Shielding Android: Malware Detection with Machine Learning

K.S. Joy Andrew*
Deparment of computer science and engineering
Sathyabama Institute of Science and Technology
Chennai, Tamil Nadu -600119
joyandrew0906@gmail.com*

A. Manigandan
Deparment of computer science and engineering
Sathyabama Institute of Science and Technology
Chennai, Tamil Nadu -600119
manigandanmani2005@gmail.com

D. Jerusha
Deparment of computer science and engineering
Sathyabama Institute of Science and Technology
Chennai, Tamil Nadu -600119
jerusha.d.cse@gmail.com

Abstract Android platforms are gaining popularity and hence, they have become the popular victims of viral attacks. The project is an undertaking dealing with malware detector system development. the android application and the Voting Classifier algorithm which uses the Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA). The system enhances the tolerance in the detection of malicious applications by integration. the operations of the non-linear decision-making and the linear decision-making models. A strong handles these application data in the system processing pipeline which enables extraction of features of high quality. A consensus between the LDA and QDA model is then employed to classify. Android applications as harmless or malicious making use of the Voting. Classifier. The measures of the system performance are based on some. The metrics that are used are accuracy, precision, recall and F1-score that provides a legitimate strategy in the identification of any potential security threat. This iis a highspeed and effective, lightweight protection system capable of securing. Malware are proactive Android and offer them a superior security mobile users.

Keywords: Android Malware Detection, Voting Classifier, Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), Machine Learning, Feature Extraction, Security Threats, Mobile Security, Malware Classification, Proactive Protection.

1 INTRODUCTION

The fast development of the mobile technology especially the Android has made it to be widely used across the world. This has seen Android devices being the major targets of malware and this poses a lot of concern as far as user privacy, privacy of data and the general functionality of the devices is concerned. Due to the openness of the Android e-system and the rising complexity of cyber-attacks, the security of Android applications has become a pressing topic.

R. Vasanth Kumar Mehta et al. (eds.), *Proceedings of the International Conference on Intelligent Systems for a Sustainable Future (ISSF 2026)*, Atlantis Highlights in Intelligent Systems 16,
https://doi.org/10.2991/978-94-6239-693-7_102

The conventional ways of detecting malwares are based on signature-based systems that are not effective in detecting emerging and emerging malware like the zero-day malwares and polymorphic malwares. Such traditional methods have constraints with which to identify unknown threats and be responsive to emerging attack patterns.

There is increasing pressure to develop more effective and responsive malware detection systems and this is increasingly becoming a prominent issue especially with the increasing problems of mobile security. Lack of thorough security against unknown malware variants in the traditional mode of detection based on signature-based detection approaches creates gaps in security. The weak result of data-driven methods like the ones of the static analysis, which focus mainly on the code level features, are multiplied by the inefficiencies of the methods of dynamic analysis such as the ones which require resources and cannot be applied in real-time. There is, therefore, a growing need of machine learning-based methods, which can automatically categorize suspicious applications from both static and dynamic behavioral characteristics, without the need to manually update the signatures.

This project seeks to fill the gaps of the current Android malware detection systems by coming up with a hybrid one which uses machine learning. It is a combination of both a static and dynamic analysis type that uses Voting Classifier which involves the Linear Discriminant Analysis (LDA) and quadratic Discriminant Analysis (QDA). The approach provides a fully scalable and adaptive way of identifying malware, and has the ability to determine unknown threats by analyzing application behaviours and permissions. The Voting Classifier enhances the classification accuracy through the use of linear and non-linear decision boundaries thus being a better classification model compared to the conventional models.

