International Journal of Advanced Digital Systems
and Multidisciplinary Studies (IJADSMS),
Volume - 2, Issue - 6 pp. 131–140, 2026

Intelligent Hybrid Android Malware Detection through Static, Dynamic, and Machine Learning Analysis

Mr. Ajit Kumar[1], Ms. Kumari Puja[2], Mr. Abhishek Kumar[3], Mr. Sachin Kashyap[4], Ms. Anupama[5], Ms. Lakshmi Kumari[6]

[1]Assistant Professor, Amity Institute of Information Technology (AIIT), Amity University Jharkhand, Ranchi

[2,3,4,5,6] Assistant Professor ,School of Computer Applications, Noida Institute of Engineering and Technology (NIET) Greater Noida

akumar7@rnc.amity.edu[1], gkp2910@gmail.com[2], abhishek.kumar@niet.co.in[3], sachinkashyap80770@gmail.com[4], anupama0140@gmail.com[5], lakshmi1051999@gmail.com[6]

Abstract

The rapid expansion of the Android ecosystem, combined with its open-source architecture and fragmented security controls, has significantly increased exposure to sophisticated mobile malware threats. Conventional detection approaches relying solely on static or dynamic analysis often fail to provide comprehensive protection due to limitations such as vulnerability to code obfuscation, high computational overhead, and reduced effectiveness against zero-day attacks. This study proposes an intelligent hybrid Android malware detection framework that integrates static code analysis, dynamic behavioral monitoring, and machine learning techniques to improve detection accuracy and robustness. Static features—including permissions, API calls, and opcode sequences—are combined with dynamic features such as system calls, network activity, and memory usage to form a unified feature representation. Multiple classification models, including Support Vector Machines, Random Forests, XGBoost, Convolutional Neural Networks, and Long Short-Term Memory networks, are employed to evaluate detection performance. The hybrid feature fusion strategy aims to leverage complementary analytical strengths while minimizing false positives and negatives. Experimental evaluation using benchmark datasets such as Drebin, AndroZoo, and CICMalDroid demonstrates the effectiveness of the proposed approach in enhancing classification reliability while addressing computational trade-offs. The findings highlight the potential of hybrid intelligent systems to provide scalable and resilient defenses against evolving Android malware, offering valuable insights for future secure mobile application ecosystems.

Keywords: Android Malware Detection, Hybrid Analysis, Static Analysis, Dynamic Analysis, Machine Learning, Deep Learning, Cybersecurity

Introduction

The pervasive adoption and open-source nature of the Android operating system, which commands over 80% of the global smartphone market, unfortunately introduces significant security challenges, exacerbated by ecosystem fragmentation and a lack of centralized oversight for third-party applications [1], [2]. This environment has fostered a dramatic increase in mobile malware, with attackers constantly evolving sophisticated tactics like code obfuscation to evade traditional signature-based detection and exploit various vulnerabilities, leading to millions of new malware samples annually [2], [3], [4], [5], [6]. Consequently, there's a critical motivation for hybrid detection models that combine the strengths of static analysis (examining code for permissions, API calls), dynamic analysis (observing runtime behavior in a sandbox), and advanced machine learning techniques to overcome the limitations of individual methods and effectively combat both known and zero-day threats [2], [7], [8]. Your research, therefore, aims to propose a novel hybrid malware detection method utilizing static, dynamic, and machine learning analysis to improve detection accuracy and provide a more resilient approach against the persistent evolution of Android malware [2], [8], [9], [10].

Literature Review

The landscape of Android malware detection is broadly characterized by two primary analytical approaches: static and dynamic analysis. Static analysis involves examining an application's code without execution, extracting features such as requested permissions, API calls, and opcode sequences [2]. While efficient and capable of identifying known patterns quickly, static methods often face limitations in detecting obfuscated malware, zero-day threats, or new variants that do not conform to existing signatures, sometimes leading to high false positive rates due to the benign co-occurrence of certain features [2], [11], [12]. In contrast, dynamic analysis monitors an application's behavior during runtime within a controlled environment, such as a sandbox, capturing actions like system calls, network communications, and memory usage [2]. This approach excels at uncovering malicious behaviors that are only triggered during execution and can effectively identify unknown malware; however, it tends to be resource-intensive, slower, and can be evaded by malware designed to detect and alter its behavior in a sandbox [2]. To overcome these inherent limitations and the constant evolution of Android malware, researchers have increasingly integrated machine learning and deep learning techniques, which are capable of identifying complex malicious patterns from vast datasets [13], [14], [15], [16]. However, a significant comparative gap in existing research lies in the consistent development and evaluation of robust hybrid models that effectively fuse diverse static and dynamic features, while also accounting for the adversarial nature of malware that attempts to circumvent detection, ensuring high accuracy, low false positives/negatives, and practical deployability against an ever-changing threat landscape [7], [17].

The following table presents a summary of relevant literature in Android malware detection.

Reference	Purpose Number of Studies, Inclusion Criteria, and Databases Searched

Dahiya et al. [18]

To systematically review Android malware analysis and detection, focusing on static analysis, deep learning, and machine learning models, and to identify challenges like obfuscation and dataset issues.	Investigated 99 articles published between 2018 and 2023..

Muzaffar et al. [19]

To critically review past works that have used machine learning for Android malware detection, categorizing approaches by static, dynamic, or hybrid features and covering supervised, unsupervised, deep learning, and online learning methods.	Not explicitly stated in excerpt, but an in-depth review of machine learning-based Android malware detection.

Liu et al. [20]

To systematically review deep learning approaches for Android malware defenses, discussing research trends, focuses, challenges, and future directions.	Investigation included 53 primary studies designing defense approaches..

Maganur et al. [21]

To provide a structured comparison of existing feature-centric techniques for Android malware analysis, identify open research gaps, and outline a roadmap for future work in improving scalability, adaptability, and resilience.	A survey of feature-centric approaches..

Guerra-Manzanares [6]

To identify and summarize research gaps and current challenges in ML-based Android malware detection through an extensive review and analysis of related literature.	An extensive review and analysis of related research literature..

Methodology

This section outlines the proposed methodology for developing an intelligent hybrid Android malware detection system. Our approach integrates static, dynamic, and machine learning analysis to enhance detection accuracy and robustness against evolving threats [7].

