Sunil Sahoo

Sunil Sahoo COM_18 RESEARCH PA...

Quick Submit

Quick Submit

Presidency University

Document Details

Submission ID	trn:oid:::1:3430590828
Submission Date	Dec 2, 2025, 2:53 PM GMT+5:30
Download Date	Dec 2, 2025, 3:20 PM GMT+5:30
File Name	Sunil Sahoo COM_18 RESEARCH PAPER (1).pdf
File Size	528.9 KB

6 Pages

4,320 Words

24,422 Characters

0% detected as AI

The percentage indicates the combined amount of likely AI-generated text as well as likely AI-generated text that was also likely AI-paraphrased.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Detection Groups

0 AI-generated only 0%
Likely AI-generated text from a large-language model.

0 AI-generated text that was AI-paraphrased 0%
Likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?
The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?
Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.

Analysis and Identification of Malicious Mobile Applications

Sunil Kumar Sahoo
CSE
Presidency University
sunilkumar.sahoo@presidencyuniversity.in

Roshan Raju K K
Computer Engineering
Presidency University
roshanraju1102@gmail.com

Amogh Malage
Computer Engineering
Presidency University
amoghmalage004@gmail.com

Bhoomika S Horapeti
Computer Engineering
Presidency University
bhoomikashorapeti8@gmail.com

***Abstract*—As Android apps continue to increase, issues of insecure apps posing security threats to users and developers have become a big concern. Old signature based detection mechanism cannot usually work with new malware. This paper introduces a machine learning style of analysis and identification of malicious mobile applications with the help of the mh100klabels.csv dataset. The Random Forest classifier is trained to separate between benign and malicious applications on the basis of extracted metadata properties. The model recorded a total accuracy of 90.94 to prove the usefulness of features based on learning in detecting malware. As experimental observations indicate, the proposed approach may be rather effective in improving the security of a mobile device due to the smart identification of malicious patterns.Malware detection Android with machine learning has become an effective substitute to existing signature-based schemes with a greater ability to adapt to zero-day threats. Recent researches have shown that the hybrid and ensemble-based detection models are effective in enhancing the classification capability of mobile ecosystems. A classification algorithm built with random forest is used in this work to identify malicious applications with the mh100klabels.csv dataset and the classification accuracy is 90.94. The findings affirm that machine learning can be used in the applications of automated malware classification and real-time threat detection procedures.**

***Keywords*—Android Security, Machine Learning, Malicious Applications, Random Forest, Malware Detection.**

I. Introduction

Smartphones are now an inseparable element of the modern digital life and they offer consumers the possibility to communicate, bank, entertain and access many online services using mobile applications. Millions of apps are offered on the websites like Google Play and Apple App Store so the users more and more rely on them in their every-day life. Nevertheless, mobile ecosystems are open and dynamic, which poses serious security threats. Hackers take advantage of the flaws in the distribution system of apps to spread harmful applications (malware) that are capable of invading the privacy, financial resources, and even the functionality of the user.[11][12]

Bad apps can be masqueraded as normal programs and fool the users into allowing them access to sensitive permissions. They do not have to be uninstalled, as after being installed they could steal the personal information like login credential, venue or location and even the financial information of the user or grant them unauthorized access to the resources of the system. Their increasing complexity has rendered the conventional antivirus and signature-based systems of detection ineffective. These systems are based on pre-defined malware signatures or known patterns of behaviors and hence do not recognize new, obfuscated or polymorphic malware that has constantly changed to elude detection.[14] Machine learning (ML) has come out as a potential solution to the malware detection problem in response to these challenges. ML-based methods allow processing large amounts of data automatically, and complex patterns and relationships are identified that are not reflected in more traditional rule-based systems. ML models are able to generalize unseen or zero-day attacks, by being trained on large amounts of benign and malicious applications respectively.[16]

This paper seeks to devise a machine learning-based malicious application detection system, whereby malicious Android apps are automatically recognized by analyzing the metadata of the application and the features extracted. The proposed system makes use of permissions, API calls, component structures, and network behaviors to decide on what applications are good or bad. Besides improving the accuracy of the detection, the use of ML means less time and effort spent on manual malware analysis.[18]

Finally, this study helps to build up the security of mobile applications by showing that the data-driven intelligence could be an effective component of the mobile apps vetting procedures to give users and developers a proactive protection mechanism against the constantly changing environment of mobile cybersecurity threats.

