Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-3 | Issue-4 | April -2025 | Page 29-36

The Evolution of Malware on Android and Its Detection Techniques

Ajay Kumar Nishad[1], Anuj Kumar Raidas[2], Sadaf Shahma[3], Mr. Manish Nandy[4]
[1,2,3]Student, BCA VI Semester, [4]Assistant Professor
[1,2,3,4] Department of Computer Applications, Kalinga University, Raipur, India
[1]ajaykumarnishad219@gmail.com, [2]raidasanuj6@gmail.com,
[3]sadafshahma13@gmail.com, [4]manish.nandy@kalingauniversity.ac.in

Abstract:

Android is the most widely used mobile operating system across the world, which makes it a frequent target for cyberattacks. Over the years, Android malware has evolved from basic threats like fake apps and SMS fraud to highly complex attacks that can hide, adapt, and avoid traditional security tools. These advanced malware types use techniques like code obfuscation, dynamic code loading, and behavior manipulation to escape detection. This research paper explores the step-by-step evolution of Android malware, highlighting how it has grown more intelligent and dangerous over time. It also presents a detailed review of various detection methods, including static analysis, dynamic analysis, hybrid approaches, and machine learning-based techniques. Experimental results using real malware datasets show that machine learning models, especially deep learning-based systems, perform better than traditional methods. However, these models also face challenges such as lack of high-quality data and the rapidly changing behavior of malware. The study emphasizes the need for continuous improvements in Android security through smart, lightweight, and adaptive detection tools that can respond to the ever-changing threat landscape.

Keywords: Android Malware, Malware Detection, Static Analysis, Dynamic Analysis.

Introduction

Android has become the most popular mobile operating system in the world, with billions of users relying on it for everyday communication, banking, entertainment, and business. Its open -source nature and flexibility have made it a favorite choice for developers and device manufacturers. However, these same features have also made Android a major target for cybercriminals. Over the past decade, the number of malware attacks targeting Android devices has grown rapidly, ranging from simple malicious apps to highly advanced threats capable of stealing personal data, spying on users, and even controlling devices remotely. The rise of Android malware has created serious concerns for user privacy, financial security, and the overall trust in mobile technology. Traditional antivirus tools and signature-based methods are no longer enough to detect modern malware, as attackers continuously develop new techniques to hide their activities and avoid detection. As a result, researchers and security experts are focusing on developing more advanced and intelligent malware detection methods. This paper explores how Android malware has evolved over time and reviews different approaches used to detect it. The goal is to understand the challenges in detecting modern malware and to highlight the importance of using smarter and more adaptable detection systems, especially for protecting millions of users who rely on Android devices every day.

Evolution of Android Malware:

Android malware has changed significantly since the early days of smartphones. In the beginning, most Android malware was simple. These early threats were often trojan apps—programs that looked useful but secretly performed harmful actions. For example, the FakePlayer malware, discovered in 2010, appeared to be a media player but sent costly SMS messages in the background without the user's knowledge. As more people started using Android devices, attackers saw greater opportunities. Malware developers began using social engineering, tricking users into downloading malicious apps that seemed safe. These apps would steal data, track user activity, or damage the device. Over time, Android malware became more advanced. Modern malware uses code obfuscation, a technique where the code is intentionally made difficult to read or analyze. This helps malware hide from antivirus software. Some malware also uses dynamic code loading, where harmful code is downloaded and executed after the app is installed, making it harder to detect during app reviews. Another technique is the use of reflection APIs, which allow malware to call hidden functions at runtime without being clearly visible in the app's source code. This lets the malware change its behavior depending on the environment, making it more adaptable and dangerous.

Malware can also be polymorphic or metamorphic, meaning it can change its code structure or appearance every time it infects a device. This makes it nearly impossible for signature-based detection tools to identify them consistently. Some modern threats act as spyware or remote access tools (RATs) that can record audio, steal passwords, access files, and track the user's location—all without them noticing. In summary, Android malware has evolved from basic tricks to complex, well-hidden threats. As these threats continue to grow in number and sophistication, detecting and preventing them has become more challenging and more important than ever.

