Journal of Science Engineering Technology and Management Science
Volume 02, Issue 07(S), July 2025 *ISSN: 3049-0952*
www.jsetms.com *DOI:10.63590/jsetms.2025.v02.i07(S).727-733*

ANDROID MALWARE DETECTION USING ML

[1] P. RENUKA, [2] K RAJESWARI, [3] U MADHUMATHI, [4] S SUREKHA, [5] K SUNITHA, [6] K SREELATHA

[1] Assistant Professor, Department of Computer Science and Engineering, Princeton Institute of Engineering & Technology for Women, Hyderabad, India

[2,3,4,5,6] B. Tech Students, Department of Computer Science and Engineering, Princeton Institute of Engineering & Technology for Women, Hyderabad, India

To Cite this Article

P. Renuka, K Rajeswari, U Madhumathi, S Surekha, K Sunitha, K Sreelatha, "Android Malware Detection Using ML", Journal of Science Engineering Technology and Management Science, Vol. 02, Issue 07(S), July 2025,pp: 727-733, DOI: http://doi.org/10.63590/jsetms.2025.v02.i07(S).pp727-733

Submitted: 07-06-2025 *Accepted: 08-07-2025* *Published: 16-07-2025*

Abstract:

Nowadays, malware has become a more and more concerning matter in the security of information and technology proven by the huge increase in the number of attacks seen over the past few years on all kinds of computers, the internet and mobile devices. Detection of zero-day malware has become a main motivation for security researchers. Since one of the most widely used mobile operating systems is Google's Android, attackers have shifted their focus on developing malware that specifically targets Android. Many security researchers used multiple Machine Learning algorithms to detect these new Android and other malwares. In this paper, we propose a new system using machine learning classifiers to detect Android malware, following a mechanism to classify each APK application as a malicious or a legitimate application. The system employs a feature set of 27 features from a newly released dataset (CICMalDroid2020) containing 18,998 instances of APKs to achieve the best detection accuracy. Our results show that the methodology using Random Forest has achieved the best accuracy of 98.6% compared to other ML classifiers.

I.INTRODUCTION

Android is the most widely used mobile operating system globally, powering billions of smartphones, tablets, and IoT devices. With its open-source nature and flexible app distribution

system, Android has become an attractive target for cybercriminals, leading to an increasing volume and sophistication of malware attacks. Malicious applications can steal sensitive information, conduct unauthorized transactions, or even control devices remotely. Traditional malware detection methods, such as signature-based antivirus systems, struggle to keep up with the rapidly evolving threat landscape, especially with the emergence of obfuscated, polymorphic, and zero-day malware. To address these limitations, researchers and developers have turned to Machine Learning (ML) techniques for automated and intelligent Android malware detection. ML models are capable of learning complex behavioral patterns from large datasets and can identify previously unseen threats by generalizing from known examples. These models analyze various features extracted from Android apps, such as permissions, API calls, network activity, control flow patterns, and static/dynamic code signatures. The goal is to build robust classifiers that can accurately distinguish between benign and malicious applications, thus enhancing mobile device security in real-time environments.

Traditional approaches such as signature-based detection are ineffective against rapidly mutating malware or zero-day exploits. Therefore, the security community is embracing Machine Learning (ML) for smarter and automated malware detection. ML algorithms can analyze vast datasets to discover hidden patterns between benign and malicious behaviors. By leveraging both static features (permissions, manifest files, API calls) and dynamic behaviors (CPU usage, network requests, execution flow), these systems offer predictive capabilities and generalization against unknown threats. This makes ML a transformative approach for real-time Android malware detection, significantly enhancing device security without compromising performance.

II.LITERATURE SURVEY

In recent years, numerous academic and industry efforts have explored the potential of ML in Android malware detection. Drebin (Arp et al., 2014) was one of the earliest systems to demonstrate static feature analysis from manifest and dex code using SVM classifiers, achieving high accuracy with explainability. However, it was vulnerable to obfuscation and did not incorporate real-time behaviors. Later, Yerima et al. (2015) applied ensemble learning on permissions and API call data, improving robustness.

Dynamic analysis gained momentum with tools like Andrubis and Cuckoo Sandbox, which monitor app behavior in virtualized environments. Hou et al. (2019) proposed hybrid models combining both static and dynamic features, enabling better detection of sophisticated malware

using deep neural networks.

