International Journal in IT & Engineering (IJITE)
Volume 13 Issue 02, February 2026 ISSN: 2321-1776 Impact Factor: 8.256
Journal Homepage: http://ijmr.net.in, Email: irjmss@gmail.com
Double-Blind Peer Reviewed Refereed Open Access International Journal

AN ENSEMBLE MULTI-CLASS CLASSIFIER FOR MALWARE CLASSIFICATION INTO KNOWN MALWARE FAMILIES

Deepak Patel[1], Bharti Kumari[2], Ashish Chaubey[3]

[1]Research Scholar, Department of Computer science and engineering, SHEAT College of Engineering, Varanasi, UP

[2,3]Assistant Professor, Department of Computer science and engineering, SHEAT College of Engineering, Varanasi, UP

ABSTRACT

Android is ubiquitous due to its flexibility and open-source nature. These reasons pose security issues since cybercriminals can exploit user's confidential information or corrupt their systems. This paper introduces a technique to ensemble four distinctive parallel classifiers with different properties for classifying zero-day Android malware.An ensemble multi-class classifier for malware classification into known malware families. Android malware are classified using ensemble parallel classifiers in ML. The aforementioned work yields the feature vectors consisting of the features of the APK namely- permissions, libraries used, services, broadcast receivers, and version number. In addition to the static features, the class of malware family which is to be predicted for a particular APK is also added to the dataset. The proposed methodology in this paper, which classifies Android malware with an accuracy of 93.90%, is an improvement compared to individual classifiers.

KEYWORDS: Android malware, ensemble classifiers, parallel classifier, precision, recall etc.

INTRODUCTION

Smartphone technology has advanced to the point where it is comparable to personal computers. With greater computer capability, cellphones are becoming more common in our daily lives. As a result, the number of smartphone users has increased significantly during the previous five years. Furthermore, mobile Operating System (OS) vulnerabilities have increased by ~50% since the first half of 2019. This jump is entirely due to an increase in new Google Android dangers, from 241 (in 2019) to 492 (in 2020) [1], as seen in Figure 1. Although other mobile systems exist, Google's Android and Apple's iOS have a combined market share of around 99% as of 2020 [2]. As a result, attackers focus mostly on Android and iOS.

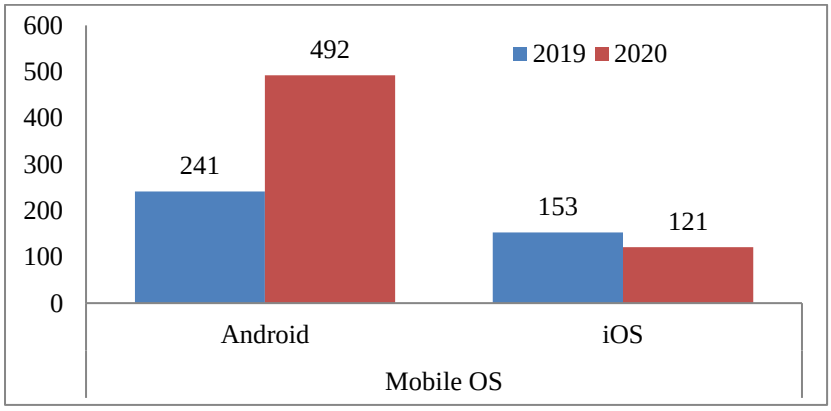

FIG 1: NEW VULNERABILITIES FROM 2019-2020

According to market and security trends noticed in 2025, new Google Android vulnerabilities climbed from around 39,962 in 2024 to 48,185 in 2025. Attacks on Android users increased by 29% in the first half of 2025, compared to the first half of 2024. By late 2025, Android and iOS combined have almost 99% of the worldwide market share (Android at ~71.4% and iOS at ~28.2%). High-severity risks continue to emerge, with a 37.5% exploitation rate among reported serious issues in the most recent Android edition.