2.1 Dataset Collection and Preprocessing

A comprehensive and diverse dataset is crucial for training and evaluating robust malware detection models [22]. Our study will leverage well-established and publicly available datasets, including:

● Drebin: Introduced in 2014 by Arp et al., the Drebin dataset is a prominent collection of Android applications, categorized as benign or malicious, with pre-extracted static features [22], [23]. It includes a significant number of malware samples, often comprising applications from various malware families, and is widely utilized in Android malware detection research for static analysis and model training [24], [25].

● AndroZoo: As a continually expanding dataset, AndroZoo aggregates millions of Android applications from diverse sources, including the official Google Play store [26], [27]. It provides a vast repository of real-world samples, both benign and malicious, enabling large-scale studies and supporting reproducible experiments in the mobile security domain [27], [28], [29].

● CICMalDroid: This dataset, such as CICMalDroid2017 and CICMalDroid2020, is designed to reduce the shortcomings of earlier datasets by providing both malware and benign applications collected from various resources [30], [31]. It offers rich dynamic behavior reports, including network traffic captures, system calls, process logs, and memory usage, making it particularly useful for dynamic analysis and behavior-based detection [31], [32], [33]. The dataset also addresses evasion techniques used by advanced malware by capturing network traffic at different stages of execution [32].

Prior to feature extraction, all collected applications will undergo thorough preprocessing, including decompression, manifest file parsing, and static code analysis setup. Benign samples will be carefully curated to represent typical legitimate application behavior, ensuring a balanced dataset for training.

2.2 Feature Extraction

Our hybrid detection system relies on a combination of static and dynamic features to capture a comprehensive view of an application's characteristics and behavior [34], [35].

2.2.1 Static Feature Extraction

Static analysis involves examining an application's code and manifest file without executing it [36], [37]. This approach is efficient and can identify known patterns quickly [11]. The following static features will be extracted:

● Permissions: Analysis of requested Android permissions as declared in the AndroidManifest.xml [38]. Malicious applications often request excessive or suspicious permissions to access sensitive user data or perform unauthorized actions, making permissions a critical indicator for malware detection [37], [39], [40], [41], [42], [43].

● API Calls: Identification of critical or sensitive Application Programming Interface calls within the application's bytecode [40]. The frequency, specific names, and sequences of these calls can indicate malicious intent, as malware often invokes a different set of API calls compared to benign applications [44], [45], [46], [47].

● Opcode Sequences: Extraction of sequences of Dalvik opcodes from the application's DEX files [48]. Certain opcode patterns can reveal obfuscation techniques or malicious logic, and their analysis has shown effectiveness in classifying Android applications as malware or trusted [48], [49], [50], [51].

These features will be converted into a numerical representation suitable for machine learning models, typically through one-hot encoding for categorical features (permissions) and frequency counts or n-gram analysis for API calls and opcode sequences [49], [51].

2.2.2 Dynamic Feature Extraction

Dynamic analysis monitors an application's behavior during runtime within a controlled environment, such as a sandbox or emulator [2]. This approach excels at uncovering malicious behaviors triggered only during execution [2]. The following dynamic features will be collected:

● System Calls: Monitoring and logging of system calls made by the application (e.g., file system operations, process execution, inter-process communication) [52], [53]. Malware often interacts with the operating system through specific system call sequences to achieve malicious goals like data access or network communication [54], [55], [56].

● Network Behavior: Analysis of network traffic generated by the application, including destination IP addresses, port numbers, communication protocols, and data exfiltration attempts [57], [58], [59]. Malware frequently communicates with command-and-control servers or attempts data exfiltration, making network activities a strong indicator of maliciousness [31], [47], [60], [61], [62], [63].

● Memory Usage: Observation of the application's memory consumption patterns [64]. Anomalous memory usage or access behaviors can indicate malicious activity, as malware might consume more resources or manipulate memory in unusual ways [47], [65], [66], [67], [68].

Dynamic features will be extracted by executing each application in a simulated environment for a predetermined duration. Logs generated during execution will be parsed and processed to extract the aforementioned behavioral indicators, which will then be quantified for machine learning input [52].

2.3 Hybrid Feature Fusion Strategy

To leverage the complementary strengths of static and dynamic analysis, a robust feature fusion strategy will be implemented. Combining static and dynamic features provides a more comprehensive representation of an application, as static analysis can identify inherent properties while dynamic analysis reveals runtime behaviors that might be hidden through obfuscation [9], [21], [34], [35], [69]. This strategy aims to combine the extracted static and dynamic features into a unified feature vector. Techniques such as concatenation, where static and dynamic feature vectors are combined into a single, longer vector, are commonly explored [70]. More advanced fusion methods, such as early fusion (merging features before model training) or late fusion (combining predictions from models trained on separate feature sets), can also be investigated to determine the optimal integration point for improved detection performance [71]. While combining features can enhance accuracy, careful consideration is needed to manage increased dimensionality and avoid issues like the "curse of dimensionality" [72].

2.4 Machine Learning Models

A diverse set of machine learning and deep learning models will be employed to classify applications as benign or malicious based on the fused feature vectors [73].

● Support Vector Machine: A powerful discriminative classifier known for its effectiveness in high-dimensional spaces and its ability to find optimal hyperplanes for classification [74], [75]. SVMs have been successfully applied in Android malware detection to classify applications based on various features like permissions and API calls [76], [77], [78], [79], [80].

● Random Forest: An ensemble learning method that constructs a multitude of decision trees to improve prediction accuracy and control overfitting [73], [81]. RF is robust and has shown high accuracy in Android malware detection, utilizing features like permissions and API calls [82], [83], [84], [85].

● XGBoost: A highly efficient and scalable implementation of gradient boosting, known for its performance with large and varied datasets due to its regularization and parallel processing capabilities [70], [86], [87]. XGBoost has demonstrated strong capabilities in malware classification, achieving high accuracy rates [82], [88], [89], [90].

● Convolutional Neural Networks: These deep learning models are effective at automatically learning hierarchical patterns and have been used in Android malware detection by processing raw opcode sequences, API sequences, or even images derived from application files [36], [47], [81], [91], [92], [93], [94], [95], [96].