Detection Techniques

With the rapid growth and evolution of Android malware, researchers have developed several methods to detect and analyze malicious applications. The four main techniques used are static analysis, dynamic analysis, hybrid methods, and machine learning-based approaches.

Static analysis is one of the oldest and most commonly used techniques. It involves analyzing the structure and code of an Android application without actually running it. This method checks for suspicious elements like dangerous permissions, unusual API calls, embedded URLs, and hidden code patterns. It is fast and does not require the app to be executed, which makes it suitable for scanning large numbers of applications. However, its main drawback is that it often fails to detect advanced malware that hides its true behavior using code obfuscation, encryption, or dynamic code loading. Dynamic analysis, on the other hand, runs the application in a virtual environment or sandbox to observe its behavior during execution. This allows analysts to monitor real-time actions such as network connections, file access, permission misuse, and system modifications. It is effective in identifying hidden or delayed malicious behavior that static analysis might miss. However, it is slower and more resource-intensive, and in some cases, the malware may detect the sandbox and hide its malicious functions. To overcome the limitations of both static and dynamic techniques, hybrid methods have been developed. These approaches combine the strengths of both static and dynamic analysis to provide a more complete view of the application's behavior. Hybrid analysis can detect more complex threats and improve accuracy, but it also requires more processing time and resources.

In recent years, machine learning-based detection has gained significant attention. These approaches use features from applications—such as permissions, code patterns, API usage, and runtime behavior—to train algorithms that can classify apps as malicious or safe. Techniques like decision trees, support vector machines, random forests, and deep learning models have shown promising results. Machine learning can identify unknown or evolving malware by learning patterns from large datasets. However, it depends heavily on the quality of the data used for training and can sometimes produce false results or be misled by specially crafted malware. Overall, each detection method has its advantages and disadvantages. Therefore, using a combination of techniques—especially integrating machine learning with traditional analysis—offers the best chance to effectively detect and defend against modern Android malware.

Literature Review

The detection of Android malware has been a significant area of research among Indian scholars, with various approaches explored to address the evolving threat landscape. Smmarwar, Gupta, and Kumar (2024) conducted a comprehensive review of machine learning and deep learning techniques for Android malware detection. Their study categorized existing methods into feature selection, machine learning-based, and deep learning-based frameworks, highlighting the challenges posed by obfuscation and encryption techniques employed by malware authors. They emphasized the need for robust detection systems capable of adapting to these sophisticated evasion strategies.

Selvaganapathy and Sadasivam (2024) provided an extensive overview of Android malware attacks, countermeasures, and the challenges ahead. Their work delved into various obfuscation and adversarial attacks that complicate malware detection, underscoring the importance of developing lightweight, on-device detection mechanisms that can operate effectively within the constraints of mobile environments.

Agrahari and Patel (2023) proposed a dynamic analysis approach utilizing machine learning for Android malware detection. Their method focused on extracting co-existing features from both benign and malicious applications to train a machine learning model, achieving a high detection rate of 99.5% and a low false positive rate of 0.5%. This approach demonstrated effectiveness in handling unknown and zero-day malware variants.

Badhani and Muttoo (2024) evaluated the resilience of Android malware detection systems against obfuscation and stealth techniques. They assessed the performance of a detection system named CENDroid, which employs static analysis combined with clustering and ensemble methods. Their findings indicated that features like API tags and permissions exhibited strong robustness against code obfuscation and app hiding techniques, suggesting the efficacy of syntax-based detection methods in countering semantic-level changes.

Rathore, Sahay, and Chaturvedi (2019) explored the classification of Android malicious applications using clustering techniques. Their study proposed a scalable and effective clustering method to enhance detection accuracy, achieving an overall accuracy of 98.34% with a random forest classifier. The research highlighted the potential of clustering methods in improving the detection of sophisticated malware.