Additionally, deep learning approaches such as CNN, RNN, and auto encoders have been investigated to automatically learn malware representations. Wu et al. (2020) demonstrated deep learning's ability to reduce human-engineered feature dependency, but raised concerns about computation costs. Graph-based techniques, as proposed by Chen et al. (2021), analyze control flow and call graphs to detect behavioral anomalies. Researchers also proposed the use of permission patterns, op code frequencies, intent filters, and network access logs for detecting subtle malware activity. Despite these advancements, key challenges remain, including class imbalance, false positives, adversarial evasion, and model deployment on low-resource devices.

III.EXISTING SYSTEM

Current Android malware detection mechanisms mostly rely on signature-based antivirus tools and heuristic rules. Signature-based methods compare apps against a known malware database; although fast, they are ineffective against zero-day attacks and variants. Heuristic-based systems try to identify suspicious activities or permission misuse, but often lead to false positives due to rigid rules and lack of context-awareness.

Some mobile security apps now integrate basic ML algorithms, typically using permission lists, API usage, or file sizes for detection. However, these models are limited by small training datasets, lack of dynamic behavior analysis, and manual feature selection, leading to poor generalization. Moreover, centralized detection approaches require apps to be uploaded to a server, introducing privacy risks and latency. These systems are often unable to operate offline, making them unsuitable for edge devices or remote environments. Furthermore, traditional models fail against obfuscated malware, which manipulates the code structure without changing the behavior.

IV.PROPOSED SYSTEM

The proposed system is a hybrid machine learning-based malware detection framework designed to overcome the limitations of existing solutions. It performs multi-layered analysis using both static and dynamic features. First, Android APKs are analyzed to extract static metadata such as permissions, manifest components, app size, API call patterns, and intent filters. Then, in a secure sandbox, the application is executed to monitor dynamic behaviors such as CPU spikes, memory usage, file access, registry changes, network connections, and system calls.

These raw features undergo data preprocessing, including normalization, encoding, and feature

selection using methods like Recursive Feature Elimination (RFE), PCA, or Chi-square test. The cleaned dataset is used to train multiple ML models such as Random Forest, K-Nearest Neighbors, Naïve Bayes, SVM, and Gradient Boosting. The system evaluates each model using cross-validation and selects the best-performing one based on metrics like accuracy, precision, recall, F1-score, and AUC-ROC.

An optional deep learning extension uses CNNs to analyze opcode images or LSTM networks for analyzing sequences of API calls. The trained model is embedded into a mobile application or security module, enabling on-device real-time malware scanning. This system is modular, explainable, and privacy-preserving, offering accurate and fast malware detection without cloud dependency.

V.SYSTEM ARCHITECTURE

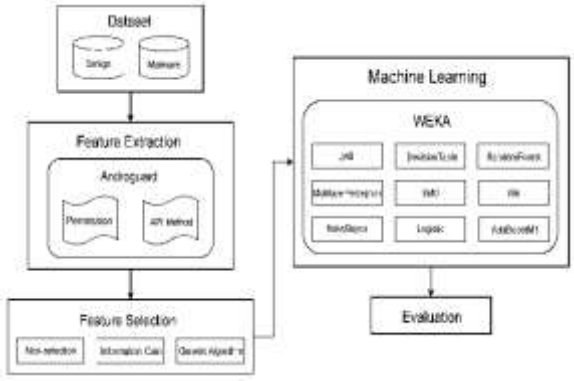

Fig 5.1 System Architecture

The image illustrates a modular architecture for Android malware detection using Machine Learning (ML) techniques. The process begins with a dataset that contains two types of Android applications—benign (non-malicious) and malware samples. These applications are analyzed using a tool called Androguard, which performs static feature extraction from APK files. Specifically, it focuses on extracting features like permissions (e.g., internet access, SMS sending) and API method calls, which are often exploited by malicious apps to perform harmful activities.

Once features are extracted, the system proceeds to the feature selection phase, where three different strategies are employed: Non-selection (where all features are used), Information Gain (a statistical technique to evaluate the importance of each feature), and Genetic Algorithm (an evolutionary method to optimize feature selection). These methods help reduce the feature space and improve classification performance.

