ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 13, Issue 07, July 2026

A Survey of Dynamic Analysis Approaches for Android Malware Detection

[1] Ashok Kumar Mamidi, [2] Dr. Chitra P

[1] [2] Department of Computer Science & Engineering, SRM Institute of Science and Technology (SRMIST), Kattankulathur, Tamil Nadu, India

Email: [1] ashok.mamidi@hotmail.com, [2] Chitrap1@srmist.edu.in

***Abstract*— Android is a growing platform with applications that have spurred a rise in threat of malware attacks, making Android security an important topic for study. Static analysis techniques are not effective against modern malware that uses code obfuscation, code encryption, packing and dynamic code loading. Dynamic analysis has proven to be a good method of detecting Android malware in runtime by observing the behavior of applications and detecting malicious activities that might not be visible in static code. This paper presents a comprehensive survey of dynamic analysis approaches for Android malware detection. Different dynamic analysis methods such as sandbox analysis, system call analysis, API call analysis, taint analysis, behavioral analysis and machine learning based are discussed and compared. The study also looks at some of the major dynamic analysis frameworks and recent deep learning-based Android malware detection models that have been developed. In addition, the benefits, constraints, and problems with dynamic analysis, including anti-emulation methods, partial execution coverage, high execution time and scalability issues, are reviewed. Finally, novel research directions such as Explainable Artificial Intelligence, Federated Learning, Cloud-based analysis and Adversarially Robust Detection Models are highlighted. The survey offers valuable insights about the current progress and future possibilities in Android malware detection using dynamic analysis techniques.**

***Index Terms*— Android Malware, Dynamic Analysis, Malware Detection, Mobile Security, Behavioral Analysis, Machine Learning, Deep Learning, Sandboxing, Cybersecurity.**

I. INTRODUCTION

Android is the most popular mobile operating system globally for its open-source code, versatility, and rich application base. Android devices have made a significant impact in the world of mobile applications with their rising popularity in many fields like communication, banking, healthcare, education and ecommerce. This widespread use has resulted in Android being a common target for cybercriminals, which has led to a quick surge in Android malware attacks and security threats [1]. Android malware includes various types of malicious software such as spyware, ransomware, banking Trojans, adware, and botnets. These threats aim to gain access to sensitive data, engage in unauthorized financial transactions, track user activities, and breach device security [2].

Table I: Major Android Malware Categories and Their Impact

Malware Type	Description	Impact on Users
Trojan	Malware disguised as a legitimate application	Unauthorized access and data theft
Spyware	Secretly monitors user activities and collects information	Privacy leakage and surveillance
Ransomware	Encrypts files or locks devices until payment is made	Data loss and financial damage
Adware	Displays excessive or unwanted advertisements	Poor user experience and performance degradation
Banking Malware	Targets banking and financial applications	Theft of credentials and financial fraud
Botnet Malware	Turns infected devices into remotely controlled bots	Remote attacks and unauthorized network activities

A. Android Malware and Its Impact on Users

Android malware is a major threat to user privacy and the security of businesses, as it can steal sensitive data and disrupt the devices of users. Privacy violations can result in financial losses and malicious applications can obtain personal information like contacts, messages, location information and banking information [3]. Common types of malwares that are able to attack user information and online transactions include Banking Trojans and spyware. Apart from stealing data, Android malware can also conduct unauthorized activities like sending premium SMS, downloading malwares, accessing device resources without consent of user. The consequence of these actions could be higher service costs, lower device performance, higher battery usage, and system instability [4]. Some malware variants also track user activities, gather browsing information and pass the sensitive information to remote servers, which poses additional security risks [5]. Android malware doesn't just affect individual users; it can affect

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 13, Issue 07, July 2026

enterprises, too, as infected mobile devices can serve as an access point to corporate networks and valuable business data. Android mobile devices have become vital in daily life, and effective and efficient malware detection and prevention mechanisms are now a must to safeguard users from the emerging threats in the cyber world [6], [7].

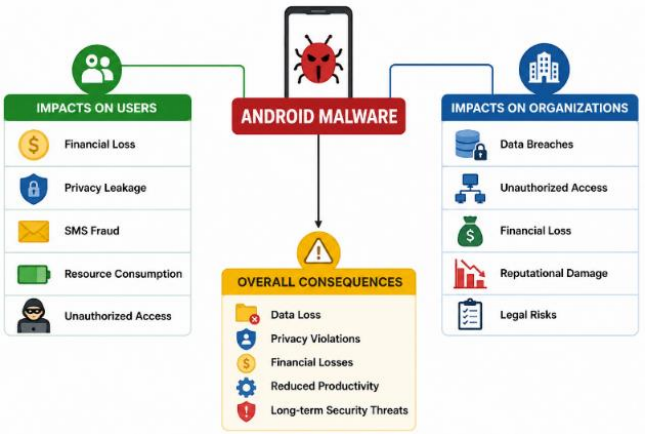

Figure 1. Major Impacts of Android Malware on Users and Organizations

B. Need for Dynamic Analysis

Malware detection was originally based on static analysis, which involves analyzing application code without running the application. While static methods are efficient to a certain extent, they are of little use against the modern malware that uses obfuscation, encryption, packing and dynamic code loading to hide malicious behavior [8]. The restrictions have given rise to dynamic analysis approaches which are more effective in the detection of malware. Dynamic analysis is the analysis of application behaviour in a controlled environment, e.g. sandboxes, emulators, or real devices, during runtime. It is able to record the execution level details such as system calls, API calls, modification in file system, activities done on memory and network traffic patterns [9]. Dynamic analysis is especially useful at capturing unknown or zero-day malware as malicious activity will reveal itself at runtime, even if the code is hidden or obfuscated [10]. There are few research attempts that have focused on operating system level monitoring and behavioral analysis of Android applications, such as TaintDroid, DroidBox and Andrubis [11]. They can assist to detect shadow IT activity, including data leakage, unauthorized SMS messages, privilege escalation, and malicious internet communications.