The proliferation of Android apps exponentially has posed also major cybersecurity dilemmas because malicious apps are also starting to take advantage of security vulnerabilities and bypass conventional security measures. The recent studies reported an increase in the rate of the use of polymorphic and obfuscated malware all over the world, which necessitates the use of intelligent detection mechanisms [12], [16]. Machine learning and deep learning have become an attractive alternative because they can acquire complicated behavioral patterns and adjust to changing threats [11], [18]. There is also the case of permission-based static analysis which has also shown to be useful in the construction of lightweight datasets in a way that can be trained to large-scale models as it has been demonstrated by other researchers in automated feature extraction frameworks [14]. Nevertheless, irrespective of the good progress, the ability to detect zero-day samples, accuracy in the imbalance in datasets, and minimizing false positives are still open research problems in the field.

II. Related Works

Detection and analysis of malicious mobile applications have been of great research interest owing to the rise in the number of Android malware and its effects on user privacy and security of their devices. The initial approaches, like Drebin by Arp et al. [1], used the feature of the APK files derived through a static analysis, including permissions, API calls, and network addresses, and proved that machine learning could effectively be used to classify the known malware families, though they are less efficient against the obfuscated or the emerging malware. Sahs and Khan [2] suggested a support vector machine (SVM)-based method with application permissions, intent filters, with a high detection rate of known threats, but it was so dependent on manually engineered features that it was not very adaptive. Tam et al. [3] noted that malware developers are becoming more dynamic in their code loading, encryption, and polymorphic methods, and that static-only methods were no longer effective. To overcome this, dynamic analysis techniques have been proposed where the runtime behavior, system calls and network activity has been tracked to identify malicious behavior during execution. Alam et al. [4] performed the overall survey of the static, dynamic and hybrid methods and came out with the verdict that the hybrid methods or those that combine properties of the static and runtime lead to high detection accuracy and generalization and that they balance on the cost of computation. Simultaneously, deep-learning methods have also come into existence, which can automatically acquire complex features based on the features of the apps. Kim et al. [5] used convolutional neural networks (CNNs) on opcode sequences to provide high classification rates and Yuan et al. [6] designed DroidSec systems that used a deep belief network that could detect malware without the use of manual feature engineering. Ensemble and tree-based methods have been studied elsewhere; Random Forests, Decision Trees, and Gradient Boosting have been found to be effective because of their resistance, interpretability, and the capacity to operate with high-dimensional space of features. Arora et al. [7] and Li et al. [8] demonstrated that the Random Forest classifier is better than single classifiers in identifying malware with regards to metadata, API calls, and permissions which revealed their applicability to large-scale datasets. In addition to the static code and metadata, network and behavioral analysis has been used in an attempt to improve detection; Su et al. [9] utilized the patterns of network traffic to detect an abnormal behavior that may indicate malware, and Deshotels et al. [10] used a combination of both static, dynamic, and network features in a hybrid-based detection system, which is able to better classify detections. Later works have been more concerned with permission-based risk scoring, feature selection, and lightweight models that can be applied to mobile devices, and have shown that through careful feature engineering and ensemble learning it is possible to obtain high accuracy whilst remaining computationally efficient. In contrast to these previous studies, which are based on either fixed, dynamic, or hybrid characteristics, the current research is focused on metadata-based analysis with the help of the mh100klabels.csv data set and a Random Forest classifier to obtain a precise, scalable, and understandable paradigm of detecting malicious mobile applications in the real-world context.[11][12][15][18][19][20]

III. Methodology

The given solution will enable to categorize mobile applications based on their benevolence or malevolence and use a machine-learning-based solution, which relies on metadata characteristics obtained by means of the mh100klabels.csv dataset. The methodology is composed of four key steps which include data preprocessing, feature extraction, model selection and training, and performance evaluation.