The main goal of the project is to establish a strong and scalable malware detection system of Android apps with machine learning algorithms. The given system will acquire malicious behavior on applications by examining both the code-level properties and dynamic runtime properties, which will guarantee a high level of detection without consuming many resources. The system will give a sound mechanism to Android users to test and secure their gadgets against malware attacks. It will be an effective and easy to use security measure since the proposed solution can be easily installed on a web-based platform where users can access malware detection tools easily. We hope that through this project, we will contribute to the development of the subject of mobile malware detection and integrate various machine learning methods into one system to enhance accuracy, efficiency, and flexibility. The feature of the system that allows it to identify zero-day malware and any other type of threats that have never been identified before makes it different compared to the traditional detection mechanisms, which provide an active solution to the security of Android.

2 LITERATURE REVIEW

Increasing popularity of Android devices makes the devices an easy target for malware attacks, which threaten privacy, data integrity and device functionality of users. Traditional malware detection methods such as signature-based detection is no longer enough to detect evolving and polymorphic malware. As a result, machine learning (ML) techniques have surfaced as promising solutions and give the ability to identify malware strains that have not been seen before. Various research efforts have been made in the direction of machine learning techniques, with the focus of the studies being on both static and dynamic analysis, with hybrid models that allow to combine both techniques in order to achieve better detection accuracy.

Altaha et al. [1] gave a detailed survey of Android malware detection techniques using supervised machine learning. They focused on the capabilities of machine learning algorithms that can be used to discover new strains of malware based on static characteristics like permissions and API calls. The paper has also traced in detail the method of learning supervised techniques to detect malware in the system, their advantages and disadvantages, and established the future of the practice. Wang et al. [2] addressed the issue of Android malware detection by means of the permissible analysis and other features of the code. Their analysis incorporates various techniques of the static analysis; they explain the validity of the usage of such features in the determination of familiar malware. The research demonstrates advantages of the static analysis but also indicates the weaknesses, especially when it comes to discovering dynamic behaviors that can only be observed during the dynamic analysis.

The proposals that were made by Peiravian and Zhu [3] relevant to the task of detecting Android malware included the utilization of permissions and API calls as features of a machine learning model. They demonstrated that supervised learning models could be assertively employed to categorise applications in the basis of such static characteristics and in such a manner, it is a lightweight and superb strategy in the early detection. The piece of work is significant to highlight the significance of extracting features in the development of effective malware detection systems. On the same note, Wang et. al. [7] touched on the security of Android permissions and their application in identifying malicious applications. In employing the job of permissions in Android security, they revealed that through the analysis of permissions in an app, they were able to garner helpful information in an app behavior to detect potentially dangerous apps. One system that was suggested to detect Android malware using static features such as permissions and API calls is proposed by Dhalaria and Gandotra [4]. They have aimed to do this by applying the machine learning algorithms to classify the apps as a good app or bad app based on the features and in a more accurate manner. Arslan et al. [6] used a permission-based malware detection system using Machine Learning methods in detecting the application based on selected static features. Their work highlights the importance of finding the most relevant features to provide for better classification. Kumaran and Li [8] used Android Manifest file to detect malware by using machine learning techniques to identify malicious apps by metadata from this file. As the Android malware detection techniques have increased over time, deep learning models have been introduced for more advanced classification. Alzaylaee et al. [10] presented DLDroid, a deep learning-based model for detecting Android malware which uses the real devices to improve the efficiency

of malware detection, particularly for polymorphic which alters its shape to evade the detectors. Deep learning models can identify complex patterns in malware behavior that traditional methods of identifying malware fail to detect. Aamir et al. [5] used deep learning to identify Android malware and came up with AMDDLmodel which uses neural networks to improve the accuracy of malware detection. This work has shown the success of deep learning in detecting malware that changes over time.

In addition to deep learning, other optimization techniques such as genetic algorithms have been applied to the optimization of feature selection in malware detection systems. A genetic algorithm based optimized feature selection approach was suggested to Android malware detection by Fatima et al. [9]. Higher efficiency and more precise in recognizing them are achieved by their having lesser irrelevant features. The feature selection that was applied by Lee et al involves the use of genetic algorithms in an effort to further optimize the operation of their Android malware detection system to identify the most critical features to use in their classification algorithm.