● Long Short-Term Memory Networks: A type of Recurrent Neural Network particularly well-suited for processing sequential data, LSTMs can capture long-term dependencies in opcode sequences, API call sequences, or system call traces, making them valuable for analyzing behavioral patterns in Android malware [97], [98], [99], [100], [101], [102], [103], [104].

Each model will undergo rigorous training and validation using cross-validation techniques to ensure generalization ability. Hyperparameter tuning will be performed to optimize the performance of each classifier. The performance metrics, including accuracy, precision, recall, F1-score, and ROC curves, will be used to evaluate and compare the effectiveness of individual and hybrid models.

Results

This section would detail the empirical outcomes of the proposed hybrid Android malware detection system, followed by a thorough analysis and interpretation of these findings.

Performance Comparison

The performance of the various machine learning models on the fused static and dynamic feature sets would be rigorously evaluated. This comparison would highlight the strengths and weaknesses of each model in detecting Android malware and benign applications. Key metrics would include:

- Accuracy: The overall proportion of correctly classified instances [105]. While a high accuracy is desirable, it can be misleading in datasets with class imbalance, where benign samples significantly outnumber malware [105].

- Precision: The ratio of true positives to the sum of true positives and false positives. High precision indicates that the model rarely misclassifies a benign app as malware, thus reducing false alarms [105], [106].

- Recall: The proportion of actual malicious instances that are correctly identified. High recall signifies the model's ability to detect malware effectively, minimizing false negatives [105], [106].

- F1-Score: The harmonic mean of precision and recall, providing a balanced measure of a model's accuracy and its ability to recognize positive instances [105], [106].

- Confusion Matrix: A detailed breakdown illustrating true positives, true negatives, false positives, and false negatives [107]. This allows for a granular understanding of the model's classification behavior across different classes [108].

The results would likely be presented in tables and figures, comparing the metrics across different models and potentially across various datasets or experimental setups. For instance, studies often find that ensemble methods like Random Forest and XGBoost achieve high accuracy in Android malware detection [76], [82].

Analysis of False Positives/Negatives

A critical aspect of evaluating any malware detection system is the analysis of its false positive and false negative rates.

- False Positives: Occur when a legitimate application is incorrectly classified as malware [107]. A high FP rate can lead to user dissatisfaction, unnecessary application removals, and a lack of trust in the detection system [109], [110]. In the context of Android malware, legitimate applications sometimes request dangerous permissions for valid reasons, which can contribute to false positives in permission-based detection [110].

- False Negatives: Occur when a malicious application is incorrectly classified as benign [107]. A high FN rate is a significant security risk, as undetected malware can cause severe harm to users and their data [109], [110], [111]. The evolving nature of malware, including obfuscation and adversarial attacks, makes it challenging for detection systems to avoid false negatives [112].

The discussion would delve into the reasons behind observed FP and FN rates for each model. For example, some models might prioritize a lower FN rate (to catch more malware) at the expense of a slightly higher FP rate, while others might aim for high precision to minimize user inconvenience. Strategies to mitigate both types of errors, such as refining feature sets or employing more robust learning algorithms, would be discussed.

Computational Overhead Considerations

The practical deployability of an Android malware detection system heavily depends on its computational overhead. This section would analyze the resources required for feature extraction, model training, and real-time detection on a mobile device or server.

- Feature Extraction Cost: The time and computational resources consumed during the extraction of static and dynamic features. Dynamic analysis, while insightful, generally requires higher computational resources compared to static analysis due to the need for execution monitoring [106], [113], [114]. Hybrid approaches, by combining both, inherently introduce more processing demands [115].

- Model Training Time: The duration and computational power needed to train the various machine learning models on the collected datasets. Deep learning models like CNNs and LSTMs, while powerful, often demand significant computational resources and time for training [112].

- Detection Latency: The time taken by the trained model to classify a new, unseen application. For on-device detection, low latency is crucial to avoid impacting user experience. Cloud-based detection might tolerate slightly higher latency but requires efficient network communication [113].

- Resource Consumption: The amount of CPU, memory, and storage utilized by the detection system during operation. This is particularly important for mobile devices with limited resources [109]. Lightweight static analysis often requires fewer resources [106].

The discussion would address the trade-offs between detection accuracy and computational efficiency. While a robust detection mechanism is paramount, it must also be practical for deployment. Solutions to reduce overhead, such as optimized feature selection, model compression, or offloading complex computations to cloud services, could be explored [106], [114]. The challenge of maintaining model effectiveness against evolving malware without excessive retraining costs would also be considered [116], [117].

Conclusion

This research aimed to develop an intelligent hybrid Android malware detection system by integrating static, dynamic, and machine learning analysis. The proposed methodology detailed the utilization of diverse datasets such as Drebin, AndroZoo, and CICMalDroid, followed by a comprehensive feature extraction process encompassing permissions, API calls, and opcode sequences for static analysis, and system calls, network behavior, and memory usage for dynamic analysis. These features were then intended to be fused and fed into various machine learning models, including Support Vector Machines, Random Forests, XGBoost, Convolutional Neural Networks, and Long Short-Term Memory networks, to classify applications as benign or malicious.

References

[1] N. MOHAMUDALLY, Smartphones from an Applied Research Perspective . 2017. doi: 10.5772/65172.

[2] “1006885.pdf.”

[3] U. J. Nzenwata, F. Uchendu, H. Ismail, E. M. Jumoke, and H. O. Johnson, “Malware Investigation and Analysis for Cyber Threat Intelligence: A Case Study of Flubot Malware,” Computer and Information Science , vol. 16, no. 4, p. 47, Nov. 2023, doi: 10.5539/cis.v16n4p47.

[4] T. Sun et al. , “MalLoc: Toward Fine-grained Android Malicious Payload Localization via LLMs,” 2025, doi: 10.48550/ARXIV.2508.17856.

[5] K. A. Kumar, A. Raman, C. L. P. Gupta, and R. R. Pillai, “The Recent Trends in Malware Evolution, Detection and Analysis for Android Devices,” Journal of Engineering Science and Technology Review , vol. 13, no. 4, p. 240, Aug. 2020, doi: 10.25103/jestr.134.25.

