Archives available at journals.mriindia.com

Open Access International Journal of Science and Engineering

ISSN: 2456-3293

Volume 9 Issue 05, 2026

Malware Analysis and Detection using Hybrid Machine Learning

[1]Mayur Rasal, [2]Monika Rokade, [3]Sunil Khatal
[1,2,3] Dept of Computer Engg. Sharadchandra Pawar College of Engineering, Otur, India
[1]rasalmayur98@gmail.com, [2]monikarokade4@gmail.com, [3]khatalsunils88@gmail.com

Peer Review Information

Submission: 30 April 2026
Revision: 09 May 2026
Acceptance: 26 May 2026

Keywords

Android APK, Malware Detection, Machine Learning, Random Forests, Software Security

Abstract

More Android applications mean more troublesome apps. This also shows a rise in user data concerns. This paper hopes to find a way to detect Android malware. Since detection is tough because of the many ways Malware flows through different networks, I have decided to combine a Machine Learning algorithm (ML) with Network Flows to detect Android malware. The dataset is 4,000 network flows. These network flows have source and destination data flows, including port numbers, protocols, flow time, number of packets, number of flows, inter-arrival times, flow flags, and counts. These features help analyze network flow Each of these data flows has a count of inter-arrival time, with its mean, standard deviation, and variance. Incomplete data flows had their gaps filled to maintain integrity. An info gain ranker was used to reduce data flow dimension. A total of 30 data flows was the target to optimize model performance. The dataset was split into 2800 and 1200 for training and testing, respectively, at 70 and 30 %. Several classifying algorithms were used for tests, including Random Forest (RF), J48, and Naïve Bayes (NB). The results show that NB has an accuracy of 93.5%. In contrast, J48 and RF achieved 94.35% and 96.25% (mean accuracy) respectively. This model has outperformed the HML models (decision stump, Random Forest, and Vote) with an accuracy of 98.33%, 96.41%, 99.66%, and 98.01%, respectively (F-measure), and with a 0.33%, 0.33%, and 5.41% improvement over the HML models. The results show that hybrid models enhance both detection accuracy and robustness of the models when compared to single classifiers. The results of this work show that the combination of feature selection and ensemble learning can process high-dimensional data of networks and offer an efficient and robust solution to practical Android malware detection.

Introduction

The development of malware is one of the biggest problems in the modern age of computing. Malware is highly advanced and is developing more evasive behavior, countermeasures, obfuscation techniques, and stealth techniques. Malware is also becoming more polymorphic, and because of that, most computers will not be able to identify malware signatures. That means that detection methods need to be more intelligent. Malware is a rapidly advancing field that needs innovative methods to detect it. Malware is a highly evolving field that relies on cutting-edge methods to detect it, and even then, it continues to grow and advance. Machine Learning (ML) is an up-and-coming field that seems very promising, and it is proliferating to be a useful way to detect evolving fields, malware included. Machine Learning is used to detect malicious apps, and also, benign apps that help develop ML systems can proliferate, especially on the Android app

development side. In turn, these apps can be used to build advanced malware and also develop smart techniques to identify advanced malware. ML can also be utilized to build more advanced static analysis techniques. In this realm, apps that develop static analysis will be very beneficial. More advanced techniques to build applications across multiple operating systems can also be seen. More advanced and integrated applications are being developed alongside better techniques.

Systems and applications are becoming advanced and also less tailored. Techniques that adapt, develop less complicated and less tailored systems. Some advanced techniques lead to the development of more complicated systems. These systems are more tailored, and in this case, they are referred to as more tailored systems. These systems should not be confused with faster systems that are less tailored, which imply that adaptable systems are not developing systems that refine techniques. Systems that use technology possess faster techniques, and refined techniques entail more complicated systems.

Literature Survey

Alhogail et al. [1] a machine learning-based framework for detection and categorization of malware in Android. Their method integrates static feature extraction with advanced preprocessing methods for improved classification. A number of ML methods were tested, and ensemble techniques outperformed single classifiers with respect to precision. The paper stresses the need for datasets being unprejudiced to improve generalization. The framework's applicability to real-life malware analysis is undoubtedly strong.