In recent years, machine learning and deep learning approaches have been combined with dynamic analysis to enhance the accuracy of malware detection by leveraging behavioral features like system call sequences, API invocation patterns, and network traffic behavior [12]. Such strategies have helped to identify more advanced malware variants and new attacks that haven't been seen before. However, dynamic analysis has its own problems, such as emulator detection, low runtime coverage, high computing complexity and scalability. Anti-analysis techniques like delayed execution, environment-aware triggering and anti-emulation techniques are increasingly being used by malware developers to bypass the detection systems [13]. To tackle this, hybrid solutions that integrate static and dynamic analysis techniques as well as AI-based classification techniques are being investigated to achieve robustness and defectiveness [14]. Going forward, there are several areas of research that require further investigation: cloud malware analysis, federated learning frameworks, and adversarial robust detection models to enhance Android security from new emerging threats [15].

C. Objectives

The main objective of this review paper is to analyze and understand different dynamic analysis approaches used for Android malware detection. The specific objectives are:

- Examine the rapid growth of Android malware and evaluate its impact on individual users, enterprises, and mobile ecosystems.
- Analyze the major Android malware detection approaches, with particular emphasis on dynamic analysis techniques and their effectiveness in identifying malicious behavior.
- Review widely used dynamic analysis frameworks, including sandbox-based environments, API and system call monitoring, taint analysis, and behavioral profiling methods.
- Compare machine learning and deep learning models employed in dynamic malware detection, highlighting their strengths, limitations, and detection performance.
- Identify current challenges and research gaps in dynamic analysis-based Android malware detection and discuss future directions for improving detection accuracy, scalability, and efficiency through advanced artificial intelligence techniques.

II. RELATED WORKS

Several researchers have proposed a wide range of dynamic analysis techniques for detecting Android malware, focusing on runtime behavior analysis, system-level monitoring, and automated sandboxing environments. Costa et al. [16] investigated the combination of static and dynamic analysis techniques for Android malware detection. Their study showed that the DroidFax static analysis tool detected a significant portion of malware samples, while FlowDroid taint analysis achieved performance comparable to dynamic analysis tools. The results indicate that integrating static analysis with mining sandbox approaches can improve the effectiveness of Android malware detection. Aldhafferi [17] suggested an Android malware detection framework that combines dynamic feature analysis with Support Vector Regression (SVR). The approach analyzes the runtime behavior of Android applications and applies machine learning techniques to distinguish between benign and

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 13, Issue 07, July 2026

malicious apps. The study demonstrated that dynamic behavioral features can effectively improve malware detection and outperform several traditional classification methods. De Lorenzo et al. [18] suggested VizMal, a visualization tool for Android malware analysis that highlights potentially malicious behavior within application execution traces. The tool assists security analysts in identifying and understanding malicious activities during runtime, thereby improving the efficiency of malware inspection and analysis. Ibrahim et al. [19] suggested an Android malware detection and classification framework that combines static analysis with a deep learning model. The approach extracts useful application features and employs a functional API-based architecture to distinguish between benign and malicious applications. The study demonstrated that deep learning techniques can effectively improve Android malware detection and classification performance. Jeon et al. [20] introduced Dynamic Analysis for IoT Malware Detection (DAIMD), a dynamic analysis framework for IoT malware detection that combines behavioral analysis with a Convolutional Neural Network (CNN). The approach monitors malware behavior in a cloud-based environment, extracts runtime activities, and converts them into images for classification. The study demonstrated that dynamic behavior analysis and deep learning can effectively detect both known and evolving IoT malware variants. Abdelwahed et al. [21] developed MalpMiner, a dynamic malware detection framework based on Answer Set Programming (ASP). The approach uses logical reasoning and API behavior analysis to identify malicious activities without prior training. However, its effectiveness depends on predefined rules, which may limit the detection of highly sophisticated malware. Haq et al. [22] developed a hybrid deep learning framework that combines CNN and Bi-LSTM models for Android malware detection. The approach effectively identifies complex malware behaviors, but its performance depends on large training datasets and high computational resources. Alzaylaee et al. [23] developed DL-Droid, a deep learning-based Android malware detection system that combines dynamic analysis with enhanced input generation techniques. The framework improves malware detection by analyzing application behavior during execution. However, the approach requires extensive runtime analysis and computational resources, which may increase detection overhead. Overall, existing literature strongly indicates that dynamic and behavior-based analysis techniques are more effective than traditional static methods in detecting modern Android malware. However, challenges such as evasion techniques, scalability limitations, and execution overhead remain open research problems that require further investigation.

Table II: Summary of Existing Dynamic Analysis Approaches for Android Malware Detection

Author	Method/Framework	Technique Used	Key Contribution	Limitation
Costa et al. [16]	DroidFax + FlowDroid	Static and Dynamic Analysis	Improved malware detection through combined analysis approaches	Requires integration of multiple analysis tools
Aldhafferi [17]	Dynamic Analysis with SVR	Machine Learning (SVR)	Utilized runtime behavioral features for malware classification	Performance depends on feature quality
De Lorenzo et al. [18]	VizMal	Visualization-Based Dynamic Analysis	Enhanced understanding of malicious runtime behaviors	Primarily supports analyst interpretation
Ibrahim et al. [19]	Deep Learning Framework	Static Analysis + Deep Learning	Improved malware detection and classification accuracy	Dependent on training data quality
Jeon et al. [20]	DAIMD	Dynamic Analysis + CNN	Detected IoT malware through behavioral image classification	Computationally intensive
Abdelwahed et al. [21]	MalpMiner	ASP-Based Dynamic Analysis	Detected malware using logical reasoning and API behavior analysis	Relies on predefined rules
Haq et al. [22]	CNN-BiLSTM Framework	Hybrid Deep Learning	Captured complex malware behavior patterns	Requires large datasets and computational resources
Alzaylaee et al. [23]	DL-Droid	Dynamic Analysis + Deep Learning	Improved detection through enhanced runtime behavior analysis	High runtime overhead

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 13, Issue 07, July 2026

III. DYNAMIC ANALYSIS TECHNIQUES

Dynamic analysis techniques are used to observe and check at runtime if an Android application displays malicious behavior. Dynamic analysis observes the behavior of an application during runtime, unlike static analysis which looks at source code or application packages without executing the code. It is especially useful for catching malware that incorporates code obfuscation, encryption or dynamic loading to evade traditional anti-virus detection techniques. There are many different dynamic analysis techniques that have been created for tracking system activities, network communications, API calls and file operations. Sandbox-based analysis, one of these, is widely used.

