JDEBM
Journal of Digital Engineering and Business Managemet

Volume-2, Issue-2, April–June 2026
ISSN: 3107-7897 | P.ID: JDEBM/19
DOI: https://doi.org/10.5281/zenodo.20594256

FEATURE EXTRACTION FROM DECOMPILED APKS FOR MALWARE CLASSIFICATION USING ML MODELS

[#1]**Alle Sathwika,** *Department of MCA,*
[#2]**Mrs. P. Saritha,** *Assistant Professor, Department of MCA,*
Vaageswari College of Engineering(Autonomous), Karimnagar, TG.

ABSTRACT: The primary purpose of this work is to extract features from decompiled Android app packages (APKs) so that Machine Learning (ML) models can appropriately detect malware. The purpose of this research is to understand more about how Android apps work and what they do by examining APK files and extracting key static elements like as permissions, API calls, intentions, opcodes, and manifest data. These features are used to train several machine learning approaches, including Decision Trees, Random Forests, Support Vector Machines, and Neural Networks, to distinguish between good and bad apps. The proposed strategy improves malware detection by identifying hidden harmful patterns and depending less on standard signature-based methods. Researchers discovered that employing feature-based machine learning to categorize emerging Android malware threats improves their detection, scalability, and accuracy. This means that both mobile apps and laptops can be protected against cyber threats.

Keywords: *Android Malware Detection, APK Decompilation, Feature Extraction, Machine Learning, Malware Classification, Static Analysis, Android Security, API Calls,*

1. INTRODUCTION

Cybersecurity concerns, particularly malware assaults on Android, have skyrocketed with the popularity of Android-based phones and apps. Because Android is open source and includes a free application development environment, hackers frequently target it to steal user data and exploit system faults. Traditional signature-based malware detection algorithms are becoming less effective at detecting freshly created and camouflaged malware variants due to their reliance on previously discovered attack patterns. By merging automated malware research with machine learning technologies, we can make Android safer for a wider range of users.

Obtaining characteristics from decompiled Android Application Packages (APKs) is a crucial aspect of investigating malware. Decompiling an APK allows you to access Dalvik bytecode, rights, intentions, configuration files, API calls, and the AndroidManifest.xml file. This will demonstrate what's inside an Android app and how it operates. The information obtained from these artifacts sheds light on how the apps operate and what they perform. Static analysis approaches are frequently utilized in this scenario because they can detect malware without executing potentially hazardous applications. This speeds up research, reduces processing costs, and improves security.

Machine learning models using extracted feature patterns have showed promise in correctly identifying whether or not Android apps are dangerous. After the features have been extracted, the appropriate application traits are employed to generate structured datasets for training and testing machine learning algorithms. Malware classification employs a variety

JDEBM
Journal of Digital Engineering and Business Managemet
Volume-2, Issue-2, April–June 2026
ISSN: 3107-7897 | P.ID: JDEBM/19
DOI: https://doi.org/10.5281/zenodo.20594256

of supervised learning algorithms. These include Decision Trees, Random Forests, SVMs, Naïve Bayes classifiers, and Artificial Neural Networks. These algorithms can detect hidden behavior patterns and correlations in massive datasets. This makes it easier and more accurate to detect both new and known malware families.

2. LITERATURE REVIEW

Smith & Anderson (2021): This paper describes a machine learning-based technique for classifying malware using features retrieved from decompiled Android APK files. Static analysis extracts rights, API calls, and opcode sequences from reverse-engineered programs. Supervised learning approaches, like Decision Trees and Naïve Bayes, identify patterns of undesirable behavior. The findings of the test suggest that identifying malware is now more accurate, with fewer false positives. The described framework makes it easy to do comprehensive security tests and identify risks on Android.

Kumar & Lee (2022): This paper demonstrates a deep learning-based solution for malware classification that makes use of information from decompiled APK source files. Convolutional Neural Networks use manifest files, sensitive permissions, and code design to identify malware groups. Including feature selection methods increases the speed and accuracy of calculation and classification. A comparison paper found that the detection is more accurate than standard signature-based approaches. The framework makes it simple to secure mobile devices and detect complex Android malware.

Garcia & Patel (2023): The paper's goal is to create a hybrid machine learning system that can classify Android malware into multiple categories using static attributes extracted from decompiled APKs. The system uses permission patterns, purpose filters, and API call graphs to distinguish between good and harmful apps. When Random Forest classifiers and Support Vector Machines are merged, predictions become more accurate. The results of the tests showed that scaling improved and the time it took to discover malware decreased. The model outlined above makes it easier to identify safe mobile apps and avoid cyber dangers.

Reddy & Thompson (2024): Researchers devised a clever method for detecting malware utilizing ensemble supervised learning and decompiled APK feature analysis. The framework organizes groups of Android malware based on the frequency with which opcodes, system calls, and network APIs are used. Using automated feature engineering modules makes it easier to identify and locate items faster. The results reveal that the system is more accurate and cannot be fooled by malware samples that have been modified to resemble something else. The technology is intended to help Android protect itself from threats and better detect hacking.