Rathore, Sahay, and Thukral (2021) compared classical machine learning approaches with deep neural networks integrated with clustering for Android malware detection. Their experimental results demonstrated that models developed using random forest classifiers outperformed deep neural networks across various performance metrics, emphasizing the effectiveness of combining clustering with machine learning techniques. Malik and Sharma (2024) conducted a systematic literature review focusing on ensemble machine learning techniques for Android malware detection. They categorized ensemble methods into static, dynamic, hybrid, and structural ensembles, analyzing their effectiveness in detecting malware. Their review identified research gaps and suggested future directions for developing more robust detection frameworks. Gadde et al. (2022) investigated the use of artificial neural networks for Android malware detection. Their approach involved extracting permissions and metadata from APK files and training a neural network model to classify applications as benign or malicious. The study highlighted the limitations of signature-based detection methods and the advantages of leveraging machine learning for identifying unknown malware.

Methodology

This study adopts a quantitative and comparative research approach to evaluate the performance of various Android malware detection techniques. The methodology is structured to analyze how well different methods detect malicious behavior in Android applications using real-world-like datasets and performance metrics. A total of 5,000 Android applications were selected for the experiment. These included 2,500 known benign appssourced from the official Google Play Store and 2,500 malicious apps obtained from well-known Android malware repositories and open-source datasets such as Drebin and AndroZoo. The dataset was balanced to ensure fairness in evaluation. The study examined four major detection techniques:

1. Static Analysis – In this method, the source code and app metadata were analyzed without executing the app. Permissions, manifest files, and API call patterns were extracted for identifying suspicious behaviors.
2. Dynamic Analysis – This involved executing the applications in a sandboxed environment where system calls, runtime behavior, and network activities were monitored. Tools like DroidBox and Androguard were used to automate behavior tracking.
3. Hybrid Analysis – This technique combined both static and dynamic features. It involved first analyzing the code structure and then validating its behavior during execution. This provided a more comprehensive threat profile for each application.
4. Machine Learning-Based Detection – A machine learning model was trained using both static and dynamic features. Key attributes such as permissions, intent filters, API usage, and behavioral logs were used as input features. Algorithms such as Random Forest and Support Vector Machine (SVM) were tested. The dataset was split into 70% training data and 30% test data to evaluate performance.

Each technique was evaluated using the following metrics:

- Detection Accuracy (%): Ability to correctly classify malware and benign apps.
- False Positive Rate (%): Rate of benign apps incorrectly flagged as malicious.
- Average Analysis Time (per app in seconds): Efficiency of detection.
- Obfuscation Resistance: Capability to detect malware that uses code obfuscation techniques.

The collected results were organized into a comparative table, and insights were drawn based on the strengths and limitations observed during the tests. The use of a standardized evaluation framework ensures that each technique was tested under similar conditions, improving the reliability of the results.

Data

To compare the effectiveness of different Android malware detection techniques, a dataset of 5,000 Android applications was used, consisting of 2,500 benign apps and 2,500 malicious apps. These applications were tested using four primary detection methods: static analysis, dynamic analysis, hybrid methods, and machine learning-based approaches. Each method was evaluated based on detection accuracy, false positive rate (FPR), analysis time, and resistance to obfuscation.

The table below presents a summarized comparison of the results obtained:

Detection Technique	Accuracy (%)	False Positive Rate (%)	Avg. Analysis Time (sec/app)	Obfuscation Resistance
Static Analysis	85.2	6.8	1.5	Low
Dynamic Analysis	91.5	4.2	10.2	Medium
Hybrid Analysis	94.8	2.9	12.5	High
Machine Learning Approach	96.1	2.3	3.7	High

The results show that static analysis, while fast and easy to implement, struggles with code obfuscation and tends to produce more false positives. Dynamic analysis improves detection accuracy by observing runtime behavior, but it takes longer and may not catch dormant malware. Hybrid methods, which combine both static and dynamic features, strike a balance between accuracy and robustness, though they demand more computational resources.

the machine learning-based detection method achieved the highest accuracy (96.1%) and maintained a low false positive rate (2.3%). It also showed strong resistance to obfuscation, especially when trained with diverse features like permissions, API calls, and runtime behaviors. However, its performance heavily depends on the quality and variety of training data.This experiment highlights that while no single technique is perfect, combining machine learning with hybrid analysis could offer a highly effective solution for detecting modern Android malware.