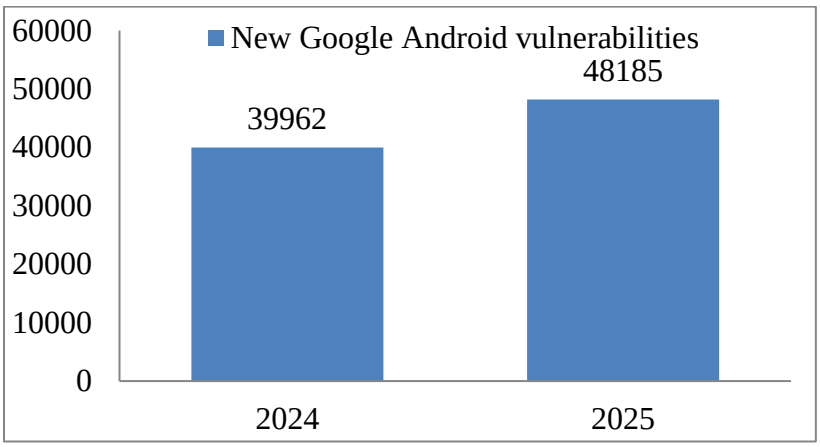

FIG 2: NEW GOOGLE VULNERABILITIES 2024-2025

The 2025 environment for mobile zero-day exploits shows a high-stakes "arms race" between attackers and platform developers. According to the Google Threat Intelligence Group's (GTIG) 2025 Review, mobile OS zero-day vulnerabilities increased to 15 in 2025 from 9 in 2024.

TABLE 1: COMPARISON: ANDROID VS. IOS ZERO-DAYS (2025)

Feature	Google Android	Apple iOS
Total Zero-Days	**15** identified for mobile platforms (majority Android)	**8** total zero-days across all Apple products
Exploit Cost	**$5M - $7M** for a full zero-click chain	**$5M** for a full zero-click chain
Primary Actors	Commercial Surveillance Vendors (CSVs) & State-sponsored (PRC-nexus)	High-end spyware providers (NSO Group, Paragon)
Attack Complexity	Often requires chaining **3+ vulnerabilities**	Targeted, sophisticated "one-shot" or zero-click attacks

MALWARE

Malware is a malicious software program that disrupts computer operations, steals sensitive information, bypasses access controls, gains access to private computer systems, and displays unwelcome adverts [3]. Malware may be classified into many categories based on its intended use, as seen in Figure 3.