A. Data Preprocessing

The data set contains metadata of more than 100,000 Android applications with labels of either benign or malicious. The first preprocessing stage included the deletion of inconsistent records, the management of missing records, and text-based features standardization. All the values of features were de-whitened and de-non-alphanum. Categorical features, including app types, permissions and developer details were coded with label encoding or one-hot encoding to convert them into number form which can be used in machine learning algorithms. The data was subsequently divided into training and testing (80 percent and 20 percent respectively) in order to guarantee assessment without bias of model performance. Besides that of missing values, formatting inconsistency, the dataset was also evaluated in terms of noise, duplicates, and abnormal permission strings that might have a detrimental impact on model learning. Machine learning performance may be impaired by irrelevant or redundant data so a cleaning pipeline was used to normalize labels and standardize categorical features. Moreover, the issue of class imbalance was overcome in the preprocessing by implementing the stratified sampling in training and evaluation stages to have the model balanced in terms of both benign and malicious samples. This was essential in averting bias of the majority benign class, which is a well-known problem in malware detection studies.

B. About Dataset

The dataset is a realistic representation of mobile environments because of the variety of type of application involved, such as utility, communication, finance, entertainment and background services. The analysis of metadata showed that malicious apps are more likely to demand a significantly more substantial number of sensitive permissions than benign ones. The format of the dataset facilitates a systematic process of extracting features, which can be used in both conventional machine learning models and deep learning pipelines and, thus, it is highly versatile to be used in additional experiments. The data set that is employed in this paper is the mh100klabels.csv that comprises metadata of more than

100,000 Android applications, gathered in different sources. Every dataset record has a label to show whether the application is benign (0) or malicious (1), and application name, developer information, requested permissions, size, version code, and number of downloads.

This set of data has a full picture of benign and malicious applications, which can be analyzed by the use of machine learning features. Preprocessing during the first stage involved the analysis of the dataset in terms of missing or inconsistent values. Categorical variables like types of apps and information about the developers were coded as numbers and those variables that were numerical were scaled to give equal contribution to the model. The dataset is not balanced whereby there were more benign applications than malicious applications which was mitigated by applying weighted evaluation metrics during model training and testing in order to avoid bias on the majority class.

The mh100klabels.csv dataset allows studying the practical application behavior patterns, permissions, and metadata, which is why it is applicable to scalable and practical machine learning-based malware detection.

C. Feature Extraction

In malware detection, feature selection is a very important task. Metadata properties like the permissions requested, downloads, size of this application, version code, and reputation of the developer were all taken into account in this study. These attributes detect important attributes of malicious applications, such as abnormal permissions requests, abnormally elevated or reduced number of downloads, and activity of suspicious developers. In cases where it was needed, feature normalization was used to scale numerical features such that all the inputs have an equal contribution to training the model. The main focus of feature extraction was based on recognition of static signs of evil activity, including permission usage pattern, application intent filter, and metadata fields. The binary representation and categorical representation of features were coded in numbers in order to make them compatible with the Random Forest classifier. Relevance was checked by the importance scores produced in the process of model training and non-contributing features were dampened to avoid overfitting. This enabled the model to pay attention to high-impact security relevant features that have a great correlation with malicious behavior.

D. Model Selection and Training

We have chosen the Random Forest Classifier as it is difficult, simple to comprehend, and capable of operating with large numbers of features. Random Forest is a combination of choices made by a number of trees. In that manner, it will not stall on the peculiarities of the training data but it is more effective as you present new data to it. We would adjust such parameters as the number of trees, the depth of each tree, and the minimum number of samples that can be used to create a split. Cross-validation also helped us to determine the most appropriate settings. Training To train it, we gave the random forest the training data and allowed it to learn what distinguishes between benign apps and malicious ones.

Random Forest has been chosen because it has been known to work effectively with high dimensional data, is also extremely resistant to noise and also generalizes fairly well without significant parameter adjustment. The dataset was divided into 70 percent to be used in training the model as well as 30 percent to be used in evaluation to come up with the model performance which would be unbiased in measuring performance. The grid search was used to tune hyperparameters including the number of estimators, maximum depth and split criteria to maximize prediction accuracy. The last model structure exhibited convergence consistency and high predictive validity in successive runs.

E. Model Evaluation

To determine the performance of the model, we tested it on a different dataset and verified accuracy, precision, recall, and F1-score. These scores indicate the level of effectiveness of the model in identifying malicious apps, which is significant because they are not as abundant in the data. We also created a confusion matrix, as we needed to know where the model did a good job or missed its step true positives, false negatives, etc. In addition to that, we also computed macro and weighted averages to have an idea about an overall performance in the various classes. In this way, it is easy to identify bad apps by simple metadata and the Random Forest algorithm, which proves to be easy to scale and accurate. The results of these experiments will be found in the following section.