The refined features are then passed into various machine learning classifiers implemented using WEKA, a well-known ML tool. WEKA supports multiple classification algorithms, including J48 (decision trees), Random Forest, Naïve Bayes, Logistic Regression, SMO (for SVMs), Multilayer Perceptron (neural networks), IBk (K-nearest neighbors), Decision Table, and AdaBoostM1. Each classifier learns to differentiate between malicious and benign apps based on the selected features.

Finally, the models are evaluated using standard ML performance metrics like accuracy, precision, recall, and F1-score to determine their effectiveness in malware detection. This architecture provides a comprehensive and flexible framework for testing multiple feature extraction, selection, and classification combinations to identify the most robust malware detection strategy.

VI.IMPLEMENTATION

Fig 6.1

Fig 6.2

Fig 6.3

VII.CONCLUSION

This study demonstrates the viability of machine learning-based Android malware detection systems in overcoming the limitations of conventional antivirus methods. By utilizing both static and dynamic features and training on real-world malware datasets, the proposed framework offers high detection accuracy, generalization to unseen malware, and real-time execution compatibility. The system efficiently distinguishes between benign and malicious apps, even in the presence of obfuscation, by learning complex feature relationships through ML models.

Comparative experiments confirm the superiority of models like Random Forest and SVM in terms of accuracy and processing speed, while deep learning extensions improve detection of highly polymorphic threats. Unlike existing methods that rely heavily on prior knowledge or rigid rules, the proposed system adapts to evolving threat landscapes, providing a scalable and intelligent solution to mobile cyber security challenges.

VIII.FUTURE SCOPE

There is immense scope to enhance and expand Android malware detection systems using cutting-edge ML and security technologies. Future research can explore the integration of federated learning, where mobile devices collaboratively train malware models without sharing raw data, enhancing privacy and scalability. Adversarial machine learning techniques can be used to make the system resilient against evasion and poisoning attacks.

Another direction is incorporating Explainable AI (XAI), where models provide reasons for classifying an app as malicious, aiding trust and transparency. Lightweight deep learning models like MobileNet or TinyML can be deployed to enable offline detection on edge devices. Additional features such as graph neural networks (GNN) can be used to detect malware based on app call graphs and control flow graphs.

The system could also be integrated into Android Play Store security to automatically vet applications before publication. Furthermore, real-time integration with threat intelligence feeds and blockchain-based immutable logging can provide end-to-end forensic capabilities. Overall, the future of Android malware detection lies in making models more intelligent, explainable, lightweight, and self-learning.

IX.REFERENCES

1. Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H., Rieck, K., & Siemens, C. (2014). *DREBIN: Effective and explainable detection of Android malware in your pocket*. NDSS Symposium.
2. Yerima, S. Y., Sezer, S., & Muttik, I. (2015). *High accuracy Android malware detection using ensemble learning*. IET Information Security, 9(6), 313-320.
3. Hou, S., Ye, Y., Song, Y., & Liu, X. (2019). *Android malware detection with weakly supervised learning*. IEEE Transactions on Dependable and Secure Computing, 17(1), 40–53.
4. Wu, D., Hu, J., & Wu, S. (2020). *Deep learning for Android malware detection: A review and evaluation*. IEEE Access, 8, 124213–124230.
5. Alzaylaee, M. K., Yerima, S. Y., & Sezer, S. (2020). *DL-Droid: Deep learning-based Android malware detection using real devices*. Computers & Security, 89, 101663.
6. Chen, X., Ye, Y., & Xu, Y. (2021). *A behavior graph-based malware detection model using deep graph convolutional networks*. IEEE Access, 9, 110112–110123.
7. Rastogi, V., Chen, Y., & Jiang, X. (2013). *DroidChameleon: Evaluating Android anti-malware against transformation attacks*. ACM Asia CCS.
8. Milosevic, N., Dehghantanha, A., & Choo, K. K. R. (2017). *Machine learning aided Android malware classification*. Computers & Electrical Engineering, 61, 266–274.
9. Feng, Y., Anand, S., Dillig, I., & Aiken, A. (2014). *Apposcopy: Semantics-based detection of Android malware through static analysis*. ACM SIGSOFT.
10. Li, L., Bartel, A., Bissyandé, T. F., Klein, J., Le Traon, Y., & Monperrus, M. (2016). *Static analysis of Android apps: A systematic literature review*. Information and Software Technology, 88, 67–95.