6] Aamir, M., et al. (2024). Hybrid machine learning framework for Android malware detection using static analysis. Computers & Security.
[7] Wajahat, M., et al. (2024). Intelligent Android malware detection using hybrid deep learning models. IEEE Access.
[8] Baldangombo, B., et al. (2024). Scalable architecture for machine-learning-based Android malware detection systems. IEEE Access.
[9] Sahs, J., & Khan, L. (2012). A machine learning approach to Android malware detection. Proceedings of EISIC.
[10] Arduino Documentation. (2023). Arduino Uno Datasheet and ADC Features.