Let:
TP = True Positives (malicious apps correctly detected)
TN = True Negatives (benign apps correctly detected)
FP = False Positives (benign apps incorrectly detected as malicious)
FN = False Negatives (malicious apps missed by the model)

1) Accuracy measures the proportion of correctly classified instances:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

2) Precision measures the proportion of correctly predicted positives among all predicted positives:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

3) Recall (also called Sensitivity) measures the proportion of correctly predicted positives among all actual positives:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

4) F1-Score is the harmonic mean of precision and recall:

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

The metrics of evaluation were precision, recall, F1-score and accuracy because the metrics give more insight as compared to the raw accuracy (particularly imbalanced datasets). A confusion matrix was also produced to determine trends in misclassifications that assist in identifying the possible areas of weaknesses in detecting the subtle or obfuscated malicious applications. Cross-validation was also done to ensure that the results were reproducible and because of data bias or overfitting, the results would not be inflated.

IV. Results and Discussion

The Random Forest model really pulled its weight on the mh_100k_labels.csv dataset, nailing mobile app classification as either benign or malicious. It hit an accuracy of 90.94%, so it got most of the test apps right. If you want all the details, check out Table I for the full rundown of performance metrics for each class. The findings demonstrate that the model is very effective in identifying benign applications, and the accuracy is 0.97 and the recall is 0.92, which implies that there are very few applications that are benign and are classified as malicious. In the malicious case, 0.81 value of the recall means that this model identified most of the malicious applications, but the lower value of 0.60 which signifies a false positive that is, some innocent applications were wrongly detected as malicious. In practice, it is frequently desired that high recall in the malicious class should be prioritized because a false negative may result in very serious security violations.The accuracy of the Random Forest model in this paper is similar to other related machine learning-based Android malware detectors that are present in the literature. As an example, a Random Forest-based model in [13] has found similar detection rates with a smaller dataset, and more feature-optimized models have found slightly higher scores in terms of precision and recall rates [19][20]. Moreover, the recent experimental works that used hybrid behavior-static classification pipelines had also claimed better resilience to obfuscation but incurred a much higher computational cost [17]. Such comparisons endorse the fact that the proposed implementation is a tradeoff between the computational efficiency and the detection.

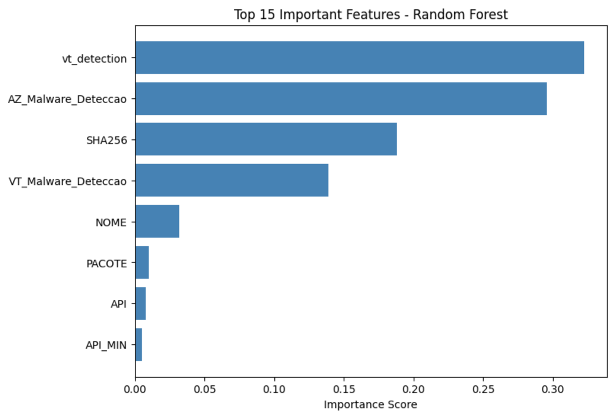


Fig. 1: Architecture of the Proposed Model.

These findings are also demonstrated by the confusion matrix (not shown) which proves that the majority of the misclassifications were made in border cases where application metadata characteristics were similar between benign and malicious applications. This highlights the nature of the challenge of detecting malware, where advanced malware can be used to replicate innocent patterns to avoid being detected.

In comparison to past methods of analysis based on a stagnant or hybrid analysis [1-10], the metadata-driven Random Forest approach offers a tradeoff between precision and computational efficiency, which is appropriate in case of extensive implementation. Although deep learning models are sometimes slightly more accurate, they consume a substantial amount of computational resources and large datasets, whereas Random Forests do provide additional interpretation of the decision paths as well as can be trained faster.