[6] A. Guerra-Manzanares, “Machine Learning for Android Malware Detection: Mission Accomplished? A Comprehensive Review of Open Challenges and Future Perspectives,” Computers & Security , vol. 138. Elsevier BV, p. 103654, Dec. 14, 2023. doi: 10.1016/j.cose.2023.103654.

[7] D. Pulido-Cortázar, D. Gibert, and F. Manya, “DeepTrust: Multi-Step Classification through Dissimilar Adversarial Representations for Robust Android Malware Detection,” arXiv (Cornell University) , Oct. 2025, doi: 10.48550/arxiv.2510.12310.

[8] H. Kumar and A. A. Chopan, “Hybrid ML-DL Approach for Android Malware Detection,” Research Square (Research Square) , Nov. 2024, doi: 10.21203/rs.3.rs-5358924/v1.

[9] A. K.A., P. Vinod, R. R. K. A., N. Raveendran, and M. Conti, “Android malware defense through a hybrid multi-modal approach,” Journal of Network and Computer Applications , vol. 233, p. 104035, Sep. 2024, doi: 10.1016/j.jnca.2024.104035.

[10] M. Ashawa and S. Morris, “Analysis of Android Malware Detection Techniques: A Systematic Review,” International Journal of Cyber-Security and Digital Forensics , vol. 8, no. 3. p. 177, Jan. 01, 2019. doi: 10.17781/p002605.

[11] T. Alam, D. Bhusal, and N. Rastogi, “Revisiting Static Feature-Based Android Malware Detection,” arXiv (Cornell University) , Sep. 2024, doi: 10.48550/arxiv.2409.07397.

[12] A. Anand, J. P. Singh, and V. Dhoundiyal, “Android Malware Detection using HexCode Features,” Research Square (Research Square) , Jun. 2024, doi: 10.21203/rs.3.rs-4544871/v1.

[13] T. A. A. Abdullah, W. Ali, and R. Abdulghafor, “Empirical Study on Intelligent Android Malware Detection based on Supervised Machine Learning,” International Journal of Advanced Computer Science and Applications , vol. 11, no. 4, Jan. 2020, doi: 10.14569/ijacsa.2020.0110429.

[14] H. Rathore, S. K. Sahay, P. Nikam, and M. Sewak, “Robust Android Malware Detection System Against Adversarial Attacks Using Q-Learning,” Information Systems Frontiers , vol. 23, no. 4, p. 867, Nov. 2020, doi: 10.1007/s10796-020-10083-8.

[15] M. Rashid et al. , “Hybrid Android Malware Detection and Classification Using Deep Neural Networks,” International Journal of Computational Intelligence Systems , vol. 18, no. 1, Mar. 2025, doi: 10.1007/s44196-025-00783-x.

[16] S. Aurangzeb and M. Aleem, “Evaluation and classification of obfuscated Android malware through deep learning using ensemble voting mechanism,” Scientific Reports , vol. 13, no. 1, Feb. 2023, doi: 10.1038/s41598-023-30028-w.

[17] A. Rashid and J. M. Such, “MalProtect: Stateful Defense Against Adversarial Query Attacks in ML-Based Malware Detection,” IEEE Transactions on Information Forensics and Security , vol. 18, p. 4361, Jan. 2023, doi: 10.1109/tifs.2023.3293959.

[18] A. Dahiya, S. Singh, and G. Shrivastava, “Android malware analysis and detection: A systematic review,” Expert Systems , vol. 42, no. 1. Wiley, Oct. 25, 2023. doi: 10.1111/exsy.13488.

[19] A. Muzaffar, H. R. Hassen, M. A. Lones, and H. Zantout, “An in-depth review of machine learning based Android malware detection,” Computers & Security , vol. 121, p. 102833, Jul. 2022, doi: 10.1016/j.cose.2022.102833.

[20] Y. Liu, C. Tantithamthavorn, L. Li, and Y. Liu, “Deep Learning for Android Malware Defenses: A Systematic Literature Review,” ACM Computing Surveys , vol. 55, no. 8.

Association for Computing Machinery, p. 1, Jun. 22, 2022. doi: 10.1145/3544968.

[21] S. Maganur, Y. Jiang, J. Huang, and F. Zhong, “Feature-Centric Approaches to Android Malware Analysis: A Survey,” arXiv (Cornell University) , Sep. 2025, doi: 10.48550/arxiv.2509.10709.

[22] T. J. C. Miranda, “Profiling and Visualizing Android Malware Datasets,” HAL (Le Centre pour la Communication Scientifique Directe) , Nov. 2022, Accessed: Aug. 2025. [Online]. Available: https://theses.hal.science/tel-04003806

[23] D. J. Arp, M. Spreitzenbarth, M. Huebner, H. Gascón, and K. Rieck, “Drebin: Effective and Explainable Detection of Android Malware in Your Pocket,” Jan. 2014, doi: 10.14722/ndss.2014.23247.

[24] A. T. Kabakuş, “What Static Analysis Can Utmost Offer for Android Malware Detection,” Information Technology And Control , vol. 48, no. 2, p. 235, Jun. 2019, doi: 10.5755/j01.itc.48.2.21457.

[25] A. Alhussen, “Advanced Android Malware Detection through Deep Learning Optimization,” Engineering Technology & Applied Science Research , vol. 14, no. 3, p. 14552, Jun. 2024, doi: 10.48084/etasr.7443.

[26] M. Alecci, P. J. R. Jiménez, K. Allix, T. F. Bissyandé, and J. Klein, “AndroZoo: A Retrospective with a Glimpse into the Future,” p. 389, Apr. 2024, doi: 10.1145/3643991.3644863.

[27] K. Allix, T. F. Bissyandé, J. Klein, and Y. L. Traon, “AndroZoo,” May 2016, doi: 10.1145/2901739.2903508.

[28] L. Li, “Mining AndroZoo: A Retrospect,” vol. 49, p. 675, Sep. 2017, doi: 10.1109/icsme.2017.49.

[29] J.-F. Lalande, V. V. T. Tong, M. Leslous, and P. Graux, “Challenges for Reliable and Large Scale Evaluation of Android Malware Analysis,” p. 1068, Jul. 2018, doi: 10.1109/hpcs.2018.00173.