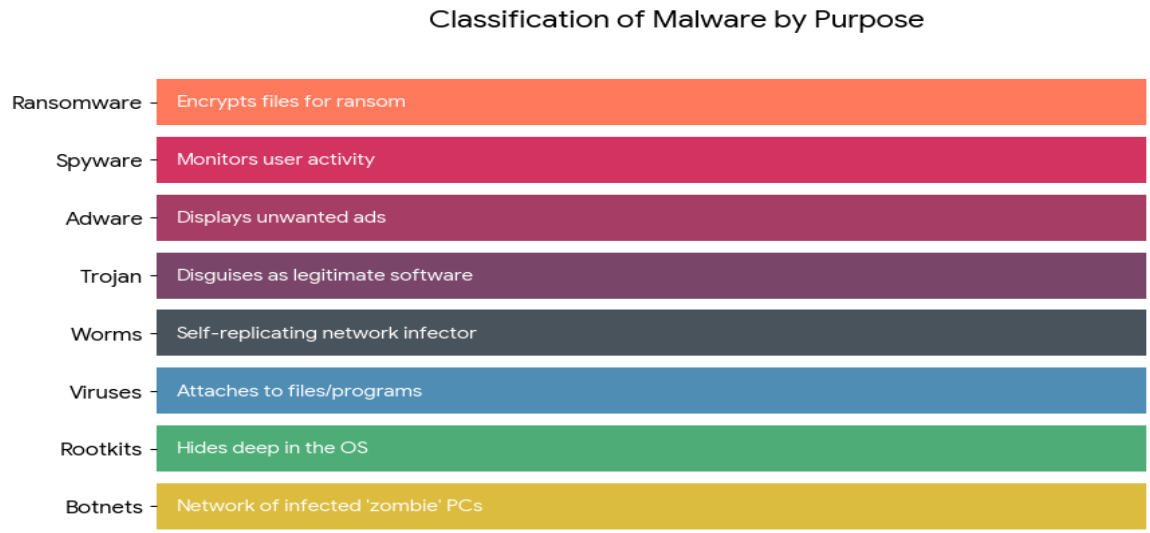

FIGURE 3: TYPES OF MALWARE

Malware may be delivered to mobile devices via a variety of attack channels, posing substantial hazards. An attack vector is a method or technique by which a hacker obtains access to another computing device or network in order to inject a "bad code" known as payload [4]. This channel enables hackers to exploit system weaknesses. Figure 4 depicts a schematic picture of the procedure used by attack vectors to propagate malware in mobile devices.

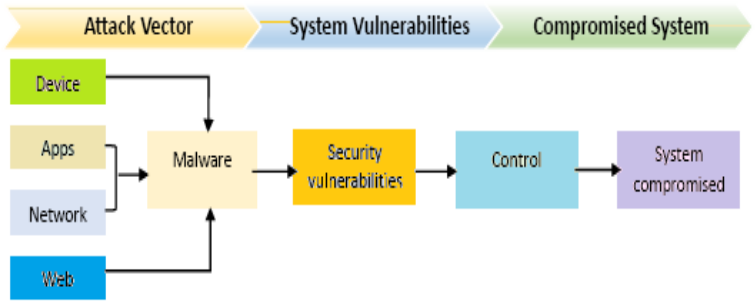

FIG. 4: SCHEMATIC VIEW OF MALWARE PROPAGATION

Malware spread in the mobile environment is very systematic, taking use of cellphones' particular connection and hardware capabilities. Mobile devices, unlike PCs, are continually linked to numerous radio frequencies (cellular, Wi-Fi, Bluetooth) and store sensitive personal information.

1. Distribution (The Delivery)

App Store Injection: Attackers upload "Trojanized" versions of popular programs or games to third-party shops, or they exploit flaws in official store screening procedures.

Smishing & IM: High-urgency links can be sent by SMS or Instant Messengers (WhatsApp, Telegram). In 2025, zero-click attacks (which need no user activity) will be the principal vector for high-end spyware.

QR Code Fraud (Quishing): QR codes can be physical or digital, and can route mobile browsers to exploit kits or phishing URLs.

2. Infection & Privilege Escalation

When a user contacts with the vector, the malware tries to employ OS-level vulnerability to "break out" of the application's sandbox.

Goal: Get root (Android) or kernel (iOS) access to manipulate devices such as the camera, microphone, and GPS.

3. Autonomous Propagation (Lateral Movement)

Contact Worming: The virus instantly distributes itself to the device's complete contact list via SMS or social media.

Proximity Attacks: Using Bluetooth or NFC to look for nearby susceptible devices in public places, and attempting to propagate without an online connection.

4. C&C Synchronization

The infected device communicates with a Command & Control (C&C) server to obtain fresh instructions, post stolen credentials, or engage in dispersed assaults (Mobile botnets).

LITERATURE REVIEW

Faruki et al. [5] investigated malware growth, anti-analysis strategies, and malware detection methods. They also addressed how stealthy approaches like encryption and code modification might produce different types of malware. They investigated static and dynamic techniques to malware detection.

Shrestha et al. [6] presented Tap-Wave-Rub, a lightweight permission enforcement technique for smart phone malware avoidance. The solution relied on two mechanisms: acceleration-based phone tapping detection and proximity-based finger tapping, rubbing, or hand waving detection. This technique was effective in detecting malware, with a low false-positive rate.

Cai et al. [7] developed and deployed DroidCat, a program that detects and categorizes Android malware using systematic dynamic profiling and supervised machine learning approaches. They established 122 behavioural measures, 70 of which were substantially different between benign and malignant programs. DroidCat achieved an accuracy of 92.00% using random forests.

According to Karbab et al. [8], fingerprints of Android malware were obtained using dynamic analysis. The authors presented DySign, a unique approach for fingerprinting Android malware's dynamic behaviours. They created a digest of a malware sample using dynamic analysis of previously known malware. Although DySign used dynamic analysis to serve as the first line of defence against Android malware assaults, it had numerous drawbacks. First,

numerous DySign runs produced various fingerprints, demonstrating that DySign is non-deterministic. Second, DySign was unable to identify new malware families.