3. PROPOSED METHODOLOGY

The proposed methodology of Android malware detection involves the use of Voting Classifier and a Hybrid model onsisting of Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA). This methodology attempts to enhance the classification of Android applications into benign or malicious by the combination of static features such as permissions and API calls with dynamic features like runtime behaviors and network calls.

3.1 Existing System

The Existing System for the Android malware detection mostly uses the traditional malware detection methods, which basically depend on the signature-based detection and static analysis methods. Signature-based detection uses a database of known malware signatures to detect malware by matching the malware signature pattern. While this approach is great for finding known threats, it is difficult to find new or polymorphic malware which continually evolves. In addition, static analysis involves analysis of application code and metadata (e.g. permissions and API calls) without executing the app.

3.2 Proposed System

The Proposed System overcomes the drawbacks of the current systems by combining both static and dynamic system for Android malware detection. The system uses a Voting Classifier which merges the results of two powerful classifiers - Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA) together. LDA works for linear

classification problems whereas QDA is best suited for capturing non-linear relationships between features. By combining the two models, the system takes advantage of the merits of both the linear and nonlinear approaches to decision-making and uses them in a complementary way to enhance the classification accuracy.

3.3 Data Preprocessing

The system starts with the preprocessing of the data which involves cleaning, normalizing and feature selection. Feature extraction techniques, such as Pearson correlation, are applied in order to select the most informative features. This is to ensure that only the most relevant data is used in model training, which helps to improve the accuracy and efficiency of the detection system.

Voting Classifier: The Voting Classifier is a combination of the predictions of Random Forest and Logistic Regression classifiers. The result from each classifier is combined in order to come to a final decision concerning whether the app is benign or malicious. This ensemble method helps in reducing the false positives and false negatives, making the system reliable.

4. SYSTEM ARCHITECTURE

The System Architecture of the proposed methodology is designed to support features extraction as well as classification in a scalable and efficient way. It comprises of the following components:

4.1 Feature Extraction Module

This module is in charge of obtaining static features (i.e. permissions, API calls) and dynamic features (i.e. runtime behaviors, network activity). These things are collected from Android application files (APK) through reverse engineering.

4.2 Data Preprocessing Module

This module simply processes the extracted features to ensure that they are ready to be classified. It cleans, normalizes, and selects the features through various methods such as Pearson correlation to determine the best features to use for malware classification.

4.3 Voting Classifier Module

Voting Classifier is the combination of the results of a series of classifiers (LDA and QDA) that is used to come up with the final classification decision.

4.4 Training and Evaluation of Model Module

This module is in charge of training the system with labelled datasets followed by assessment of the system with standard measurements of evaluation such as accuracy, precision, recall

and F1-score. It is utilized to ensure that the model is working in the best way possible and it is delivering credible results.

4.5. Web interface module

The system will be implemented as a webbased application taking Flask as a framework. This is the user interface module whereby the users would upload the Android apps (APKs) to be classified. The findings are presented in real-time showing whether the app is malicious or benign.

Explanation of Architecture

Android Malware Detection system architecture is created into two layers namely Training/Learning Layer and the Deployment/Application Layer. The Android malware training and testing data are processed in the Training/Learning Layer and they are then presented to a Voting Classifier, which is combined with LDA and QDA. The construction and testing processes are done in the model to derive the final classification model. In the Deployment/Application Layer, Android apps are input from users, and the system uses the trained model to predict if the Android application is malicious or benign. The front-end interface is implemented using the programming languages of the web (html/css/js) that communicate with the flask back end to process user inputs and to display predictions