[30] A. H. E. Fiky, A. E. Shenawy, and M. A. Madkour, “Android Malware Category and Family Detection and Identification using\n Machine Learning,” arXiv (Cornell University) , Jul. 2021, doi: 10.48550/arxiv.2107.01927.

[31] M. K. A. Abuthawabeh and K. Mahmoud, “Enhanced Android Malware Detection and Family Classification, using Conversation-level Network Traffic Features,” The International Arab Journal of Information Technology , vol. 17, p. 607, Jul. 2020, doi: 10.34028/iajit/17/4a/4.

[32] T. Liu, H. Zhang, H. Long, J. Shi, and Y. Yao, “Convolution neural network with batch normalization and inception-residual modules for Android malware classification,” Scientific Reports , vol. 12, no. 1, Aug. 2022, doi: 10.1038/s41598-022-18402-6.

[33] E. G. V. Enriquez and J. Gutiérrez-Cárdenas, “Dynamic Malware Analysis Using Machine Learning-Based Detection Algorithms,” Interfases , no. 19, p. 119, Jul. 2024, doi: 10.26439/interfases2024.n19.7097.

[34] S. Kumar, S. Indu, and G. S. Walia, “Optimal Unification of Static and Dynamic Features for Smartphone Security Analysis,” Intelligent Automation & Soft Computing , vol. 35, no. 1, p. 1035, Jun. 2022, doi: 10.32604/iasc.2023.024469.

[35] M. V. Ngo, T. Truong-Huu, D. Rabadi, J. Y. Loo, and S. G. Teo, “Fast and Efficient Malware Detection with Joint Static and Dynamic Features Through Transfer Learning,” in Lecture notes in computer science , Springer Science+Business Media, 2023, p. 503. doi: 10.1007/978-3-031-33488-7_19.

[36] M. Dhalaria and E. Gandotra, “Convolutional Neural Network for Classification of Android Applications Represented as Grayscale Images,” International Journal of Innovative Technology and Exploring Engineering , vol. 8, p. 835, Dec. 2019, doi: 10.35940/ijitee.l1189.10812s19.

[37] J. M. Arif, M. F. A. Razak, S. Awang, S. R. T. Mat, N. S. N. Ismail, and A. Firdaus, “A static analysis approach for Android permission-based malware detection systems,” PLoS ONE , vol. 16, no. 9, Sep. 2021, doi: 10.1371/journal.pone.0257968.

[38] H. A. Alkaaf, A. Ali, S. M. Shamsuddin, and S. Hassan, “Exploring permissions in android applications using ensemble-based extra tree feature selection,” Indonesian Journal of Electrical Engineering and Computer Science , vol. 19, no. 1, p. 543, May 2020, doi: 10.11591/ijeecs.v19.i1.pp543-552.

[39] A. Ehsan, C. Catal, and A. Mishra, “Detecting Malware by Analyzing App Permissions on Android Platform: A Systematic Literature Review,” Sensors , vol. 22, no. 20. Multidisciplinary Digital Publishing Institute, p. 7928, Oct. 18, 2022. doi: 10.3390/s22207928.

[40] M. Alazab, M. Alazab, A. Shalaginov, A. Mesleh, and A. Awajan, “Intelligent mobile malware detection using permission requests and API calls,” Future Generation Computer Systems , vol. 107, p. 509, Feb. 2020, doi: 10.1016/j.future.2020.02.002.

[41] H. Peng et al. , “Using probabilistic generative models for ranking risks of Android apps,” in Proceedings of the ACM Conference on Computer and Communications Security , Oct. 2012, p. 241. doi: 10.1145/2382196.2382224.

[42] Y. Sharma and A. Arora, “A comprehensive review on permissions-based Android malware detection,” International Journal of Information Security , vol. 23, no. 3. Springer Science+Business Media, p. 1877, Mar. 04, 2024. doi: 10.1007/s10207-024-00822-2.

[43] C. La and K. A. Mar, “Permission-based Feature Selection for Android Malware Detection and Analysis,” International Journal of Computer Applications , vol. 181, no. 19, p. 29, Sep. 2018, doi: 10.5120/ijca2018917902.

[44] M. D’Onghia, M. Salvadore, B. M. Nespoli, M. Carminati, M. Polino, and S. Zanero, “Apícula: Static detection

of API calls in generic streams of bytes," Computers & Security , vol. 119, p. 102775, May 2022, doi: 10.1016/j.cose.2022.102775.

[45] M. S. Salehi and M. Amini, "Android Malware Detection using Markov Chain Model of Application Behaviors in Requesting System Services," arXiv (Cornell University) , Mar. 2022, doi: 10.48550/arxiv.1711.05731.

[46] L. Onwuzurike, M. S. de S. Almeida, E. Mariconti, J. Blackburn, G. Stringhini, and E. D. Cristofaro, "A Family of Droids -- Android Malware Detection via Behavioral Modeling:\n Static vs Dynamic Analysis," arXiv (Cornell University) , Mar. 2018, doi: 10.48550/arxiv.1803.03448.

[47] K. Bakour, H. M. Ünver, and R. Bakır, "The Android malware detection systems between hope and reality," SN Applied Sciences , vol. 1, no. 9, Aug. 2019, doi: 10.1007/s42452-019-1124-x.

[48] B. Kang, S. Y. Yerima, K. McLaughlin, and S. Sezer, "N-opcode analysis for android malware classification and categorization," Jun. 2016, doi: 10.1109/cybersecpods.2016.7502343.

[49] Q. Jérôme, K. Allix, R. State, and T. Engel, "Using opcode-sequences to detect malicious Android applications," Jun. 2014, doi: 10.1109/icc.2014.6883436.

[50] S. Torka and Ş. Albayrak, "Android App Feature Extraction: A review of approaches for malware and app similarity detection," arXiv (Cornell University) . Cornell University, Dec. 16, 2024. doi: 10.48550/arxiv.2412.11539.

[51] G. Canfora, A. D. Lorenzo, E. Medvet, F. Mercaldo, and C. A. Visaggio, "Effectiveness of Opcode ngrams for Detection of Multi Family Android Malware," Aug. 2015, doi: 10.1109/ares.2015.57.

[52] A. Ferrante, E. Medvet, F. Mercaldo, J. Milošević, and C. A. Visaggio, "Spotting the Malicious Moment: Characterizing Malware Behavior Using Dynamic Features," p. 372, Aug. 2016, doi: 10.1109/ares.2016.70.