Cho, T., and Seo, S.H. [9] presented a way for mounting malware that exploited Android's code-signing mechanism. They also recommended a countermeasure to avert the assault. Because of this, repackaged malware is easily transmitted. The methodologies presented are insufficient to cover all of Android and iOS's security features. Most research publications lack a comprehensive understanding of smart phone vulnerabilities and their influence on confidentiality, integrity, and availability.

According to Tam, K. et al. [10], the Pegasus malware targets Android smartphones and is widely distributed owing to its significant effect. Pegasus is activated by a simple click on a phishing SMS and spies on conversations going place around the device by using the phone's cameras and microphone. It may steal end-to-end encrypted messages between clients and track the victim's travels. Multiple known vulnerabilities exist on mobile devices as well. Google releases security bulletins at regular intervals that outline the patches for the most recent device vulnerabilities.

Wang et al. [11] found Android malware by analyzing network traffic text semantics. They treated each HTTP flow created by mobile apps as a text document, and used NLP to extract text-level information. Text semantic properties of network traffic were exploited to create a successful virus detection model. They analyzed 31,706 benign flows and 5,258 harmful flows. They obtained an accuracy of 99.15%, but had significant restrictions. For starters, several of the destructive actions were not fully activated without effective human interaction. Second, the system used n-gram features to discover unknown samples that had certain characteristics with the examples of malware in the training dataset.

Shen et al. [12] advocated using information flow analysis to detect mobile malware. Their approach centred on the structure of complex flows, patterns, and behaviours in both benign and malevolent applications. Because no native code was evaluated in this case, the technique was unable to detect malicious behaviour.

Garcia et al. [13] introduced RevealDroid, a platform for Android malware detection and family identification. Malware families were identified using supervised machine learning methods. They attained a 95.00% accuracy rate in determining malware families. However, RevealDroid's dataset poses a danger to external validity.

Zhang, L. [14] investigated smartphone security vulnerabilities in apps that used the SSL protocol for secure communication, Web View technology, and HTML5-based. The author demonstrated SSL validation, WebView code injection, and HTML5-based application threats in an exact and straightforward manner. However, the work was not particularly thorough because they just used the functions for analyzing SSL vulnerabilities.

Talat et al. [15] fully presented smart phone security in four separate categories: mobile security elements, smart phone security solutions, research on dangers posed by smartphone malware, and malware classification by family. The first category evaluated survey articles about smart phone security. The second category focused on articles on smart phone security solutions. The third category includes malware-related research and risks, while the fourth category

discusses malware ranking and classification into families. There are several issues about Android and iOS security.

Karbab and Debbabi [16] presented the MalDy framework for malware detection, which incorporates powerful NLP and supervised machine learning. MalDy behavioural reports were converted into word sequences, which were then used to automatically construct appropriate security measures using NLP and ML techniques. MalDy had strong results, but it was unable to monitor the quality of the behavioural reports, which had an influence on performance.

Peynirci et al. [17] introduced a unique feature selection technique and an Android malware detection strategy. The static features employed in this study included permissions, API calls, and strings. The feature vector was then input into several ML algorithms to identify Android malware. For feature selection, the authors employed the Delta IDF technique, which is based on document frequency. They achieved the best accuracy and malware detection rates using the MalGenome dataset (99.40% to 99.80%), the AndroZoo dataset (99.70% to 100.00%), and the Drebin dataset (98.80% to 99.60%).

Dhalaria, M., and Gandotra, E. [18] explain the fundamentals of Android malware, including its evolution and tools and methodologies for malware analysis. This study presents a systematic and complete examination of the tools and techniques used to analyze, classify, and identify malicious Android apps. It covers a chronology of Android malware evolution, as well as tools and techniques for analyzing them statically and dynamically in order to extract characteristics, which are then used for detection and classification using machine learning and deep learning algorithms.