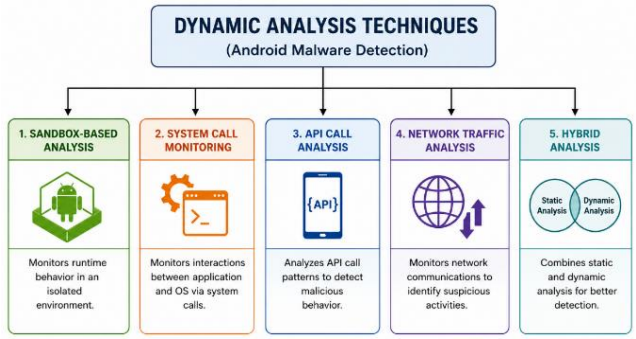

Figure 2. Major Dynamic Analysis Techniques for Android Malware Detection

A. Sandbox-Based Analysis

Sandbox-based analysis is the most popular type of dynamic analysis methods for detection of Android malware. This method involves running applications in a sandboxed environment that is separate, isolated and controlled, allowing for their behaviour to be monitored during execution without impacting the user or the device. The sandbox tracks files, networks, SMS messages, file permissions, access to sensitive information and other activities. Studying the patterns allows researchers to recognize malicious behavior that may not be apparent from the static code. Hu et al. [24] presented a dynamic taint analysis tool, TaintDroid, that monitors the movement of sensitive data in real-time as it flows through Android apps. The system tracks the access and transmission of private data like contacts, location, device identifiers etc., and catches when it is leaking privacy. They showed that many Android apps sent sensitive information about users without adequate permission, wherein runtime monitoring proved to be effective. Likewise, Bhan et al. [25] created DroidBox, a dynamic analysis framework which extends the monitoring capabilities of Android by monitoring application actions, including file operations, network usage, information leakage and cryptographic usage. DroidBox produces comprehensive behavioral reports, which helps researchers to identify malicious activities carried out while the application is running. In addition, Mills and Legg [26] developed an automated platform for the analysis of malware called Andrubis, which executes malware on a sandbox and monitors it extensively for behavior. It looks at file system changes, communication with the network, API calls, and system interaction to detect malicious apps on a massive scale. The outcomes showed that they could effectively identify a variety of malware families in the automated sandboxes and gain useful information on malware behavior. Sandbox-based analysis has several benefits: Suspicious applications can be run in a safe environment and obfuscated or encrypted malware can be identified while it is running in the sandbox, which is not possible when analyzing using source code inspection. Today, however, many malwares contain emulator-detection mechanisms that are able to sense virtual machines and limit their malicious behavior, making sandbox-based systems less effective [27]. Furthermore, an incomplete execution path and/or reduced user interaction in sandbox environments can make it impossible to trigger some malicious actions. Even with these problems, sandbox-based analysis is still considered as a basic and popular method for detecting malicious software on Android devices, as it can provide insights on the in-the-wild behavior of malicious apps.

B. System Call Monitoring

System call monitoring is a technique in dynamic analysis which analyzes the interaction between Android applications and the operating system kernel while running. System calls like open (), read (), write (), connect (), and socket () are useful for understanding an application's behavior as they are direct requests for system resources and services. With the objective of detecting malware, Torres et al. [28] presented a system call activity monitoring and behavioural pattern analysis-based malware detection system during the application run. Their study showed that there are different sequences of system calls in malicious apps and benign apps. In the same way, Zhan et al. [29] have used behavioral analysis techniques with system calls monitoring to detect abnormal activities relating to Android malware. To analyse the system call patterns of applications, researchers have used techniques such as sequence analysis and N-gram modelling, which are widely used in the classification of malicious and benign applications. One of the key benefits of system call monitoring is that it gives the lowest level of visibility into application behavior, making it hard for malware to obscure itself from the system call monitor. This does lead to high data volume, however, which leads to high computation overhead and complex feature space which may impact on the detection efficiency.

C. API Call Analysis

API call analysis is about tracking the interactions of the application at the application level using the Android Application Programming Interfaces (APIs). Android Apps use APIs to interact with device resources and conduct

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 13, Issue 07, July 2026

activities, hence the way they utilize APIs can expose malicious targets. Popular APIs that are analyzed and targeted include APIs for SMS, location, telephony and device administration. Xu et al. [30] proposed DroidAPIMiner, a malware detection system that relies on the frequency and the pattern of API calls that the Android applications utilize. Their results showed that sensitive APIs commonly used for data theft and unauthorized accesses to the systems are often called by malware. Further, Meena and Prabha [31] developed a system dubbed DREBIN that used API related behavioral features for classification of Android Applications. Frequency and machine learning algorithms that examine the sequences of API calls to determine whether they are malicious or part of a legitimate app, are some of the methods used for detection. The features of application-level behavior captured in an API call analysis are meaningful for machine learning-based malware detection systems and effective for capturing application-level behavior.

D. Network Traffic Analysis

Another useful dynamic analysis tool examines network traffic between an Android app and an external server, which is known as network traffic analysis. Numerous malware families rely on the Internet for communication with command and control (C2) servers, downloading malicious payloads, or sending stolen data. Krishnan et al. [32] have proposed a network-based approach to malware detection by analyzing the communication patterns to detect suspicious behavior. Malware detection involves clues like suspicious outbound traffic, command and control, data exfiltration, and DNS tunneling. While network traffic analysis is very useful in detecting malware that communicates remotely, the growing proliferation of encrypted communication channels is a problem for conventional inspection techniques. However, network-based monitoring is still crucial for today's Android malware detection systems.

E. Hybrid Analysis

Hybrid analysis is a combination of static and dynamic analysis techniques that take the best aspects of both methods and reduces the limitations of each. While static analysis allows for swift inspection of application code and permissions, dynamic analysis can be used to observe the behavior and execution patterns of the application at runtime. To enhance the classification accuracy, Taher et al. [33] presented a hybrid malware detection framework, combining static features and dynamic behavioral information. Likewise, Johny et al. [34] have shown that the fusion of various sources of features contributes to improving the malware detection accuracy and decreasing the false positive rates. The hybrid methods have the advantages of comprehensive feature extraction, high detection accuracy, and resistance to code obfuscation methods. With the continuous growth of Android malware, hybrid analysis is a potential way of building powerful and intelligent malware detection systems.

IV. COMPARATIVE ANALYSIS

Comparative analysis is crucial to assess the effectiveness of various dynamic analysis techniques for detecting Android malware. Various approaches such as sandbox analysis, system call monitoring, API call analysis, network traffic analysis, and hybrid analysis have been proposed in the literature. The accuracy, computational load, scalability, real-time detection and malware evasion resistance of these techniques varies.