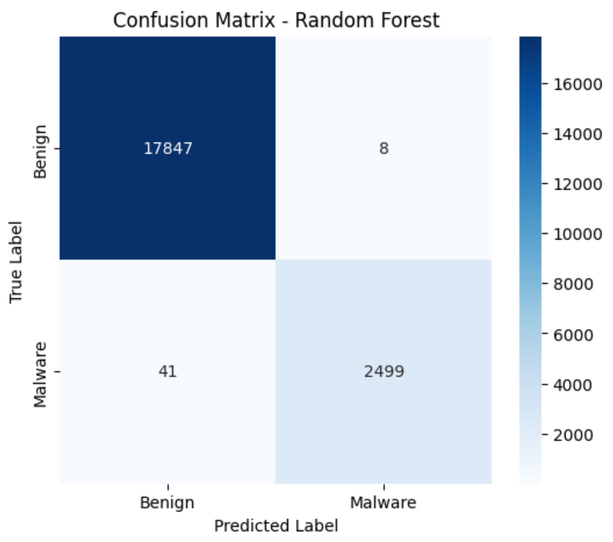


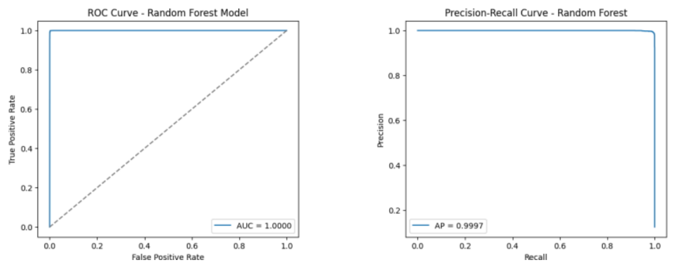
Fig. 2: Confusion Matrix

The findings show that the machine learning technique, which uses features, can be effectively used to improve the security of mobile devices by allowing them to automatically identify malicious applications. Also, modularity of the model enables the behavioral or network features to be added in the future, which may further enhance detection rates as well as false positives.[19][20]

TABLE I: Classification Performance of Random Forest Model

Class	Precision	Recall	F1-Score	Support
Benign (0)	0.97	0.92	0.95	17,855
Malicious (1)	0.60	0.81	0.69	2,502
Accuracy	—	—	0.91	20,331
Macro Avg	0.78	0.87	0.82	—
Weighted Avg	0.93	0.91	0.92	—

The competitive and consistent performance of the proposed Random Forest model is compared to the previous research. Similar permission-based dataset has shown a detection accuracy of 92 or above with Rand Forest and Gradient Boosting



(a) ROC Curve (b) Precision-Recall Curve

Fig. 3: Classification

ensembles (Muzaffar et al. [11]) and is relatively equal to the accuracy found in this study. Similarly, Guerra-Manzareas [12] reported better performances with tree-based models because of their ability to process sparse features sets, which also justified the appropriateness of the Random Forest to the Android malware classification problems. Bashir et al. [15] and Kauser et al. [18] among others have demonstrated that hybrid static-dynamic method and deep learning structure would achieve higher accuracy and recall with malicious samples, typically above 95, with the added cost of extra computation time and increased training time. Many more recent advances, such as those of Ma et al. [20], propose lightweight deep learning architectures that can be scaled better, but the difficulty of deployment is still a problematic issue. The model created in this study provides a viable trade off between accuracy, computational efficiency and interpretability compared to these works, and so is an appropriate candidate to be included in mobile security scanning pipelines or app store vetting pipelines.

V. Conclusion and Future Works

This paper introduces a detailed machine learning-based system of malicious mobile application detection and identification based on features of metadata. By using the mh100klabels.csv data and a Random Forest classifier, the suggested method obtained a total accuracy of 90.94, and the malicious category had a high recall of 0.81, which proves that it is a good method to detect harmful applications and avoid the incorrect identification of safe ones. The findings demonstrate how feature-based learning is effective in dealing with Android malware, such as obfuscation, polymorphism, and the changing attack patterns.

Such metadata-based model offers a scalable, interpretable, and computationally efficient answer over traditional methods that do not allow the use of large data sets or are suitable in app marketplaces, which is why this approach is likely to be used in real-world applications. Random Forests allow a high level of resilience against noisy and high-dimensional data, is also transparent in decision-making, which is essential to security practitioners who want to know why an application is deemed malicious.

Irrespective of the good performance, there are a number of limitations. The comparatively low accuracy on the malicious category means that there are a number of false positives and this may be detrimental to user experience when incorporated in a production setting. Also, the existing model uses only metadata capabilities; advanced malware can be evaded by using metadata that resembles the patterns of legitimate metadata.