Using opcode patterns in the memory and semantic analysis, Lowev et al. [2] a state-of-the-art method for the detection of ransomware. Their model is not signature-based, and therefore, relies on the analysis of the intent to be malicious in order to early on detect the inner workings of a convoluted ransomware encryption. The detection accuracy and stamina in the presence of an encryption mechanism is well documented in the findings of the experiments conducted. Proactive cyber security is dependent on memory-based analysis, and this method illustrates this fact.

Xing et al. [3] a malware detection model that employs a deep learning framework and autoencoders. The model is able to carry out unsupervised learning and feature extraction to form concise and meaningful abstractions from high dimensional datasets. When this learned feature vector is employed in a classification algorithm, it is superior to all other manually designed feature sets and vectors. The method demonstrates a decline in the amount of manual feature engineering, and improved generalization. The method its self is a confirmation of its usefulness on a number of datasets.

In the study by Indumathi et al., [4] a machine learning framework was developed by reverse engineering Android applications. The model understands the behavior of the various applications by extracting detailed static features from the decompiled APK files. Several ML techniques were evaluated and were able to achieve a high rate of detection. The framework also promotes interpretability through the analysis of internal structures of the malware. This study affirms the importance of reverse engineering in the field of malware detection.

In their study, Pathak et al., [5] a novel feature-selection approach based on feature importance scores obtained from machine learning models like Random Forest. This approach gives a better feature set, thus shrinking the dimensionality of the data, which in turn allows for better classification and lowers the computational burden. Enhanced performance was reported when fewer features were used among the classifiers that were compared. This work emphasizes the importance of judicious feature selection.

In his study, Ansori et al. [6] the combination of the gain ratio method of feature selection and ensemble machine learning for the classification of Android malware. His method first eliminates irrelevant features, and then applies the classifiers of random forest and gradient boosting. The ensemble model resulted in a high level of accuracy and a low rate of false positives. The method was evaluated across a wide variety of datasets. The results proved the effectiveness of feature selection in combination with ensemble learning.

In [7] their study, Islam et al. presented what is arguably the first feature-selection approach for Android malware classification. his method bettered the performance of the model by removing redundant and noisy features. Also, classification performance was improved by the application of ensemble machine learning. In addition, the system extended beyond binary detection to support multi-class classification. The study illustrated the importance of feature selection in model robustness.

Manzil et al. [8] an innovative method for constructing feature vectors that identifies Android Malware. Their method proves to be more accurate than previous methods for considering malware's both structural and

behavioral. The classification results for machine learning models are more accurate for these feature vectors. Malware classification is improved by this method. This makes for an important development in expressive feature representation.

Arslan et al. [9] designed an ensemble machine learning model to detect various categories of Android malware. This technique combines the outputs of several classifiers to enhance the model's strength and reliability. The model addresses the multi-class classification problem as opposed to merely detecting malware. Compared to single models, the results of this study are more accurate. This study established the foundation for contemporary ensemble methods.

Hasan et al. [10] created a malware detection system that incorporates data scaling and feature selection. The study suggests that model's stability and accuracy can be improved with normalization. A myriad of machine learning techniques were employed and evaluated on various datasets. Performance was notably improved by the combined use of scaling and feature selection. The authors argue that when it comes to malware detection, preprocessing is a significant component.

Methodology

This shows us how to build a machine learning based system to detect Android malware. It has every part that you would expect in a machine learning system: a data preprocessor that works on its own, feature engineering, model training, and even model evaluation. The framework allows you to build a machine learning system that can go from a set of data all the way to an automated model that can tell the difference between good and bad behaviors. This uses a primary set of features based on network flows along with other static and dynamic features. The system covers the entire machine learning workflow from data collection to model evaluation. This system describes other types of machine learning models that are combined to yield better accuracy and performance.