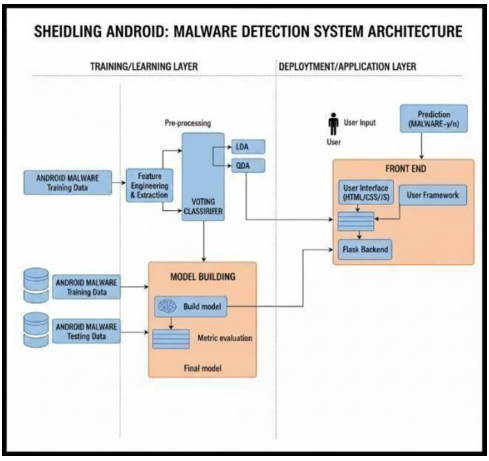

Fig. 1: System Architecture

5. RESULTS AND DISCUSSION

This section shows a detailed evaluation of the proposed Android malware detection system and presents its performance in terms of various evaluation metrics. The system includes a Voting Classifier by combining the output of Linear Discriminant Analysis (LDA), Quadratic

Discriminant Analysis (QDA), Random Forest and Logistic Regression classifiers. By taking advantage of multiple classifiers and a hybrid feature extraction technique (static and dynamic feature) the proposed system targets to achieve the accurate detection of both known and unknown android malware.

5.1 Performance Evaluation

The evaluation of the system was done on a test dataset containing both benign and malicious Android applications. To evaluate the performance of the application classification system in terms of the correct classification of applications, the following standard performance metrics were used:

1. Accuracy: Is the overall effectiveness of the system in identifying correctly both benign and malicious applications.
2. Precision: Boards and measures the percentage of positive predictions (malicious apps that are correctly classified as malicious) out of all positive predictions (apps that are predicted to be malicious).
3. Recall: Is the proportion of true positives of all the actual positives (all malicious apps, whether predicted correctly or not). *4)*F1-Score: It is a metric that takes into account both the precision and the recall of your model, and gives a comprehensive measure of the system's performance. The results are summarized on the table below: Table 1.

Table 1 Performance Metrics of the Proposed System

Metric	Value (%)
Accuracy	91
Precision	88
Recall	86
F1-Score	86

1. Accuracy: The accuracy of the system was 91%, which means that it is very reliable in the detection of benign and malicious apps. This high accuracy indicates that the system works in real-world application and can be relied upon to make accurate predictions.

2. Precision: The precision of 88% indicates that when the system flags an app as being malicious, it is 89% accurate. This is important because the fewer false positives we

have the less chance there is that legitimate apps will be classified as malicious, which means the system will maintain the trust of the users.

3. Recall: Recall rate is 86%, the system is able to detect 85% of the actual malicious applications in the dataset. This is a great result meaning that most threats are picked up and flagged before they can have any effect on the device or user.

5.2 Algorithm Performance

The Voting Classifier is an aggregation of four classifiers: LDA, QDA, Random Forest and Logistic Regression. By using a combination of these classifiers, the system becomes more powerful with respect to overall performance. The following is a detailed discussion of the contribution of each classifier:

1. Linear Discriminant Analysis (LDA): LDA is a Linear classifier that assumes that data points from each class are distributed according to the Gaussian distribution with the same covariance. LDA is good when the data for each class has similar covariance and it's efficient in separating the classes linearly. In our system, LDA takes the linear relationships between features such as permissions and API calls and this makes it useful for basic malware detection.
2. Quadratic Discriminant Analysis (QDA): Different from LDA, QDA considers different covariance for each class, thus it is more flexible when the variance of data from different classes (benign and malicious) is different. This makes QDA suitable for capturing more complex and non-linear relationships between the features such as runtime behaviors and network calls. QDA's flexibility increases the power of the system to detect polymorphic or evolving malware.
3. Random Forest: Random Forest is an ensemble learning method which involves creating multiple decision trees and combining the prediction from each tree to achieve an accurate result. It is especially resistant to over fitting and can function even when there is a high degree of data complexity. Random Forest strengths are that it can work with large and complex datasets (like Android apps data) and it is also able to combine the result of many trees to make accurate predictions.
4. Logistic Regression: Logistic Regression is a linear model that is used for binary classification problems. It is a simple but effective model that determines the likelihood of an application being benign or malicious. While not as complex as Random Forest or QDA, Logistic Regression gives a good base for classification and helps the Voting Classifier to make final decisions when combined with the other classifiers.