According to Kouliaridis, V., and Kambourakis, G. [19], the bulk of extant studies use distinct metrics and models, as well as diverse datasets and classification characteristics derived from various analytic methodologies, such as static, dynamic, or hybrid. This complicates the cross-comparison of the many suggested detection systems and may cast doubt on the resultant conclusions. Furthermore, based on these axes, we propose a converging strategy that can lead future Android malware detection systems and offer a solid foundation for machine learning activities in this sector.

Yadav, C. S., and Gupta, S. [20] discovered that for improved model construction and assessment, both are strongly suggested for the deployment of successful malware detection algorithms. In addition to these models, Honeynet, IDS, IPS, hardware-based security such as CPU and memory, and forensic analysis are particularly useful for protecting the infrastructure. This study investigates the cause of threads/vulnerabilities in IoT, IIoT, SCADA, and Android application systems. In addition, a general framework-based introduction to IoT and Android is provided, along with common vulnerabilities at each level and mitigation measures.

Elsersy et al. [21] provide a survey of current evasion tools and approaches. The paper highlights the current detection research gap in the most recent Android malware detection frameworks and compares classification performance to various evasion tactics. The study closes research gaps in assessing the resilience of the present Android malware detection architecture against cutting-edge evasion tactics. The study closes on recent Android malware detection-related challenges and lessons gained that researchers should consider in the future.

Tang, L., et al. [22] offer a fine-grained and detailed investigation of Android malware. The experimental approach systematically shows the evolutionary links between variant sets, allowing for a more in-depth investigation of malware development. To assess the accuracy of our study, they use five metrics: silhouette coefficient, creation date, variant labels, the presentativeness of the variant set formula, and the correctness of the associated edges. All of the linked variation sets are connected using our PhyloNet building criteria. They then examine the code specifics of Android malware for each variation set and summarize concepts of evolutionary development.

Faruki, P. et al. [23] provide a complete taxonomy and evaluation of Android-based malware evasion approaches used to avoid malware detection. The paper divides such evasion tactics into two major categories: polymorphism and metamorphism, and it examines techniques for stealth malware detection based on the virus's distinctive properties. Furthermore, the study provides a qualitative and methodical evaluation of evasion detection frameworks and detection approaches for Android malware. Finally, the poll examines open-ended topics and possible future study areas in mobile virus detection.

Sharma, M., and Kaul, A. [24] investigated the issue of detecting malware in these devices and presented several methodologies and strategies. This in-depth review essay delves into the history, progress, and sustainability of Android malware detection. It provides an in-depth assessment of the most recent methodologies and research trends for malware detection, ranging from static to dynamic analysis, machine learning, and deep understanding.

Mohd Saudi, et al., [25] describe a cutting-edge deep analysis of malware targeting iOS cellphones. This comprises extensive research on malware design, including payload, propagation, operating algorithm, infection, and activation, as well as underlying integration with a phylogenetic idea. The results demonstrated that there is a method for identifying the evolution of malware, and as a consequence, a model was constructed. According to the study, 4% of mobile applications followed the patterns outlined in this model. This demonstrates that the methodology described in this work can detect any potential security exploit involving social media and online banking for iOS mobile applications. This study can be used to guide future scholars working on comparable topics.

Dahiya, A., et al., [26] provide a rapid comprehension and comprehensive picture of malware detection and analysis. The current study used a systematic literature review (SLR) to identify significant shifts in malware detection by evaluating a variety of scientific publications and conference papers. Certain obstacles exist in Android malware research, such as obfuscation strategies, dynamic code loading, and concerns with experimental datasets.

Almarri, S., et al. [27] address important advances in malware detection approaches, including those that use behavioural analysis, artificial intelligence (AI), and machine learning (ML). The study highlights flaws in current detection methods and emphasizes the significance of hybrid approaches that combine static and dynamic analysis to address challenges like obfuscation, polymorphism effectively and encrypted threats. Furthermore, the study looks into the effectiveness of collaborative threat intelligence sharing and AI-powered technologies in malware detection and analysis.