[53] S. Shakya and M. Dave, "Analysis, Detection, and Classification of Android Malware using System Calls," arXiv (Cornell University) , Jan. 2022, doi: 10.48550/arxiv.2208.06130.

[54] A. D. Lorenzo, F. Martinelli, E. Medvet, F. Mercaldo, and A. Santone, "Visualizing the outcome of dynamic analysis of Android malware with VizMal," Journal of Information Security and Applications , vol. 50, p. 102423, Nov. 2019, doi: 10.1016/j.jisa.2019.102423.

[55] M. Zheng, M. Sun, and J. C. S. Lui, "DroidTrace: A ptrace based Android dynamic analysis system with forward execution capability," p. 128, Aug. 2014, doi: 10.1109/iwcmc.2014.6906344.

[56] A. Saracino, D. Sgandurra, G. Dini, and F. Martinelli, "MADAM: Effective and Efficient Behavior-based Android Malware Detection and Prevention," IEEE Transactions on Dependable and Secure Computing , vol. 15, no. 1, p. 83, Mar. 2016, doi: 10.1109/tdsc.2016.2536605.

[57] "Association for Computing Machinery," Encyclopedia of New Media . Jan. 01, 2003. doi: 10.4135/9781412950657.n9.

[58] J. Feng, L. Shen, Z. Chen, L. Yu, and H. Li, "HGDetector: A hybrid Android malware detection method using network traffic and Function call graph," Alexandria Engineering Journal , vol. 114, p. 30, Nov. 2024, doi: 10.1016/j.aej.2024.11.068.

[59] M. Hatada and T. Mori, "Finding New Varieties of Malware with the Classification of Network Behavior," IEICE Transactions on Information and Systems , no. 8, p. 1691, Jan. 2017, doi: 10.1587/transinf.2016icp0019.

[60] X. Ling et al. , "Adversarial attacks against Windows PE malware detection: A survey of the state-of-the-art," Computers & Security , vol. 128, p. 103134, Feb. 2023, doi: 10.1016/j.cose.2023.103134.

[61] M. Conti, Q. Q. Li, A. Maragno, and R. Spolaor, "The Dark Side(-Channel) of Mobile Devices: A Survey on Network Traffic\n Analysis," arXiv (Cornell University) , Aug. 2017, doi: 10.48550/arxiv.1708.03766.

[62] A. Nadler, A. Aminov, and A. Shabtai, "Detection of malicious and low throughput data exfiltration over the DNS protocol," Computers & Security , vol. 80, p. 36, Sep. 2018, doi: 10.1016/j.cose.2018.09.006.

[63] L. Vigneri, J. Chandrashekar, I. Pefkianakis, and O. Heen, "Taming the Android AppStore: Lightweight Characterization of Android\n Applications," arXiv (Cornell University) , Apr. 2015, doi: 10.48550/arxiv.1504.06093.

[64] E. B. Karbab, M. Debbabi, S. Alrabaee, and D. Mouheb, "DySign: dynamic fingerprinting for the automatic detection of android malware," p. 1, Oct. 2016, doi: 10.1109/malware.2016.7888739.

[65] F. Idrees, M. Rajarajan, M. Conti, T. Chen, and Y. Rahulamathavan, "PIndroid: A novel Android malware detection system using ensemble learning methods," Computers & Security , vol. 68, p. 36, Mar. 2017, doi: 10.1016/j.cose.2017.03.011.

[66] L. Massarelli, L. Aniello, C. Ciccotelli, L. Querzoni, D. Ucci, and R. Baldoni, "Android malware family classification based on resource consumption over time," Oct. 2017, doi: 10.1109/malware.2017.8323954.

[67] A. J. Anand, "Android Dynamic Malware Analysis," International Journal for Research in Applied Science and Engineering Technology , vol. 13, no. 6, p. 3577, Jun. 2025, doi: 10.22214/ijraset.2025.72665.

[68] S. M. R. Hasan and A. Dhakal, "Obfuscated Malware Detection: Investigating Real-World Scenarios Through Memory Analysis," p. 1, Dec. 2023, doi: 10.1109/ictp60248.2023.10490701.

[69] E. Gandotra, D. Bansal, and S. Sofat, "Malware Analysis and Classification: A Survey," Journal of Information

Security , vol. 5, no. 2, p. 56, Jan. 2014, doi: 10.4236/jis.2014.52006.

[70] M. F. Rabby, “Enhancing Android Malware Detection with Hybrid Feature Fusion and Explainable AI: A Practical Approach,” Research Square (Research Square) , Sep. 2025, doi: 10.21203/rs.3.rs-7743689/v1.

[71] A. Ponte, D. Trizna, L. Demetrio, B. Biggio, and F. Roli, “SLIFER: Investigating Performance and Robustness of Malware Detection Pipelines,” arXiv (Cornell University) , May 2024, doi: 10.48550/arxiv.2405.14478.

[72] S. Dambra et al. , “Decoding the Secrets of Machine Learning in Malware Classification: A Deep Dive into Datasets, Feature Extraction, and Model Performance,” in Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security , Nov. 2023, p. 60. doi: 10.1145/3576915.3616589.

[73] O. C. Abikoye, B. A. Gyunka, and A. N. Oluwatobi, “Optimizing Android Malware Detection Via Ensemble Learning,” International Journal of Interactive Mobile Technologies (iJIM) , vol. 14, no. 9, p. 61, Jun. 2020, doi: 10.3991/ijim.v14i09.11548.

[74] M. Ş. Beştaş and Ö. B. DİNLER, “Detection of Android Based Applications with Traditional Metaheuristic Algorithms,” International Journal of Pure and Applied Sciences , vol. 9, no. 2, p. 381, Dec. 2023, doi: 10.29132/ijpas.1382344.

[75] A. Bose, “Propagation, Detection and Containment of Mobile Malware.,” Deep Blue (University of Michigan) , Jan. 2008, Accessed: Apr. 2025. [Online]. Available: https://hdl.handle.net/2027.42/60849

[76] M. S. Akhtar, “Analyzing and comparing the effectiveness of various machine learning algorithms for Android malware detection,” Advances in Mobile Learning Educational Research , vol. 3, no. 1, p. 570, Dec. 2022, doi: 10.25082/amler.2023.01.005.