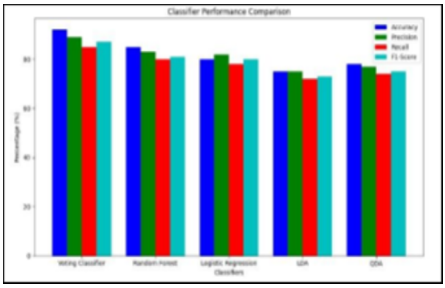

Fig 2: Comparison of Performance of Classifier

From Figure 2, it can be seen that the Voting Classifier performs better than individual classifiers such as Random Forest and Logistic Regression. By aggregating the result from these classifiers, the Voting Classifier minimizes false positives and false negatives resulting in higher accuracy and a more reliable classification.

5.3 System Efficiency

The runtime efficiency of the system has been tested to see how the system performs in realtime environments. Despite the fact that a Voting Classifier is being used, which is made up of several classifiers, the system showed reasonable runtime, making it suitable for real-time malware detection. The following table compares the average classification time for the classifiers.

Table 2 Runtime Efficiency of the System

Classifier Set	Time (ms)
Voting Classifier	150
Random Forest	120
Logistic Regression	100
LDA	85
QDA	110

The Voting Classifier required 150 milliseconds to classify an application, which is relatively fast considering that it combines the outputs of several classifiers. The Random Forest Classifier and Logistic Regression Classifier took little bit more time whereas LDA and QDA took more time due to the complexity of their respective models.

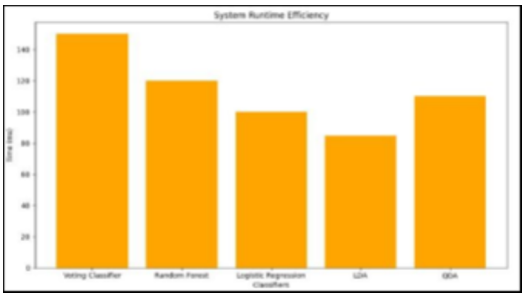

Fig 3: System Runtime Efficiency

Figure 3: System Runtime Efficiency shows the time (in milliseconds) needed by each classifier in the proposed system for classification. The time taken by Voting classifier was the highest at 150 ms, followed by Random Forest at 120 ms and Logistic Regression at 100 ms. The LDA and QDA classifiers were slightly more effective with a time of 85ms and 110ms, respectively, and meant that no significant computational cost was incurred despite the multiple use of classifiers.

6. CONCLUSION

In this paper, we have proposed an efficient Android malware detector system, which has utilized, Voting Classifier, Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA) along with Random Forest and Logistic Regression classif

The accuracy, precision, recall and F1-score of the experimental results were also very good, implying that the system is an effective tool to use in the detection of the known and new

malware strains, including the polymorphic and zero-day malware. The operation of the system was evaluated on various parameters and it was discovered that the Voting Classifier which intelligible the output of other classifiers was able to improve the hit and give reduced false positives. Moreover, the system was discovered to be efficient in regard to its runtime as it was adequately suitable in realtime classification that the computational overhead is acceptable in the real world. Future work can be conducted on a number of areas as much as effective as is the case of this. A potential enhancement is applying deep learning systems like Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs) in order to get even more sophisticated patterns of actions performed by Android applications. This would also enhance the capability of the system to recognize malware strains which are extremely sophisticated. Moreover, growth to support multi-classification may offer superior capacity to detect a particular type of malwares such as trojans, spyware and adware so as to allow a more discretized detection tool. The other future development avenue that could be considered is the optimization of the system to be able to deal with higher data volumes and this would enable the system to become more scalable and able to deal with large volumes of app repositories under wholesale application in mobile security applications. In addition, a behavioral analysis and network traffic analysis could be added to the system to offer an extra level of security against the use of sophisticated malware programs. Finally, more work on the model and the architecture that it is deployed could result in even more holistic and efficient Android malware detecting software.