Chimeleze et al. [28] provide a complete assessment of research trends, investigate Android malware behaviours over time, and evaluate patterns across platforms, families, and

locations. Furthermore, it assesses current Android malware taxonomies and detects critical gaps. To solve these gaps, we propose an updated taxonomy for sophisticated Android malware. The paper closes with meaningful recommendations for future research, which will help consumers and industry experts mitigate the rising threats posed by sophisticated Android malware assaults.

METHODOLOGY

When a malware sample is found, it is critical to categorize it into recognized families. The malware family is made up of comparable forms of malware that share common traits and behaviours. The same package names may be used to inject a payload and start an attack. A malware family's signature is defined by its common harmful behaviour, traits, and frequent usage of package names.

To address these challenges, an ensemble classifier approach for malware classification is developed, with characteristics obtained from quick and expandable, yet accurate and obfuscation-resistant APKs. In this method, four ML algorithms are chosen based on their distinct properties: MLP, SVM, PART, and RIDOR, and a mixture of these is run in parallel to improve accuracy and efficiency. Figure 5 depicts the several processes involved in the malware categorization process.

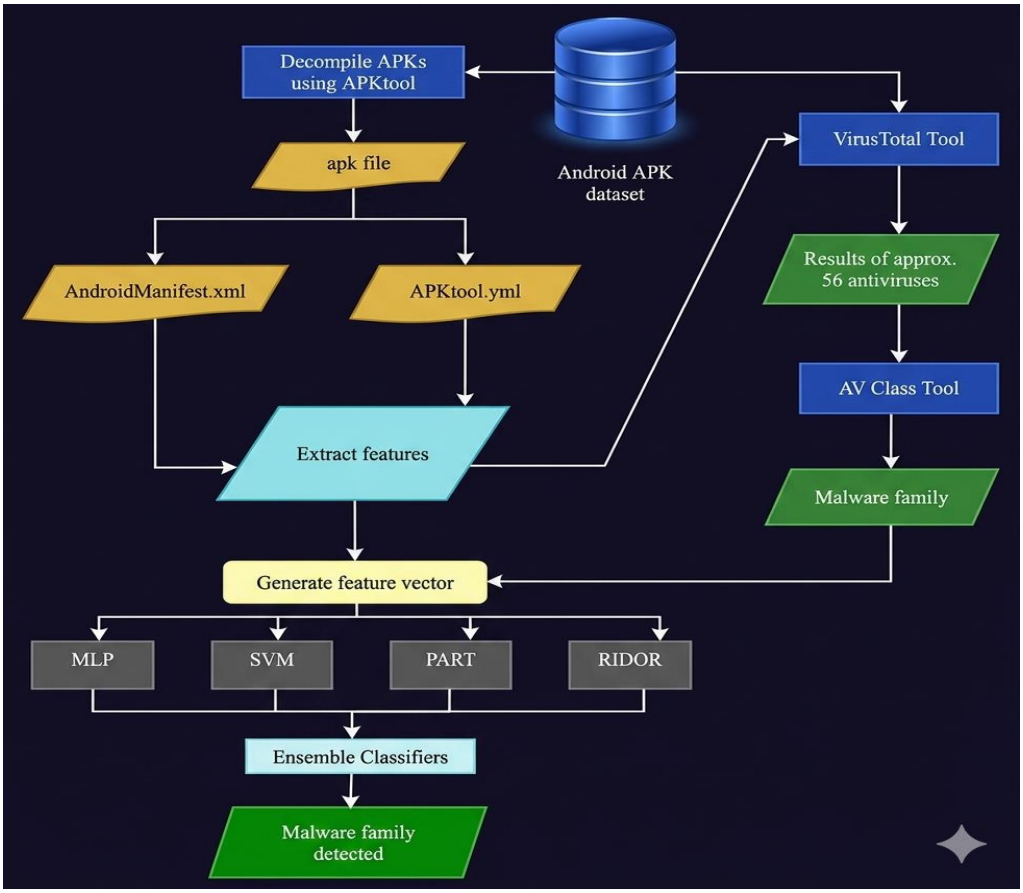

FIG. 5: SCHEMATIC REPRESENTATION FOR ANDROID CLASSIFICATION

DATA SETUP

This section outlines the procedures for gathering, filtering, and preliminary processing data.