A. Performance Comparison of Dynamic Analysis Techniques

The comparison shows that hybrid analysis can detect the most accurately because it combines multiple behavioral features. System call monitoring is also regarded as a good detection capability, capturing low level interactions in the system. Analyses of API calls and network traffic provide lower detection accuracy but have benefits of scalability and deployment efficiency.

Table III: Comparative Analysis of Dynamic Analysis Techniques

Technique	Computational Overhead	Scalability	Real-Time Detection	Evasion Resistance
Sandbox Analysis	Medium	Medium	Low	Medium
System Call Monitoring	High	Medium	Medium	High
API Call Analysis	Low	High	High	Medium
Network Traffic Analysis	Medium	High	Medium	Medium
Hybrid Analysis	Very High	Medium	Medium	Very High

B. Detection Accuracy Analysis

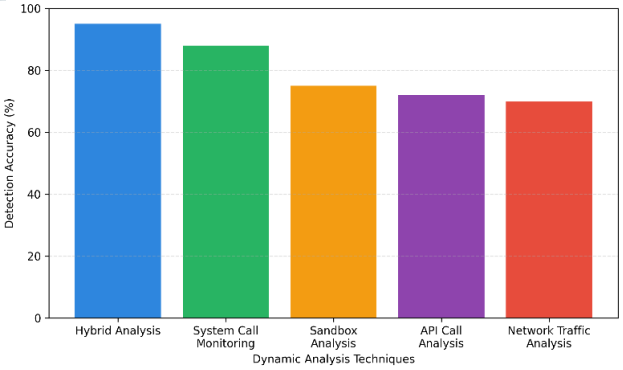

Figure 3. Detection Accuracy Comparison of Dynamic Analysis Techniques

Figure 3 compares the detection performance of different dynamic analysis techniques for Android malware detection. Hybrid analysis shows the best overall performance and system call monitoring offers a good detection capability.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 13, Issue 07, July 2026

Sandbox, API call analysis and network traffic analysis are moderately effective. The comparison shows that the detection performance of the combination of multiple behavioral features is generally better than the detection performance of any single feature.

C. Resource Requirement Analysis

There are significant differences in the computational resources for each technique, especially for dynamic analysis. API call analysis is a lightweight approach that works well when deployed on a large scale and in real time, due to its low processing needs. On the other hand, the monitoring of system calls and hybrid analysis demand extra compute resources as a result of the extensive information gathering and analysis. This indicates that it is crucial to consider the effectiveness of the detection along with the efficiency of the operations when choosing a suitable detection framework for malware.

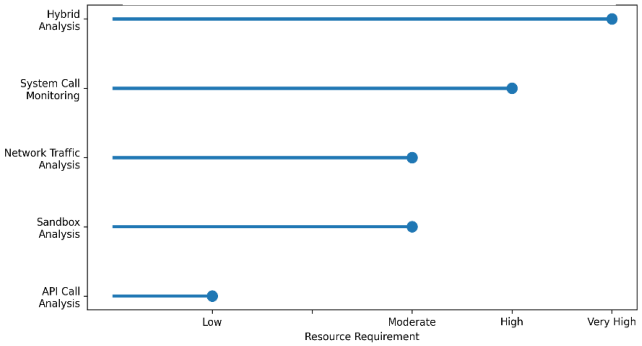

Figure 4. Comparative Resource Requirements of Dynamic Analysis Techniques

D. Strength-Weakness Assessment

There are significant differences in the computational resources for each technique, especially for dynamic analysis. API call analysis is a lightweight approach that works well when deployed on a large scale and in real time, due to its low processing needs. On the other hand, the monitoring of system calls and hybrid analysis demand extra compute resources as a result of the extensive information gathering and analysis. This indicates that it is crucial to consider the effectiveness of the detection along with the efficiency of the operations when choosing a suitable detection framework for malware.

Table IV: Strength-Weakness Matrix of Dynamic Analysis Techniques

Technique	Major Strength	Major Limitation
Sandbox Analysis	Secure execution environment	Vulnerable to environment-aware malware
System Call Monitoring	Detailed behavioral visibility	Resource-intensive monitoring
API Call Analysis	Efficient feature extraction	Limited behavioral depth
Network Traffic Analysis	Detection of communication-based threats	Challenges with encrypted traffic
Hybrid Analysis	Comprehensive behavioral coverage	Increased system complexity

V. MACHINE LEARNING IN DYNAMIC ANALYSIS

The ability of machine learning to learn malicious behavior from dynamic execution data is a crucial feature making it an important tool in the anti-malware toolbox for Android. Machine Learning (ML) algorithms can identify new versions of malware not encountered before by identifying its behavior over time. Dynamic analysis produces features like system call sequences, API calls, network traffic patterns, and resource usage statistics, and adds them to a training set for classification models.

Table V: Comparison of Recent Machine Learning and Deep Learning Frameworks for Android Malware Detection.

Author	Model/Framework	Algorithm Used	Key Contribution	Limitations
Eskandari et al. [35]	Andromaly	Anomaly Detection	Behavioral malware detection using runtime features	High false-positive rate and dependence on quality of behavioral features
Albin Ahmed et al. [36]	ML-based Framework	RF, SVM, NB, DT	Comparative analysis of machine learning algorithms	Performance varies with dataset quality and requires feature engineering
Herrera-Silva and Hernández-Álvarez [37]	DroidScribe	SVM, Random Forest	Dynamic behavior-based malware classification	Limited effectiveness against previously unseen malware variants
Feng et al. [38]	DroidDetector	Deep Learning	Automatic malware feature learning	High computational complexity and large training data requirements
Bala et al. [39]	CNN-Based Detection	CNN	High-accuracy malware classification	Requires extensive training data and may suffer from overfitting

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 13, Issue 07, July 2026

Author	Model/Framework	Algorithm Used	Key Contribution	Limitations
Owoh et al. [40]	LSTM Framework	LSTM	Sequential behavior analysis for malware detection	Increased training time and computational overhead for long behavior sequences

The combination of machine learning and dynamic analysis greatly enhances the detection accuracy of malware and helps to eliminate the need for manually created signatures. One of the earliest machine learning-based dynamic detection frameworks was Andromaly, proposed by Eskandari et al. [35]. The framework tracked the application behaviour like CPU usage, memory usage, battery usage, network activities, etc., and used Anomaly detection methods to detect malicious applications. The results showed that human behavior monitoring and machine learning can achieve effective distinction between the malware and benign applications. Albin Ahmed et al. [36] compared four machine learning algorithms: Random Forest (RF), Support Vector Machine (SVM), Naïve Bayes (NB) and Decision Trees on the detection of Android malware. They explained that its study demonstrated that Random Forest provided better classification performance as it is suitable for high dimensional feature space and complex behavioral patterns. Likewise, Herrera-Silva and Hernández-Álvarez [37] used dynamic behavioral attributes collected from application run-time and applied machine learning classifiers to enhance the accuracy of malware identification.