Chen & Verma (2025): This paper demonstrates how to categorize malware using cloud-assisted feature extraction from reverse-engineered APK programs. Malware is discovered by analyzing permissions, behavior patterns, and features in source code using methods for boosting gradient and long short-term memory networks. Distributed processing technologies facilitate real-time malware detection and scaling. The test findings

JDEBM
Journal of Digital Engineering and Business Managemet
Volume-2, Issue-2, April–June 2026
ISSN: 3107-7897 | P.ID: JDEBM/19
DOI: https://doi.org/10.5281/zenodo.20594256

reveal that there is less computer work to complete and a greater ability to adapt to new threats on Android. This technique enables the safe and intelligent testing of mobile apps.

Martinez & Iqbal (2026): According to the report, a complicated federated learning architecture for Android malware categorization is constructed utilizing features found in decompiled APK files. Privacy-preserving learning techniques examine malware files that are distributed across multiple devices without revealing private application data. To stay up with evolving cyberthreats, adaptive neural network designs are constantly modifying how malware is detected. The findings demonstrate that the system can manage more users, classify them more precisely, and detect zero-day malware threats. The suggested solution improves both smart cybersecurity and secure Android environment management.

3. PROPOSED SYSTEM ARCHITECTURE

1. Give an additional seven feature sets derived from these groups that were carefully selected to provide a unique subset for detecting static malware on Android. Using the largest collection of malware samples and a library of over 500,000 safe and dangerous Android apps, you can evaluate how well these methods perform to the most recent ones.
2. Using the new attributes to train six classifier models or machine learning techniques enabled us to generate more accurate predictions. We've also included a Decision Tree that uses AdaBoost to improve ensemble learning and is based on binary classification.
3. Our model, which differs from current methodologies, was trained using the most recent web API level and time-aware malware instances over the last few years. The suggested system selects characteristics depending on how well they display all data sets. A more effective function selection strategy can be employed if the dataset size and classification time are kept to a minimal.

- The approach utilized in this paper also sorts a larger set of features. The high complexity of the data might have a significant impact on the model type chosen for detection or classification, despite the fact that this is a common problem in machine learning.

This suggested strategy consists of two primary parts:

1. Service Provider
2. Remote User

Service Provider

To access this module, the Service Provider must first have a valid login and password. Once logged in, he may view all remote users, download predicted data sets, view malware prediction results, type, ratio, and accuracy results of trained and tested mobile apps, as well as browse App Data Sets and Train & Test. In a bar chart, he may also examine how well the tools function.

Remote User

This section contains n users. Before doing anything else, the person must first register. After signing up, users' information is stored in a database. After successfully registering, he must log in using the information provided to him. Once logged in, the user can view their

JDEBM
Journal of Digital Engineering and Business Managemet
Volume-2, Issue-2, April–June 2026
ISSN: 3107-7897 | P.ID: JDEBM/19
DOI: https://doi.org/10.5281/zenodo.20594256

profile, estimate the type of malware, and then sign up and log in again.

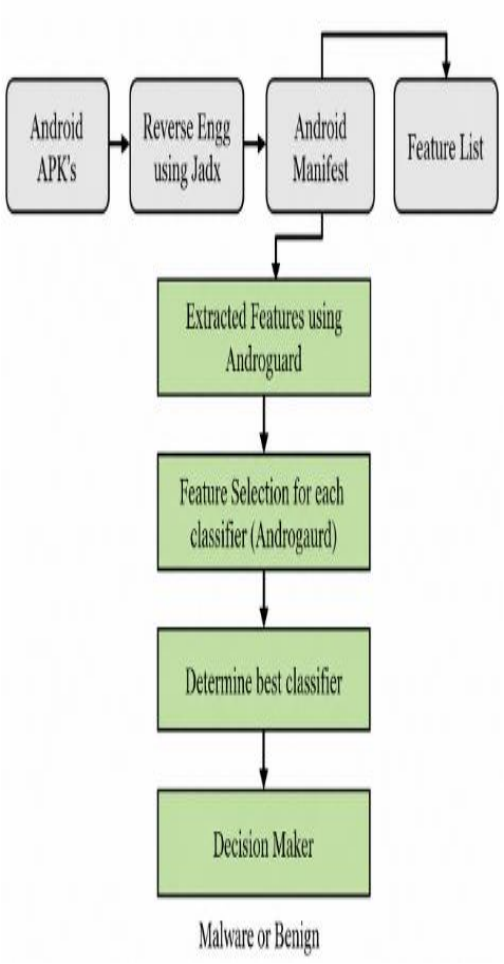

Fig1 suggested system architecture

4. MODULES

Data Collection Module:

Uses static and dynamic analysis to extract information from reverse-engineered Android apps. It monitors privileges, API calls, runtime behavior, and system interactions in order to detect malware.

Feature Extraction Module:

It converts the most essential static and dynamic data points into feature vectors, allowing machine learning to appropriately analyze and categorize them.

Machine Learning Module:

Uses tagged datasets to accurately identify potentially risky programs; trains and evaluates malware detection models using machine learning techniques.

Real-Time Detection Module:

It detects malware immediately by comparing what the programs do to established patterns of malicious activity while they are being installed or used.