DATA COLLECTION AND PRE-PROCESSING- APKs are gathered from several sources, including Google Playstore [29], Wandoujia [30], AMD [31], and Androzoo [32]. The acquired APKs are analyzed and decompiled to build the training and testing dataset.

DATA FILTRATION- Only harmful APKs are required for malware family detection and categorization before proceeding with further processing and deployment. The steps are as follows.

- A CSV file is prepared with the names of various APKs.
- Use a Python script to generate MD5 hashes for all APKs in the CSV file.
- APKs are uploaded using a Python script to the Virus Total API. The results are then obtained using the Python script. The API returns results for around 56 anti-viruses. A count of anti-viruses is kept for dangerous APKs, and each APK's JSON answer is saved in a separate file for later analysis.

DATA CLASSIFICATION- The findings from Virus Total are put into the AV class program, which returns the malware family for each APK. According to VirusTotal, the APKs are classified into 71 malware families. The families discovered by Virus Total serve as a class to be predicted using supervised machine learning. The feature vector used to train the model is made up of static features retrieved from the APK.

ML-BASED TECHNIQUES- In machine learning, a classifier algorithm assigns input data to one of several categories. Our study focuses on supervised learning, which involves training using a labelled dataset. Our technique employs many machine learning algorithms, including MLP, SVM, PART, and RIDOR. These four algorithms were chosen due to their limited implementations in previous studies and their heterogeneous character. Furthermore, no one has ever employed the combination of MLP, SVM, PART, and RIDOR in Android malware classification. Previous ML-based techniques to Android malware detection were presented, but they were not very efficient or scalable [33].

These problems necessitate more effective malware detection mechanisms in order to reduce the spread of Android malware. This is why the Chapter provides a technique for detecting and classifying Android malware in its early stages using parallel ML classifiers with various characteristics. The goal of picking only these specific classifiers is to obtain maximum accuracy because different classifiers have distinct qualities such as rule-based, function-based, etc.

Advantages of these combination classifiers include the ability to quickly classify new cases and provide extensive coverage. Simultaneous execution of the algorithms is accomplished by assigning one core to each algorithm, resulting in increased precision. Furthermore, using numerous threads in parallel on various cores may help to achieve speedier results.

RESULTS AND DISCUSSION

Using the above methods, we were able to divide the APKs into 71 malware families, which serve as the class label for our ML algorithms as well as the training and testing data. Figure 6 shows how the dataset was classified into several malware families.

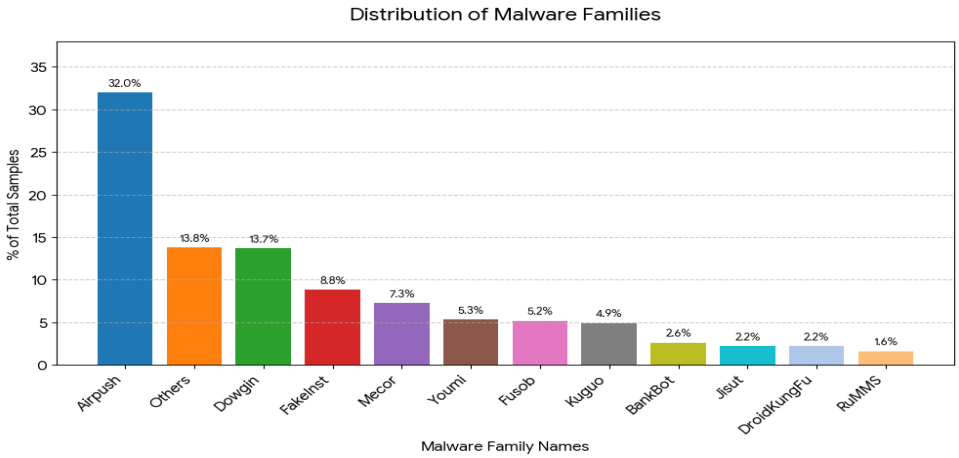

FIG. 6: SAMPLE % FOR MALWARE FAMILIES

The initial section of the examination assessed each algorithm independently. The results received from the individual execution of the algorithms are recorded, and the required representations are constructed. The APKs are categorized into 71 malware families, and the precision and recall for each family are determined by separate methods, with graphs produced to show the findings. The findings of the individual algorithms are summarized in the section that follows.