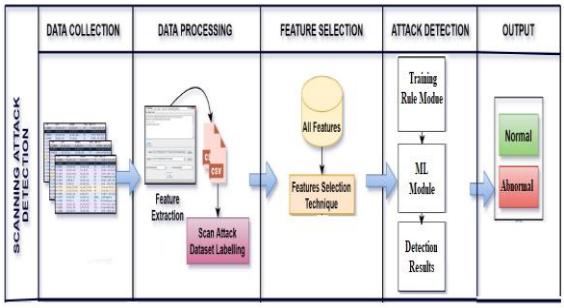

Figure 1: System Architecture

1. Dataset Collection

The first collection of raw data consists of legitimate traffic data, APK analysis, and public repositories (Kaggle, etc.). In this survey, the dataset consists of 4,000 instances of malicious and benign data, which improves the balance during the learning process. Each data record consists of detailed network flow data, including source IP, destination IP, source and destination ports, the inclusion of the flow, flow statistics, packet inter-arrival statistics, and TCP flag counts. The dataset components described illustrate the application's network behavior and capture normal and abnormal behavior. The dataset is divided into two classes: Normal (Benign) and Malware (Attack) for the purpose of supervised learning. Thanks to the wide range of features with high dimensions, the dataset enables models to adequately distinguish between normal and abnormal traffic.

2. Data Preprocessing

The dataset has been organized, formatted, and cleaned to ensure its quality. In terms of quality, the aim is to make data consistent and reliable to work with algorithms. The stage of data formatting and cleaning is crucial to minimize noise in the dataset and improve the model's predictions. The primary tasks associated with this stage are:

- Handling missing data: The Replace Missing Values filter helps to address undefined and missing values to avoid training the model with incomplete and potentially harmful data.
- Normalization or scaling: This refers to the adjustment of values of features with larger magnitudes to a smaller range, in order to help the model converge.
- Malicious network attacks are highly damaging and malicious to network systems. Network benign attacks are neither damaging nor malicious and have been shown to be useful in network systems.
- After all, instances of a dataset have been shown to be either malicious or benign based on their behavior.
- The dataset containing 4000 instances has been split, with 70 % of the dataset (2800 instances) reserved for training, and the remaining 30 % (1200 instances) reserved for testing the model to assess its performance on data it has not seen.

3. Feature Extraction

Some of the things we examine are flow duration, packet length statistics, packet inter-arrival times, and protocol features. These features assist us in understanding and analyzing network traffic statistics. Pre-processing helps to reduce the dimensionality of the dataset and improves the efficiency of analysis and processing. This is relevant, as many features of the existing dataset can be redundant or irrelevant due to the high volume of features.

- Information Gain Attribute Evaluation: a measure of the value of a feature based on the amount by which it contributes to the correct classification. In other words, this measure describes how much a feature is used, and how much it helps in classification.
- Ranker: Features are ranked by the amount of importance, and the top features are selected.

In this work, the top thirty features are selected, and thus, the dataset dimensionality will be reduced, and the volume of noise will be reduced. These features will help to identify and discriminate "malicious" versus "benign" network traffic.

4. Model Training

We created several machine learning models using the training data and the following algorithms:

- Random Forest (RF): A technique that combines the results of many decision trees to improve performance and reliability.
- Naive Bayes (NB): An efficient and optimal stream data classification algorithm, easy to implement.
- An approach based on the combination of decision stumps, Random Forest and Voting: A Hybrid Machine Learning (HML) technique is applied to combine classifiers. The results lead to a model trained to recognize patterns to identify benign and malicious behavior.

5. Model Testing

In this part, we tested the models we built with a trail data set of 1200 instances. The model used the unseen data to decide whether an instance was benign or malicious. Regarding model generalization, we checked the predictions against the actual labels. This way, we were sure that the model was not overfitted and would work successfully on new data. We checked the performance for all the models and individually tested the classifiers RF, NB, and HML.