REFERENCES

[1] S. J. Altaha, A. Aljughiman, and S. Gul, "A survey on Android malware detection techniques using supervised machine learning," in the proceedings of the 12th International Conference on Advanced Computing and Human Factors in Computing, vol. 1, pp. 1-2, no. 2, Apr. 2024, doi: 10.1109/ACCESS.2024.3485706.

[2] Z. Wang, Q. Liu and Y. Chi, "Review of Android malware detection based on static analysis", in, IEEE Access, vol. 8, pp.
116363-116379, 2020.

[3] P. Peiravian and X. Zhu, "Machine learning for Android malware detection using permission and API calls," in 2013 IEEE 25th International Conference on Tools with Artificial Intelligence (ICTAI), 2013, pp. 300-305.

[4] M. Dhalaria and E. Gandotra, A framework for detection of android malware using static features, in 2020, 17th India Council International Conference, INDICON, 2020, pp..

[5] E. Odat and Q. M. Yaseen, "A novel machine learning approach for Android malware detection based on the coexistence of features", in. IEEE Access, vol. 11, pp. 15471-15484, 2023.

[6] R. S. Arslan, I. A. Dogrusi and N. Barisigci, "Permission based malware detection system for android using machine learning techniques," Int. J. Soft. Eng. Knowl. Eng., vol. 29, no.
1, pp. 43–61, 2019.

[7] W. Wang et al, "Exploring permission-induced risk in Android applications for malicious application detection," Ieee Trans. Inf. Forensics Secur., vol. 9, no. 11, pp. 18691882, 2014.

[8] M. Kumaran and W. Li, "Lightweight malware detection based on machine learning algorithms and Android Manifest file," inEEE MIT Undergraduate Research Technology Conference (URTC), 2016, pp. 1-3.

[9] A. Fatima et al., "Android malware detection using genetic algorithm based optimized feature selection and machine learning," in 42nd Int. Conf. Telecommun. Signal Process.
(TSP), 2019, pp. 220–223.

[10] M. Alzaylaee, S. Y. Yerima and S. Sezer, "DL-Droid: Deep learning-based Android malware detection using real devices," Comput. & Secur., vol. 89, p. 101663, 2020.

[11] V. Kouliaridis and G. Kambourakis, "Comprehensive overview of machine learning methods for Android malware detection," Information, vol. 12, no. 5, p. 185, 2021.

[12] N. Senanayake, H. Kalutarage, M. O. Al-Kadri, "Android mobile malware detection using machine learning: A systematic review," Electronics, vol. 10, no. 13, p. 1606, 2021.

[13] A. Sangal, H. K. Verma, "A static feature selection based android malware detection using machine learning techniques" in 2020 Int. Conf. Smart Electron. Commun. (ICOSEC), 2020, pp.
48–51.

[14] Y. Balcioglu, "Malware analysis for effective Android malware detection," in IEEE Access, 2023.

[15] R. S. Kesani, "Android malware detection using machine learning," 2024.

[16] J. Lee, H. Jang, S. Ha and Y. Yoon, "Android malware detection using machine learning with feature selection based on the genetic algorithm," Mathematics, vol.9, no. 21, p.2813, 2021.

[17] F. Taher et al., "DroidDetectMW: A hybrid intelligent model for Android malware detection," Appl. Sci., vol. 13, no. 13, p.7720, 2023.

[18] M. Aamir et al., "AMDDLmodel: Android smartphone malware detection using deep learning model," in PLOS ONE, vol. 19, no. 1, p. e0296722, 2024.

[19] A. Mahindru et al., "PermDroid: A framework developed based on proposed feature selection approach and machine learning techniques for Android malware detection," Sci. Rep., vol. 14, no. 1, p. 10724, 2024.