[77] N. Milošević, A. Dehghantanha, and K. R. Choo, “Machine learning aided Android malware classification,” Computers & Electrical Engineering , vol. 61, p. 266, Feb. 2017, doi: 10.1016/j.compeleceng.2017.02.013.

[78] D. Geneiatakis, G. Baldini, I. N. Fovino, and I. Vakalis, “Towards a Mobile Malware Detection Framework with the Support of Machine Learning,” in Communications in computer and information science , Springer Science+Business Media, 2018, p. 119. doi: 10.1007/978-3-319-95189-8_11.

[79] P. Liu, W. Wang, X. Luo, H. Wang, and C. Liu, “NSDroid: efficient multi-classification of android malware using neighborhood signature in local function call graphs,” International Journal of Information Security , vol. 20, no. 1, p. 59, Mar. 2020, doi: 10.1007/s10207-020-00489-5.

[80] A. Hemalatha, “Mobile Malware Detection using Anomaly Based Machine Learning Classifier Techniques,” International Journal of Innovative Technology and Exploring Engineering , vol. 8, p. 260, Oct. 2019, doi: 10.35940/ijitee.k1040.09811s219.

[81] A. Anand and J. P. Singh, “Smali opcode based Android Malware detection and Obfuscation Identification,” Research Square (Research Square) , Oct. 2023, doi: 10.21203/rs.3.rs-3493657/v1.

[82] S. Bulut and A. Korkmaz, “Comparative Analysis of Machine Learning Models for Android Malware Detection,” Sakarya University Journal of Science , vol. 28, no. 3, p. 517, Jun. 2024, doi: 10.16984/saufenbilder.1350839.

[83] I. M. Ibrahim and A. B. Sallow, “Feature Selection for Android Malware Detection with Random Forest on Smartphones,” Revue d intelligence artificielle , vol. 37, no. 4, p. 857, Aug. 2023, doi: 10.18280/ria.370405.

[84] N. Daoudi, K. Allix, T. F. Bissyandé, and J. Klein, “Assessing the opportunity of combining state-of-the-art Android malware detectors,” Empirical Software Engineering , vol. 28, no. 2, Dec. 2022, doi: 10.1007/s10664-022-10249-9.

[85] C. S. Kumar, S. M. Krishna, V. Ebinazer, N. N. Naidu, and P. Kalyan, “Protecting Androids from Malware Menace Using Machine Learning And Deep Learning,” in Advances in computer science research , Atlantis Press, 2024, p. 285. doi: 10.2991/978-94-6463-471-6_28.

[86] V. G. T. da Costa, S. Barbon, R. S. Miani, J. J. P. C. Rodrigues, B. B. Zarpel�, and O. N. A., “Mobile botnets detection based on machine learning over system calls,” International Journal of Security and Networks , vol. 14, no. 2, p. 103, Jan. 2019, doi: 10.1504/ijsn.2019.100092.

[87] B. B. Zarpel�o, V. G. T. da Costa, S. Barbon, R. S. Miani, and J. J. P. C. Rodrigues, “Mobile botnets detection based on machine learning over system calls,” International Journal of Security and Networks , vol. 14, no. 2, p. 103, Jan. 2019, doi: 10.1504/ijsn.2019.10021704.

[88] H. Bragança, D. Kreutz, V. Rocha, J. Assolin, and and E. Feitosa, “MH-1M: A 1.34 Million-Sample Comprehensive Multi-Feature Android Malware Dataset for Machine Learning, Deep Learning, Large Language Models, and Threat Intelligence Research,” arXiv (Cornell University) , Nov. 2025, doi: 10.48550/arxiv.2511.00342.

[89] R. Kumar and S. Geetha, “Malware classification using XGboost-Gradient Boosted Decision Tree,” Advances in Science Technology and Engineering Systems Journal , vol. 5, no. 5, p. 536, Jan. 2020, doi: 10.25046/aj050566.

[90] H. Bragança, V. Rocha, L. Barcellos, E. Souto, D. Kreutz, and E. Feitosa, “Capturing the Behavior of Android Malware with MH-100K: A Novel and Multidimensional Dataset,” p. 510, Sep. 2023, doi: 10.5753/sbseg.2023.233596.

[91] T. KURAL, Y. Sönmez, and M. Dener, “Android Malware Analysis and Benchmarking with Deep Learning,” Düzce Üniversitesi Bilim ve Teknoloji Dergisi , vol. 9, no. 6, p. 289, Dec. 2021, doi: 10.29130/dubited.1015654.

[92] W. W. Lo, S. Layeghy, M. Sarhan, M. Gallagher, and M. Portmann, “Graph Neural Network-based Android

Malware Classification with Jumping Knowledge," p. 1, Jun. 2022, doi: 10.1109/dsc54232.2022.9888878.

[93] M. Schofield et al. , "Convolutional Neural Network for Malware Classification Based on API Call Sequence," Jan. 2021, doi: 10.5121/csit.2021.110106.

[94] S. Y. Yerima and M. K. Alzaylaee, "Mobile Botnet Detection: A Deep Learning Approach Using Convolutional Neural Networks," p. 1, Jun. 2020, doi: 10.1109/cybersa49311.2020.9139664.

[95] S. Y. Yerima and M. K. Alzaylaee, "Mobile Botnet Detection: A Deep Learning Approach Using Convolutional\n Neural Networks," arXiv (Cornell University) , Jul. 2020, doi: 10.48550/arxiv.2007.00263.

[96] A. Ksibi, M. Zakariah, L. Almuqren, and A. S. Alluhaidan, "Deep Convolution Neural Networks and Image Processing for Malware Detection," Research Square (Research Square) , Jan. 2023, doi: 10.21203/rs.3.rs-2508967/v1.

[97] R. Vinayakumar, K. P. Soman, P. Poornachandran, and S. S. Kumar, "Detecting Android malware using Long Short-term Memory (LSTM)," Journal of Intelligent & Fuzzy Systems , vol. 34, no. 3, p. 1277, Mar. 2018, doi: 10.3233/jifs-169424.