6. Evaluation

Standard performance measures were used to quantify the success of the model, including:

- Accuracy – the measure of total correct predictions
- Precision – how many predictions of a malware instance were true
- Recall – the measure of detection success for malware
- F1-score – an average of precision and recall
- Confusion Matrix – true positives, true negatives, false positives and false negatives in a visual representation.

Algorithm Process: Hybrid Voting-Based Ensemble Model (AdaBoost + Random Forest)

Input: Training Dataset (Train), Testing Dataset (Test)
Output: Trained Hybrid Voting Classifier + Evaluation Metrics
Step 1: Initialize Base Models
Step 2: Create Hybrid Voting Model
Build Vote Model Using Training Data
Step 3: Evaluate Model
Create Evaluation Object Using Training Data
Test Vote Model Using Testing Data
Step 4: Calculate Performance Metrics
Compute Accuracy = Percentage of Correctly Classified Instances
Compute Precision = Precision of Class
Compute Recall = Recall of Class
Compute F1-Score = F-Measure of Class
Step 5: Return Model
Return Trained Vote Classifier
End Hybrid Model
Step 6: Stop application

Results

Three machine learning techniques—Random Forest (RF), Naïve Bayes (NB), and Hybrid Machine Learning (HML)—assessed the performance of the proposed Android malware detection system. The evaluation dataset is comprised of 4000 instances and divided into 70% training (2800 instances) and 30% testing (1200 instances). The dataset goes through multiple preprocessing steps such as modality detection for missing values, as well as feature selection using Information Gain and Ranker methods. To optimize speed and accuracy, the top thirty features are selected. The models are trained using the training dataset and the performance of the models is evaluated using the testing dataset.

Table 1: The performance results of the implemented models are summarized below

Model	Accuracy	Precision	Recall	F-Measure
Random Forest	96.25%	95.95%	95.95%	95.95%
Naïve Bayes	93.50%	90.88%	90.88%	90.88%
Hybrid Model	**98.33%**	**96.41%**	**99.66 %**	**98.01%**

Table 1 shows the performance of different machine-learning models in Android malware detection based on the metrics of Accuracy, Precision, Recall, and F-Measure. Among the individual models, Random Forest achieves the highest level of Accuracy with 96.25%, along with a Precision and Recall score of 95.95%. This means that Random Forest performs well compared to the other models on more complicated and extended datasets. On the contrary, Naïve Bayes achieves the lowest Accuracy of 93.50%, along with a Precision and Recall score of 90.88%. This is due to a dependent feature assumption, which does not hold true for network datasets that have feature correlations. The Hybrid Model performs better than all other models and is ranked first for all metrics, achieving an Accuracy of 98.33%, with a Precision of 96.41%, a Recall of 99.66%, and an F-Measure of 98.01%. The outstanding Recall of the Hybrid Model shows that it achieves one of the most important goals in the field of Malware detection, which is highly accurate malware detection.