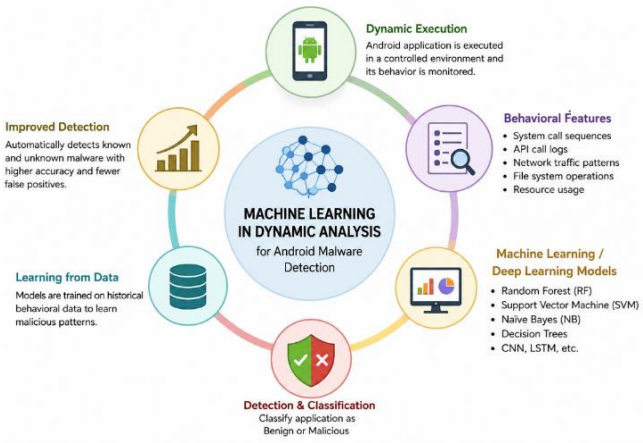

Figure 5. Machine Learning-Based Dynamic Analysis for Android Malware Detection

Deep learning was also integrated to further improve dynamic analysis of malware. Feng et al. [38] proposed DroidDetector, an automatic learning framework based on deep learning to detect the characteristics of malware. In a similar way, Bala et al. [39] used Convolutional Neural Networks (CNNs) to classify Android malware and showed that deep learning models can outperform traditional machine learning techniques for better accuracy in malware detection. Owoh et al. [40] used Long Short-Term Memory (LSTM) networks to analyse the sequences of API calls and system calls for the identification of advanced malicious activities that change over time. In a machine learning based dynamic analysis, the features extracted play an important role in the success of the analysis. These are typically: sequences of system calls, logs of calls to specific APIs, patterns of network traffic, file system activity, and patterns of resource usage. These capabilities deliver a wide range of visibility into application behavior and enhance machine learning models capability to detect malicious activities. There are some problems with machine learning strategies, however, even though they are effective. They need significant labeled data sets to learn models and are often costly. In addition, there have been recent studies on the dangers of adversarial attacks, in which the malware author changes the behavior in order to avoid detection. However, machine learning and deep learning still present the most promising technologies in the process of building intelligent, scalable and adaptive Android malware detection frameworks.

Table VI: Comparative Analysis of Android Malware Detection and Dynamic Analysis Studies

Author(s)	Framework	Techniques	Key Contribution	Limitation
Cinar and Kara [1]	Mobile Security Review	Survey Analysis	Comprehensive review of mobile security threats and challenges	No experimental validation
Yadav et al. [2]	IoT & Android Malware Framework	Defensive Mechanisms	Analysis of malware in Android and IoT environments	Limited focus on dynamic behavior analysis
Kavin et al. [3]	Secure Data Acquisition	Machine Learning	Detection of malicious	Not specifically designed

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 13, Issue 07, July 2026

Author(s)	Framework	Techniques	Key Contribution	Limitation
	Model		activities in mobile communication networks	for Android malware
Saudi et al. [4]	iOS Malware Analysis	Survey Analysis	Comprehensive review of iOS malware techniques	Focused on iOS rather than Android
Aslan et al. [5]	Intelligent Malware Detection System	ML + Behavioral Analysis	Cloud-based intelligent malware detection	High computational overhead
Bayazit et al. [6]	Android Security Review	Learning Models	Overview of Android malware detection methods and challenges	Limited experimental evaluation
Tanveer et al. [7]	Malware-SeqGuard	LSTM + GRU	Detection of evolving Android malware families	Requires large training datasets
Alkhateeb et al. [8]	Malware Packer Detection	Static Analysis	Identification of malware packers using feature engineering	Ineffective against runtime obfuscation
Maghanaki et al. [9]	IoT Malware Dataset	Dataset Generation	Development of multiclass IoT malware dataset	Dataset-specific applicability
Alhaidari et al. [10]	ZeVigilante	ML + Sandboxing	Zero-day malware detection framework	Resource-intensive analysis
Park et al. [11]	A-Pot	Container-Based Analysis	Android malware analysis platform	High deployment complexity
Meng et al. [12]	AppAngio	API Audit Logs	Context-aware Android behavior analysis	Large audit-log generation
Kim et al. [13]	Anti-Analysis Study	Behavioral Analysis	Investigation of malware anti-analysis techniques	Primarily analytical study
Joshi et al. [14]	Hybrid AI IDS	Hybrid AI	Improved intrusion detection accuracy	Not Android-specific
Rajan et al. [15]	Cloud Security Framework	Distributed Security Model	Secure cloud-based storage framework	Not focused on malware detection
Costa et al. [16]	Mining Sandbox	Static + Dynamic Analysis	Improved Android malware identification	Increased analysis complexity
Aldhafferi [17]	Dynamic Analysis Framework	SVR	Dynamic feature-based malware detection	Sensitive to feature quality
De Lorenzo et al. [18]	VizMal	Visualization Techniques	Visualization of dynamic malware behavior	Limited automated detection capability
Ibrahim et al. [19]	Malware Detection Framework	Deep Learning	Automated Android malware detection	High training requirements
Jeon et al. [20]	DAIMD	CNN	Dynamic IoT malware detection	Computationally expensive
Abdelwahed et al. [21]	MalpMiner	Dynamic Analysis	Runtime malware activity detection	Runtime overhead
Haq et al. [22]	DL-Based Framework	CNN + Bi-LSTM	Robust Android malware detection	Requires large datasets
Alzaylaee et al. [23]	DL-Droid	Deep Learning	Real-device malware detection	High resource consumption
Hu et al. [24]	SAMLDroid	Static Analysis + ML	Detection of privacy leakage risks	Limited to privacy-related threats
Bhan et al. [25]	DLCDroid	Dynamic Code Analysis	Analysis of dynamically loaded code	Complex implementation