[98] P. G. Balikcioglu, M. Sirlanci, O. A. Kucuk, B. Ulukapi, R. K. Turkmen, and C. Acartürk, "Malicious code detection in android: the role of sequence characteristics and disassembling methods," International Journal of Information Security , vol. 22, no. 1, p. 107, Nov. 2022, doi: 10.1007/s10207-022-00626-2.

[99] J. Kang, S. Jang, S. Li, Y. Jeong, and Y. Sung, "Long short-term memory-based Malware classification method for information security," Computers & Electrical Engineering , vol. 77, p. 366, Jun. 2019, doi: 10.1016/j.compeleceng.2019.06.014.

[100] D. Dang, F. D. Troia, and M. Stamp, "Malware Classification using Long Short-term Memory Models," p. 743, Jan. 2021, doi: 10.5220/0010378007430752.

[101] H. D. Misalkar and P. Harshavardhanan, "TDBAMLA: Temporal and Dynamic Behavior Analysis in Android Malware using LSTM and Attention Mechanisms," Computer Standards & Interfaces , vol. 92, p. 103920, Aug. 2024, doi: 10.1016/j.csi.2024.103920.

[102] A. Anand, J. P. Singh, R. S. Khan, A. Kumari, and D. Mishra, "Android Malware Detection using LSTM with Smali Codes," Research Square (Research Square) , Jul. 2023, doi: 10.21203/rs.3.rs-3165300/v1.

[103] P. Pathak, P. Poudel, S. Roy, and D. Caragea, "Leveraging attention-based deep neural networks for security vetting of Android applications," ICST Transactions on Security and Safety , vol. 8, no. 29, p. 171168, Sep. 2021, doi: 10.4108/eai.27-9-2021.171168.

[104] A. Bensaoud and J. Kalita, "CNN-LSTM and transfer learning models for malware classification based on opcodes and API calls," Knowledge-Based Systems , vol. 290, p. 111543, Feb. 2024, doi: 10.1016/j.knosys.2024.111543.

[105] M. M. Koushki, I. Abualhaol, A. D. Raju, Y. Zhou, R. S. Giagone, and S. Huang, "On building machine learning pipelines for Android malware detection: a procedural survey of practices, challenges and opportunities," Cybersecurity , vol. 5, no. 1, Aug. 2022, doi: 10.1186/s42400-022-00119-8.

[106] H. Huang, W. T. Huang, Y. Zhou, W. Luo, and Y. Wang, "FEdroid: Lightweight and Interpretable Detection of Android Malware Using Local Key Information and Feature Selection," Research Square (Research Square) , Aug. 2024, doi: 10.21203/rs.3.rs-4745962/v1.

[107] V. K. K, S. K. P, S. Deepak, G. K. C, and S. Rajeswari, "Design and Development of Android App Malware Detector API Using Androguard and Catboost," International Journal for Research in Applied Science and Engineering Technology , vol. 12, no. 4, p. 5121, Apr. 2024, doi: 10.22214/ijraset.2024.61156.

[108] V. Jyothsna, K. P. Dasari, S. Inuguru, V. B. R. Gowni, J. T. R. Kudumula, and K. Srilakshmi, "Unified Approach for Android Malware Detection: Feature Combination and Ensemble Classifier," in Advances in computer science research , Atlantis Press, 2024, p. 485. doi: 10.2991/978-94-6463-471-6_47.

[109] M. M. Koushki, I. Abualhaol, A. D. Raju, Z. Yang, R. S. Giagone, and S. Huang, "On building machine learning pipelines for Android malware detection: a procedural survey of practices, challenges and opportunities," arXiv (Cornell University) , Jun. 2023, doi: 10.48550/arxiv.2306.07118.

[110] L. Gitzinger, "Surviving the massive proliferation of mobile malware," HAL (Le Centre pour la Communication Scientifique Directe) , Dec. 2020, Accessed: Oct. 2025. [Online]. Available: https://tel.archives-ouvertes.fr/tel-03194472

[111] J. B. Higuera, J. M. Moreno, J. R. B. Higuera, J. A. S. Montalvo, G. J. B. Martillo, and T. M. S. Riera, "Benchmarking Android malware analysis tools," Research Square (Research Square) , Sep. 2023, doi: 10.21203/rs.3.rs-3366597/v1.

[112] J. Liu, J. Zeng, F. Pierazzi, L. Cavallaro, and Z. Liang, "Unraveling the Key of Machine Learning Solutions for Android Malware Detection," arXiv (Cornell University) , Feb. 2024, doi: 10.48550/arxiv.2402.02953.

[113] B. Sharma, "Comparing the Efficiency of Malware Detection in Android System," International Journal for Research in Applied Science and Engineering Technology , vol. 10, no. 7, p. 841, Jul. 2022, doi: 10.22214/ijraset.2022.45373.

[114] A. Davarasan, J. Samual, K. Palansundram, and A. Ali, "A Comprehensive Review of Machine Learning Approaches for Android Malware Detection," Journal of Cyber Security and Risk Auditing , vol. 2024, no. 1. p. 39, Dec. 06, 2024. doi: 10.63180/jcsra.thestap.2024.1.5.

[115] H. Berger, A. Dvir, E. Mariconti, and C. Hajaj, "Breaking the structure of MaMaDroid," Expert Systems with

Applications , vol. 228, p. 120429, May 2023, doi: 10.1016/j.eswa.2023.120429.

[116] B. Molina-Coronado, U. Mori, A. Mendiburu, and J. Miguel-Alonso, “Towards a fair comparison and realistic evaluation framework of android malware detectors based on static analysis and machine learning,” Computers & Security , vol. 124, p. 102996, Oct. 2022, doi: 10.1016/j.cose.2022.102996.

[117] B. Molina-Coronado, U. Mori, A. Mendiburu, and J. Miguel-Alonso, “Towards a Fair Comparison and Realistic Design and Evaluation Framework of Android Malware Detectors,” arXiv (Cornell University) , May 2022, Accessed: Aug. 2025. [Online]. Available: http://arxiv.org/abs/2205.12569 [58] Y. Liu, Q. Lu, G. Yu, H.-Y. Paik, H. Perera, and L. Zhu, “A Pattern Language for Blockchain Governance,” p. 1, Jul. 2022, doi: 10.1145/3551902.3564802.