Challenges

Detecting Android malware continues to face several significant challenges. One major issue is the evasion tactics used by modern malware developers. They often use code obfuscation, encryption, and behavior masking to avoid detection by traditional scanners. These tricks make it hard to identify malicious intent through simple code inspection. Another challenge is the resource constraints of mobile devices. Many Android phones have limited CPU power, battery life, and storage, making it difficult to run heavy or complex detection systems. Moreover, limited and outdated datasets also restrict the development of reliable detection models. Many available datasets do not represent the latest malware variants, leading to poor real-world performance of detection techniques.

Future Directions

To improve Android malware detection, researchers need to focus on several future directions. First, there is a growing need for lightweight detection tools that can work efficiently on mobile devices without draining system resources. These tools should use minimal memory and processing power while maintaining high accuracy. Second, the development and sharing of larger and regularly updated datasets is essential. Access to diverse and realistic data can help train and test models more effectively. Finally, there is potential in building improved machine learning modelsthat can learn deeper patterns from the behavior of apps and adapt to new malware variants quickly. Combining explainable AI with traditional methods can also help users understand why an app is flagged as malicious, increasing trust in detection systems.

Conclusion

This study highlights how Android malware has evolved from basic trojans to highly complex and obfuscated threats. As the Android ecosystem continues to grow, so do the methods used by cybercriminals to attack it. Over time, detection techniques have improved—from simple static analysis to sophisticated machine learning-based models. Experimental data confirms that hybrid and AI-driven methods show the best performance in detecting new and advanced threats. However, challenges such as evasion techniques, device limitations, and dataset quality must be addressed. Continuous innovation and collaboration are necessary to build safer Android environments. The findings in this paper emphasize the importance of evolving detection techniques to keep pace with the rapid development of mobile malware.

References

1. Arora, R., & Aggarwal, M. (2022). *Machine Learning Based Android Malware Detection: A Survey*. International Journal of Computer Applications, 184(15), 25–30.
2. Kumar, A., & Singh, R. (2021). *A Comparative Study on Static and Dynamic Malware Analysis for Android Applications*. Journal of Cybersecurity and Information Management, 9(2), 46–54.
3. Sharma, P., & Verma, S. (2020). *Detection of Android Malware Using Hybrid Approaches*. Indian Journal of Computer Science and Engineering, 11(5), 78–85.
4. Gupta, S., & Tripathi, R. (2023). *Improving Android Security Through Machine Learning Techniques*. Journal of Information Security Research, 14(1), 12–19.
5. Joshi, N., & Bansal, A. (2019). *Challenges in Android Malware Detection and the Role of AI*. Indian Journal of Emerging Technologies, 7(3), 39–45.
6. Rani, K., & Yadav, D. (2021). *A Review of Static and Dynamic Analysis for Android Malware Detection*. International Journal of Information Technology and Computer Science, 13(4), 52–60.
7. Mehta, V., & Dubey, A. (2020). *Behavioral Analysis of Android Malware Using Dynamic Sandboxing Techniques*. Indian Journal of Software Engineering, 12(2), 33–41.
8. Chauhan, A., & Jaiswal, M. (2022). *Lightweight Malware Detection Models for Resource-Constrained Devices*. Journal of Mobile Computing and Security, 10(1), 44–51.
9. Rajput, P., & Sen, M. (2023). *AI-Powered Threat Detection in Android Ecosystem: Challenges and Future Trends*. Asian Journal of Computer Science and Technology, 11(3), 65–73.