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 13, Issue 07, July 2026

Author(s)	Framework	Techniques	Key Contribution	Limitation
Mills and Legg [26]	Sandbox Reconfiguration	Anti-Evasion Analysis	Detection of malware evasion triggers	Increased sandbox complexity
Suo et al. [27]	ARAP	Runtime Analysis	Detection of anti-runtime-analysis code	High computational cost
Torres et al. [28]	Behavior Analysis Framework	Feature Engineering	Behavioral malware detection	Feature engineering dependency
Zhan et al. [29]	Behavior Sequence Model	Anomaly Detection	Adversarially robust malware detection	Complex model training
Xu et al. [30]	Behavioral-Level Detection	GCN	Graph-based malware detection	Complex graph construction
Meena and Prabha [31]	Zero-Shot Learning Framework	Zero-Shot Learning	Detection of unseen malware families	Lower accuracy for rare malware classes
Krishnan et al. [32]	MUD-Based Framework	Behavioral Profiling	IoT device security framework	Not Android-specific
Taher et al. [33]	DroidDetectMW	Hybrid Intelligent Model	Enhanced Android malware detection	Increased model complexity
Johny et al. [34]	Deep Learning Fusion	CNN Fusion	Malware detection using visual features	High computational requirements
Eskandari et al. [35]	Passban IDS	Anomaly Detection	Intelligent intrusion detection for IoT	Not Android-focused
Albin Ahmed et al. [36]	Ransomware Detection Framework	RF, SVM, DT, NB	Android ransomware detection	Dataset dependency
Herrera-Silva and Hernández-Álvarez [37]	Dynamic Dataset Framework	Machine Learning	Dynamic ransomware detection	Limited generalization capability
Feng et al. [38]	Performance-Sensitive System	Deep Learning	Efficient mobile malware detection	Resource-intensive training
Bala et al. [39]	Malware Image Classifier	CNN + Transfer Learning	High-accuracy malware classification	Risk of overfitting
Owoh et al. [40]	API Sequence Analysis Framework	GRU-GAN	API sequence-based malware detection	High model complexity

Table VI presents a comparative analysis of recent approaches in Android malware detection, dynamic analysis, machine learning and deep learning. This table presents the methodologies, contributions and limitations of some representative studies on Android malware identification methods focusing on the improvements of the dynamic analysis, behavioral monitoring, machine learning, deep learning and graph-based learning, and anomaly detection techniques.

VI. CHALLENGES IN DYNAMIC ANALYSIS FOR ANDROID MALWARE DETECTION

Despite the significant improvements made by dynamic analysis in the field of Android malware detection, there are still a number of drawbacks that restrict its efficacy. The modern malware employs anti-analysis and anti-emulation techniques, one of the major challenges. There are many malware samples that can tell if they are running in a sandbox or emulator and alter or hide actions to avoid detection. This functionality lowers the trustworthiness of traditional dynamic analysis systems [13], [26], [27]. One of the major challenges is due to poor code execution coverage. Dynamic analysis can only detect behaviors that are initiated during the running of the application. If specific malicious functions are not triggered, during the analysis period, then they will not be detected. The attackers tend to use logic bombs, time bombs or user- or event-triggered mechanisms to hide malicious activities, which makes a full behavioral analysis challenging [25] and [27]. Use of resources is also a high concern. Dynamic analysis demands running and monitoring of applications, and these need considerable computational power, memory, storage and analysis time. Scalability is an important problem with large-scale malware analysis systems as the number of Android malware applications continues to

increase [21] [22]. Real-time malware detection poses other challenges. Dynamic analysis requires adequate execution time to be able to make a reliable decision. This time gap can enable malicious activity to be carried out before an effective response from detection mechanisms can be implemented [22], [23]. Additionally, encrypted communication is becoming more common with Android malware, making network traffic analysis more difficult. In many cases, malware takes advantage of HTTPS, VPN tunnels and encrypted command and control (C2) traffic to evade security monitoring and detection. This makes traditional network-based detection methods ineffective at detecting data exfiltration and remote-control attacks [21], [28]. Furthermore, the constant change of malware variants, attacks to machine learning models, and the absence of the standard benchmark datasets remain significant challenges for research. Future efforts towards building more accurate, scalable and resilient Android malware detection systems should address these issues [29], [30], [31].

VII. FUTURE DIRECTIONS

The rapid evolution of Android malware necessitates the development of advanced and adaptive detection techniques. Future research is expected to focus on integrating artificial intelligence (AI) and deep learning models with dynamic analysis to improve the detection of sophisticated and previously unseen malware variants. Advanced architectures such as Long Short-Term Memory (LSTM) networks, Graph Neural Networks (GNNs), and Transformer-based models can effectively capture complex behavioral patterns from runtime data and enhance detection accuracy. Another promising direction is the adoption of Explainable Artificial Intelligence (XAI) techniques. While deep learning models provide high detection performance, their decision-making processes are often difficult to interpret. XAI can improve transparency by explaining why an application is classified as malicious, thereby increasing trust and supporting security analysts in decision-making. Federated learning has also emerged as a potential solution for privacy-preserving malware detection. By enabling collaborative model training across multiple devices without sharing sensitive user data, federated learning can improve detection capabilities while maintaining user privacy. Additionally, cloud-assisted and edge-based analysis frameworks can address the computational overhead associated with dynamic analysis and enable real-time malware detection. Future dynamic analysis systems should also focus on overcoming anti-analysis techniques employed by modern malware, including emulator detection, delayed execution, and environment-aware triggering mechanisms. Developing adaptive sandbox environments that closely mimic real user behavior can improve execution coverage and expose hidden malicious activities. Furthermore, integrating dynamic analysis with static analysis, threat intelligence, and behavioral profiling can lead to more robust hybrid detection frameworks. As Android ecosystems continue to evolve with emerging technologies such as IoT devices, 5G networks, and mobile edge computing, scalable and intelligent malware detection solutions will become increasingly important. Therefore, future research should emphasize accuracy, scalability, interpretability, and resilience to adversarial attacks to strengthen Android malware detection systems.