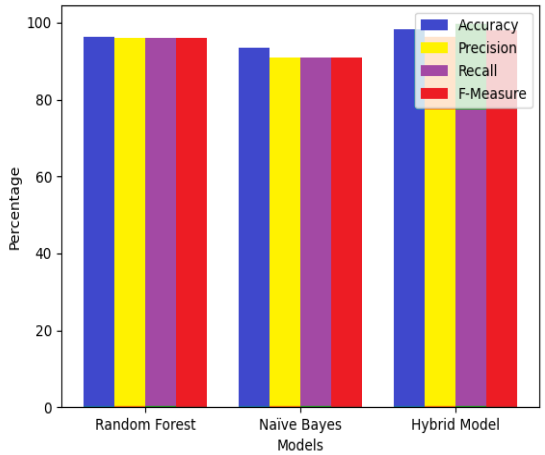

Figure 2: Accuracy Comparison of ML Models

Figure 2 shows how accurate three different machine learning models are. These models are called Random Forest, Naïve Bayes, and the proposed Hybrid Model. Accuracy is the degree to which a model is correct in categorizing something as a phishing threat or as a legitimate case. It can be seen in the graph that the Hybrid Model attains the highest level of accuracy at 98.33%. This means that the model is the best in identifying both positive and negative cases. This accuracy jump is likely because the model uses several different methods as a way to combine multiple approaches which allows for better overall model performance and correction of the shortcomings of the different individual models. The Random Forest model does pretty well at 96.25% of the time being correct. Model performance is bolstered due to the aggregation and learning of many different individual models that comprise the Random Forest Model and also due to the use of decision trees. The Naïve Bayes model is the worst performer at 93.50% accuracy. Although it is fast and performs decently with less data, because it uses a case-independent method, it performs well under simple circumstances but does poorly with complex cases like identifying phishing.

Conclusion

An application for Android malware detection and classification is described in this work. The application is based on evidence of machine-learning approach and the distinct, network flow data. The integrated machine-learning approach utilizes a combination of data preprocessing, feature extraction, feature selection, and several classification techniques to identify malicious behavior with the evidence-based network flow data. Using Information Gain and Ranks, the dataset's relevance was increased, and the efficiency of the model was improved, which ultimately resulted in better model performance. In the machine-learning techniques used in this study, the classification of Random Forest and Naive Bayes methods, Random Forest, was the most successful of the set due to its ability to manage high dimensionality and control overfitting. Nevertheless, the Hybrid Machine Learning (HML) technique, introduced in this study, was unmatched in accuracy (98.33%) when compared to all single or combined techniques and showed the greatest values of precision,

recall, and F-measure. Thus, the most important aspect of this study was the creation of a unified approach to multiple machine-learning techniques and the enhancement of the detection capability and the strength of the classification. The results showed significant improvement of classification. The complete removal of irrelevant and redundant features was largely novel in the accuracy of the outcomes and computational complexity. The imposter model helped capture most of the present malware shown in the data. The study was one of the most helpful in the real-world framework of applied cybersecurity. This is due to the model's ability to capture in the data crime.

References

Alhogail, Areej, and Rawan Abdulaziz Alharbi, "Effective ML-Based Android Malware Detection and Categorization," *Electronics*, vol. 14, no. 8, p. 1486, 2025.

Lowev, Thomas, Charlotte Fisher, and James Collins, "Advanced Ransomware Detection and Classification via Semantic Analysis of Memory Opcode Patterns," 2024.

Xing, Xiaofei, et al., "A Malware Detection Approach Using Autoencoder in Deep Learning," *IEEE Access*, vol. 10, pp. 25696–25706, 2022.

Indumathi, K., et al., "Malware Detection: A Framework for Reverse Engineered Android Applications Through Machine Learning Algorithms," *International Journal of Management Research and Business Strategy*, vol. 12, no. 4, pp. 54–66, 2022.

Pathak, Amarjyoti, Utpal Barman, and Th. Shanta Kumar, "Machine Learning Approach to Detect Android Malware Using Feature Selection Based on Feature Importance Score," *Journal of Engineering Research*, 2024.

Ansori, Dwinanda Bagoes, et al., "Android Malware Classification Using Gain Ratio and Ensembled Machine Learning," *International Journal of Safety and Security Engineering*, vol. 14, no. 1, pp. 259–266, 2024.

Islam, Rejwana, et al., "Android Malware Classification Using Optimum Feature Selection and Ensemble Machine Learning," *Internet of Things and Cyber-Physical Systems*, vol. 3, pp. 100–111, 2023.

Manzil, Hashida Haidros Rahima, and S. Manohar Naik, "Android Malware Category Detection Using a Novel Feature Vector-Based Machine Learning Model," *Cybersecurity*, vol. 6, no. 1, p. 6, 2023.

Arslan, Recep Sinan, "Identify Type of Android Malware with Machine Learning Based Ensemble Model," in *Proceedings of the 5th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, IEEE, 2021.

Hasan, Rakibul, et al., "Enhancing Malware Detection with Feature Selection and Scaling Techniques Using Machine Learning Models," *Scientific Reports*, vol. 15, no. 1, p. 9122, 2025.