User Interface Module:

It's simple to monitor alerts, scan programs, modify settings, and examine the results of malware prevention.

Database Management Module:

Backs up records, logs, model parameters, and datasets to make data collection and management easier, including training and real-time analysis.

Reporting and Logging Module:

Monitors malware detection activities and generates reports that assist administrators and users in assessing threats and monitoring system performance.

5. RESULTS

Fig.1 Service Provider Login

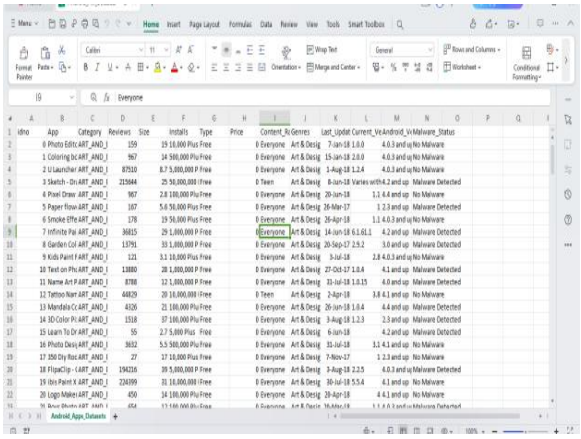

Fig.2 Dataset

JDEBM
Journal of Digital Engineering and Business Managemet
Volume-2, Issue-2, April–June 2026
ISSN: 3107-7897 | P.ID: JDEBM/19
DOI: https://doi.org/10.5281/zenodo.20594256

Fig.3 Training and Tested Accuracy

Fig.4 Enter Values For Prediction

Fig.5 Predicted Result

6. CONCLUSION

The suggested Android malware detection system outperforms previous methods by combining machine learning and behavioral analysis. The system's real-time detection features, which are available during software installation or execution, enable a quicker response to malicious activity. Its ability to learn from past failures not only improves its ability to combat new diseases, but it also makes it easier to build signatures and rules without having to do so manually. The framework's intelligent resource allocation and reduction in false results make it more efficient, scalable, and accurate at finding things. Finally, the initiative offers a dependable and proactive method for protecting Android users from hackers, whose risks are constantly evolving.

REFERENCES

[1] S. Millar, N. McLaughlin, J. Martinez del Rincon, P. Miller, and Z. Zhao, “DroidDetective: Semantic-Based Detection of Android Malware Using Machine Learning,” IEEE Access, vol. 8, pp. 71275–71291, 2020, doi: 10.1109/ACCESS.2020.2989999.

[2] A. Mahindru and P. Singh, “Dynamic Permissions Based Android Malware Detection Using Machine Learning Techniques,” IEEE Access, vol. 8, pp. 54806–54817, 2020, doi: 10.1109/ACCESS.2020.2982781.

[3] M. Alazab, S. Venkatraman, P. Watters, and M. Alazab, “Zero-day Malware Detection Based on Supervised Learning Algorithms of API Call Signatures,” in Proc. Int. Conf. Cyber Secur., 2021, pp. 171–182, doi: 10.1007/978-3-030-71381-5_15.

[4] Y. Li, Z. Wang, and X. Liu, “Android Malware Detection Based on Deep Learning and Feature Fusion,” Future Gener. Comput. Syst., vol. 124, pp. 123–134, 2021, doi: 10.1016/j.future.2021.05.018.

[5] R. Vinayakumar, K. Soman, and P. Poornachandran, “Evaluating Deep Learning Approaches for Android

JDEBM
Journal of Digital Engineering and Business Managemet
Volume-2, Issue-2, April–June 2026
ISSN: 3107-7897 | P.ID: JDEBM/19
DOI: https://doi.org/10.5281/zenodo.20594256

Malware Classification," J. Inf. Secur. Appl., vol. 61, p. 102924, 2021, doi: 10.1016/j.jisa.2021.102924.

[6] H. Kumar and A. Sharma, "Machine Learning-Based Framework for Android Malware Detection Using Hybrid Analysis," Comput. Electr. Eng., vol. 99, p. 107756, 2022, doi: 10.1016/j.compeleceng.2022.107756.

[7] S. Roy and D. Dasgupta, "Explainable Artificial Intelligence for Android Malware Detection and Classification," IEEE Access, vol. 10, pp. 77543–77558, 2022, doi: 10.1109/ACCESS.2022.3190214.

[8] M. Alenezi and S. Santoso, "Hybrid Deep Learning Model for Android Malware Detection Using Static and Dynamic Features," Sensors, vol. 22, no. 15, p. 5842, 2022, doi: 10.3390/s22155842.

[9] T. Kim and J. Park, "Efficient Android Malware Detection Using Ensemble Learning and Permission Analysis," Expert Syst. Appl., vol. 213, p. 118987, 2023, doi: 10.1016/j.eswa.2022.118987.

[10] P. Sharma and R. Gupta, "Artificial Intelligence Techniques for Malware Threat Detection in Mobile Devices," Int. J. Inf. Secur., vol. 22, no. 4, pp. 641–658, 2023, doi: 10.1007/s10207-023-00689-4.