VIII. CONCLUSION

Android malware has become a major cybersecurity threat due to the widespread use of Android devices and the growing complexity of malicious applications. Since modern malware often uses obfuscation and other evasion techniques, dynamic analysis is widely used to detect malicious behavior by monitoring applications during runtime. This survey reviewed various dynamic analysis approaches for Android malware detection, including sandbox-based analysis, system call monitoring, API call analysis, taint tracking, behavioral profiling, and machine learning-based techniques. The reviewed studies demonstrate that dynamic analysis provides valuable runtime insights and improves the detection of both known and previously unseen malware variants. Several frameworks and deep learning models have further enhanced detection accuracy and behavioral analysis capabilities. Despite these advantages, dynamic analysis still faces challenges such as anti-emulation techniques, limited execution coverage, high computational overhead, scalability issues, and adversarial attacks. Addressing these limitations remains an important research direction. Future work should focus on hybrid analysis frameworks, explainable artificial intelligence, federated learning, and cloud-assisted detection systems to improve accuracy, scalability, and resilience. Overall, dynamic analysis remains a critical component of modern Android malware detection and mobile cybersecurity.

REFERENCES

[1] A. C. Cinar and T. B. Kara, “The current state and future of mobile security in the light of the recent mobile security threat reports,” Multimed. Tools Appl., vol. 82, no. 13, pp. 20269–20281, May 2023, doi: 10.1007/s11042-023-14400-6.

[2] C. S. Yadav et al., “Malware Analysis in IoT & Android Systems with Defensive Mechanism,” Electronics, vol. 11, no. 15, p. 2354, Jan. 2022, doi: 10.3390/electronics11152354.

[3] B. Prabhu Kavin et al., “Machine Learning-Based Secure Data Acquisition for Fake Accounts Detection in Future Mobile Communication Networks,” Wirel. Commun. Mob. Comput., vol. 2022, no. 1, p. 6356152, 2022, doi: 10.1155/2022/6356152.

[4] M. Mohd Saudi, M. A. Husainiamer, A. Ahmad, and M. Y. I. Idris, "iOS mobile malware analysis: a state-of-the-art," J. Comput. Virol. Hacking Tech., vol. 20, no. 4, pp. 533–562, Nov. 2024, doi: 10.1007/s11416-023-00477-y.

[5] Ö. Aslan, M. Ozkan-Okay, and D. Gupta, "Intelligent Behavior-Based Malware Detection System on Cloud Computing Environment," IEEE Access, vol. 9, pp. 83252–83271, 2021, doi: 10.1109/ACCESS.2021.3087316.

[6] E. C. Bayazit, O. K. Sahingoz, and B. Dogan, "Protecting Android Devices from Malware Attacks: A State-of-the-Art Report of Concepts, Modern Learning Models and Challenges," IEEE Access, vol. 11, pp. 123314–123334, 2023, doi: 10.1109/ACCESS.2023.3323396.

[7] M. Usama Tanveer, K. Munir, A. Alabdulatif, A. R. Najdawi, and R. H. Jhaveri, "Malware-SeqGuard: An Approach Utilizing LSTM and GRU for Effective Detection of Evolving Malware in Android Environments," IEEE Access, vol. 13, pp. 117355–117373, 2025, doi: 10.1109/ACCESS.2025.3585605.

[8] E. Alkhateeb, A. Ghorbani, and A. Habibi Lashkari, "Identifying Malware Packers through Multilayer Feature Engineering in Static Analysis," Information, vol. 15, no. 2, p. 102, Feb. 2024, doi: 10.3390/info15020102.

[9] M. Maghanaki, S. Keramati, F. F. Chen, and M. Shahin, "Generation of a Multi-Class IoT Malware Dataset for Cybersecurity," Electronics, vol. 14, no. 21, p. 4196, Jan. 2025, doi: 10.3390/electronics14214196.

[10] F. Alhaidari et al., "ZeVigilante: Detecting Zero-Day Malware Using Machine Learning and Sandboxing Analysis Techniques," Comput. Intell. Neurosci., vol. 2022, no. 1, p. 1615528, 2022, doi: 10.1155/2022/1615528.

[11] J. Park, N.-T. Chau, L. Nguyen-Vu, J. Yoon, and S. Jung, "A-Pot: A Comprehensive Android Analysis Platform Based on Container Technology," IEEE Access, vol. 8, pp. 199638–199645, 2020, doi: 10.1109/ACCESS.2020.3035774.

[12] Z. Meng, Y. Xiong, W. Huang, F. Miao, and J. Huang, "AppAngio: Revealing Contextual Information of Android App Behaviors by API-Level Audit Logs," IEEE Trans. Inf. Forensics Secur., vol. 16, pp. 1912–1927, 2021, doi: 10.1109/TIFS.2020.3044867.

[13] M. Kim, H. Cho, and J. H. Yi, "Large-Scale Analysis on Anti-Analysis Techniques in Real-World Malware," IEEE Access, vol. 10, pp. 75802–75815, 2022, doi: 10.1109/ACCESS.2022.3190978.

[14] V. R. Joshi, K. Assa-Agyei, T. Al-Hadhrami, and S. N. Qasem, "Hybrid AI Intrusion Detection: Balancing Accuracy and Efficiency," Sensors, vol. 25, no. 24, p. 7564, Jan. 2025, doi: 10.3390/s25247564.

[15] A. Amaithi Rajan, V. V, A. M, and P. K. R, "Secure and fault tolerant cloud based framework for medical image storage and retrieval in a distributed environment," Sci. Rep., vol. 15, no. 1, p. 32965, Sep. 2025, doi: 10.1038/s41598-025-16903-8.

[16] F. H. da Costa et al., "Exploring the use of static and dynamic analysis to improve the performance of the mining sandbox approach for android malware identification," J. Syst. Softw., vol. 183, p. 111092, Jan. 2022, doi: 10.1016/j.jss.2021.111092.

[17] N. Aldhafferi, "Android Malware Detection Using Support Vector Regression for Dynamic Feature Analysis," Information, vol. 15, no. 10, p. 658, Oct. 2024, doi: 10.3390/info15100658.

[18] A. De Lorenzo, F. Martinelli, E. Medvet, F. Mercaldo, and A. Santone, "Visualizing the outcome of dynamic analysis of Android malware with VizMal," J. Inf. Secur. Appl., vol. 50, p. 102423, Feb. 2020, doi: 10.1016/j.jisa.2019.102423.

[19] M. İbrahim, B. Issa, and M. B. Jasser, "A Method for Automatic Android Malware Detection Based on Static Analysis and Deep Learning," IEEE Access, vol. 10, pp. 117334–117352, 2022, doi: 10.1109/ACCESS.2022.3219047.