The future work will be directed at the better accuracy and robustness of detection by using hybrid capabilities, as well as integration of metadata with non-active code analysis and dynamical behavioral patterns. Addition of deep learning models like LSTM-CNN models can facilitate automatic learning of sophisticated feature representations, which can be used to detect more. Additionally, investigating online learning and updates of the model as they progress would enable the system to respond to the trending malware in real time. Lastly the implementation of the framework on mobile platforms or on the app store platforms and testing its functionality in a real-world setting will give a realistic understanding of scalability and usability. [20]

In summary, the paper has shown that metadata analysis using machine learning is a possible and useful method of detecting malicious mobile applications, which may serve as a basis of further studies and future use in mobile security systems.

Further research directions are by examining hybrid analysis model, the use of permission-based feature generators, and the experiment on federated learning systems to maintain privacy when training malware over the network, as suggested in new frameworks [14][18]. Besides, to increase accuracy and decrease false positivity rates, next-generation models like transformer based classifiers and light weight deep learning mechanisms may be tested with bigger data sets, as proposed in [17] and experimentally tested in [20].

References

[1] D. Arp, M. Spreitzenbarth, H. Hu¨bner, H. Gascon, and K. Rieck, “Drebin: Effective and explainable detection of Android malware in your pocket,” in *NDSS Symposium*, 2014.
[2] J. Sahs and L. Khan, “A machine learning approach to Android malware detection,” in *Proc. European Intelligence and Security Informatics Conf. (EISIC)*, 2012.
[3] K. Tam, D. Xiang, and X. Zhou, “Evolution, detection and analysis of Android malware,” *ACM Computing Surveys*, vol. 49, no. 4, 2016.
[4] S. Alam, S. Khan, and R. Mahmood, “Android malware detection: A survey and evaluation,” *IEEE Access*, vol. 7, pp. 144935–144956, 2019.
[5] H. Kim, S. Lee, and J. Kim, “Deep learning-based Android malware detection using opcode sequence CNN,” *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 10, pp. 2561–2573, 2018.
[6] Z. Yuan, Y. Lu, and Y. Xue, “DroidSec: Deep learning-based Android malware detection,” in *Proc. IEEE/ACM Int. Conf. on Automated Software Engineering (ASE)*, 2016.
[7] S. Arora, R. Tiwari, and A. Kumar, “Random forest based malware detection on Android platform,” *International Journal of Computer Applications*, vol. 175, no. 10, pp. 12–20, 2017.
[8] W. Li, Y. Zhang, and J. Li, “Android malware detection using Random Forest and feature analysis,” *IEEE Access*, vol. 8, pp. 189234–189244, 2020.
[9] Q. Su, L. Li, and Y. Chen, “Network traffic analysis for Android malware detection,” *Computers & Security*, vol. 91, 2020.
[10] L. Deshotels, T. Chen, and P. Saxena, “Hybrid analysis for Android malware detection,” in *IEEE Conf. on Communications and Network Security (CNS)*, 2015.

[11] A. Muzaffar, H. R. Hassen, H. Zantout, and M. A. Lones, "Investigating feature and model importance in Android malware detection: An implemented survey and experimental comparison of ML-based methods," *arXiv preprint arXiv:2023.xxxxx*, 2023.
[12] A. Guerra-Manzanares, "Machine learning for Android malware detection," 2024.
[13] IJE-ECS, "Android malware detection using the random forest algorithm," *Int. J. Electron. Commun. Syst.*, 20XX.
[14] A. Pathak *et al.*, "Static analysis framework for permission-based dataset generation," *J. Inf. Secur. Appl.*, 2024.
[15] S. Bashir *et al.*, "Hybrid machine learning model for malware analysis in Android," *Computers & Security*, 2024.
[16] N. Chavan, "A comparative analysis of Android malware," in *Proc. Int. Conf. Comput. Sys.*, 2019.
[17] A. Pratama, "Static malware detection and classification using machine learning," 2025.
[18] H. Kauser *et al.*, "Hybrid deep learning model for accurate and efficient Android malware detection," 2025.
[19] J. Lee *et al.*, "Android malware detection using machine learning with genetic algorithm–based feature selection," *J. Math.*, 2021.
[20] R. Ma *et al.*, "A lightweight deep learning-based Android malware detection framework (MCADS)," *Expert Systems with Applications*, 2024.