[20] J. Jeon, J. H. Park, and Y.-S. Jeong, "Dynamic Analysis for IoT Malware Detection with Convolution Neural Network Model," IEEE Access, vol. 8, pp. 96899–96911, 2020, doi: 10.1109/ACCESS.2020.2995887.

[21] M. F. Abdelwahed, M. M. Kamal, and S. G. Sayed, "Detecting Malware Activities with MalpMiner: A Dynamic Analysis Approach," IEEE Access, vol. 11, pp. 84772–84784, 2023, doi: 10.1109/ACCESS.2023.3266562.

[22] I. U. Haq, T. A. Khan, and A. Akhunzada, "A Dynamic Robust DL-Based Model for Android Malware Detection," IEEE Access, vol. 9, pp. 74510–74521, 2021, doi: 10.1109/ACCESS.2021.3079370.

[23] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "DL-Droid: Deep learning based android malware detection using real devices," Comput. Secur., vol.

89, p. 101663, Feb. 2020, doi: 10.1016/j.cose.2019.101663.

[24] G. Hu et al., "SAMLDroid: A Static Taint Analysis and Machine Learning Combined High-Accuracy Method for Identifying Android Apps with Location Privacy Leakage Risks," Entropy, vol. 23, no. 11, p. 1489, Nov. 2021, doi: 10.3390/e23111489.

[25] R. Bhan et al., "DLCDroid an android apps analysis framework to analyse the dynamically loaded code," Sci. Rep., vol. 15, no. 1, p. 3292, Jan. 2025, doi: 10.1038/s41598-025-88003-6.

[26] A. Mills and P. Legg, "Investigating Anti-Evasion Malware Triggers Using Automated Sandbox Reconfiguration Techniques," J. Cybersecurity Priv., vol. 1, no. 1, pp. 19–39, Mar. 2021, doi: 10.3390/jcp1010003.

[27] D. Suo, L. Xue, L. Yu, R. Tan, W. Huang, and G. Sun, "ARAP: Demystifying Anti Runtime Analysis Code in Android Apps," IEEE Trans. Softw. Eng., vol. 51, no. 10, pp. 2787–2803, Oct. 2025, doi: 10.1109/TSE.2025.3596016.

[28] M. Torres, R. Álvarez, and M. Cazorla, "A Malware Detection Approach Based on Feature Engineering and Behavior Analysis," IEEE Access, vol. 11, pp. 105355–105367, 2023, doi: 10.1109/ACCESS.2023.3319093.

[29] D. Zhan, K. Tan, L. Ye, X. Yu, H. Zhang, and Z. He, "An Adversarial Robust Behavior Sequence Anomaly Detection Approach Based on Critical Behavior Unit Learning," IEEE Trans. Comput., vol. 72, no. 11, pp. 3286–3299, Nov. 2023, doi: 10.1109/TC.2023.3292001.

[30] Q. Xu, D. Zhao, S. Yang, L. Xu, and X. Li, "Android Malware Detection Based on Behavioral-Level Features with Graph Convolutional Networks," Electronics, vol. 12, no. 23, p. 4817, Jan. 2023, doi: 10.3390/electronics12234817.

[31] P. Meena and K. P. R. Prabha, "Leveraging the Power of Zero-Shot Learning for Malware Detection Using Application Programming Interface Call Sequences," IEEE Access, vol. 13, pp. 138125–138142, 2025, doi: 10.1109/ACCESS.2025.3594087.

[32] P. Krishnan et al., "MUD-Based Behavioral Profiling Security Framework for Software-Defined IoT Networks," IEEE Internet Things J., vol. 9, no. 9, pp. 6611–6622, May 2022, doi: 10.1109/JIOT.2021.3113577.

[33] F. Taher, O. AlFandi, M. Al-kfairy, H. Al Hamadi, and S. Alrabaee, "DroidDetectMW: A Hybrid Intelligent Model for Android Malware Detection," Appl. Sci., vol. 13, no. 13, p. 7720, Jan. 2023, doi: 10.3390/app13137720.

[34] J. A. Johny, K. A. Asmitha, P. Vinod, G. Radhamani, K. A. R. Rehiman, and M. Conti, "Deep learning fusion for effective malware detection: leveraging visual features," Clust. Comput., vol. 28, no. 2, p. 135, Nov. 2024, doi: 10.1007/s10586-024-04723-w.

[35] M. Eskandari, Z. H. Janjua, M. Vecchio, and F. Antonelli, "Passban IDS: An Intelligent Anomaly-Based Intrusion Detection System for IoT Edge Devices," IEEE Internet Things J., vol. 7, no. 8, pp. 6882–6897, Aug. 2020, doi: 10.1109/JIOT.2020.2970501.

[36] A. Albin Ahmed, A. Shaahid, F. Alnasser, S. Alfaddagh, S. Binagag, and D. Alqahtani, "Android Ransomware Detection Using Supervised Machine Learning Techniques Based on Traffic Analysis," Sensors, vol. 24, no. 1, p. 189, Jan. 2024, doi: 10.3390/s24010189.

[37] J. A. Herrera-Silva and M. Hernández-Álvarez, "Dynamic Feature Dataset for Ransomware Detection Using Machine Learning Algorithms," Sensors, vol. 23, no. 3, p. 1053, Jan. 2023, doi: 10.3390/s23031053.

[38] R. Feng, S. Chen, X. Xie, G. Meng, S.-W. Lin, and Y. Liu, "A Performance-Sensitive Malware Detection System Using Deep Learning on Mobile Devices," IEEE Trans. Inf. Forensics Secur., vol. 16, pp. 1563–1578, 2021, doi: 10.1109/TIFS.2020.3025436.

[39] Z. Bala et al., "Transfer Learning Approach for Malware Images Classification on Android Devices Using Deep Convolutional Neural Network," Procedia Comput. Sci., vol. 212, pp. 429–440, Jan. 2022, doi: 10.1016/j.procs.2022.11.027.

[40] N. Owoh, J. Adejoh, S. Hosseinzadeh, M. Ashawa, J. Osamor, and A. Qureshi, "Malware Detection Based on API Call Sequence Analysis: A Gated Recurrent Unit–Generative Adversarial Network Model Approach," Future Internet, vol. 16, no. 10, p. 369, Oct. 2024, doi: 10.3390/fi16100369.