MLP - Figure 7 shows a histogram of malware families (71 in all), as well as the precision with which they were categorized using the MLP classifier. The red line represents the total percentage of families falling within certain accuracy ranges. Approximately 45% of families (e.g., Simple Locker, FakeInst, Vidro) were categorized with a precision of 0.9-1.0. After investigation, about 80% of the malware families were categorized with an accuracy range of 0.7-1.0.

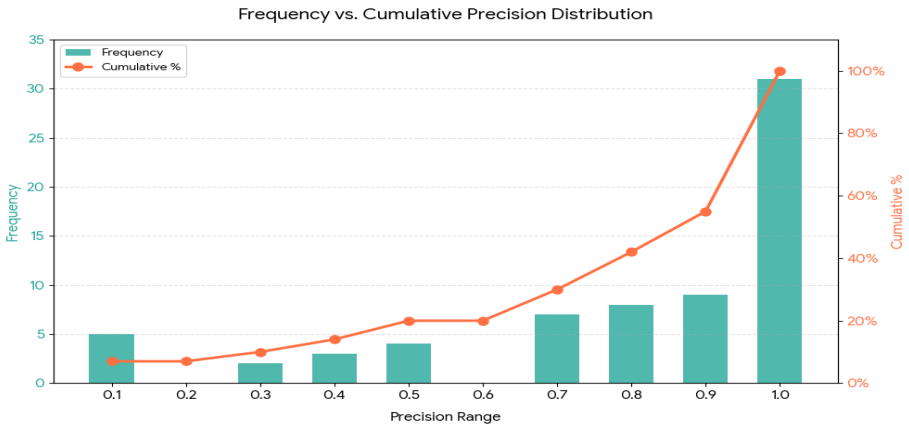

FIG. 7: HISTOGRAM FOR PRESISION MALWARE FAMILIES FOR MLP

Figure 8 shows that over 52% of all families had a memory level of 0.8-1.0, whereas almost 80% had a recall level of 0.5-1.0.

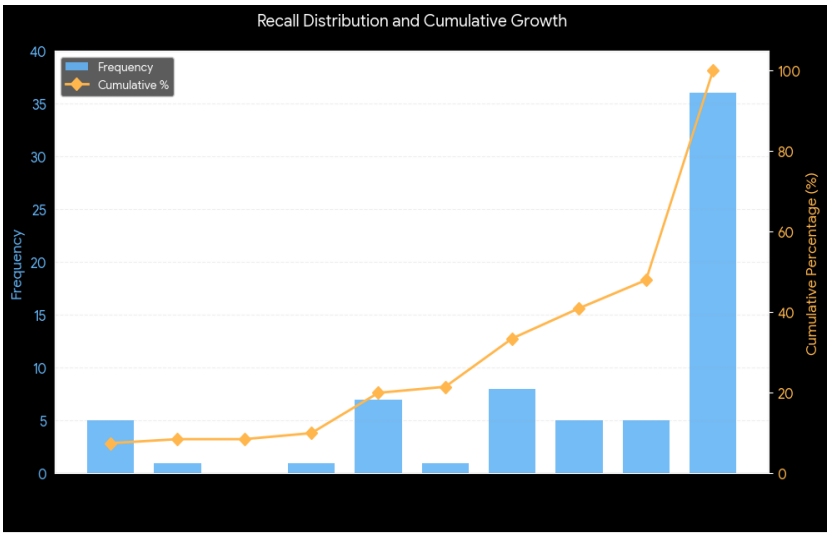

FIG. 8: HISTOGRAM FOR RECALL MALWARE FAMILIES FOR MLP

SVM - Figure 9 depicts a histogram of the incidence of malware families (71 in total), as well as the precision with which they were categorized using an SVM. Approximately 75% of families, including Airpush, Cova, and FakeAV, were categorized with a precision level of 0.9-1.0. After investigation, about 80% of the malware families were categorized with an accuracy range of 0.8-1.0.

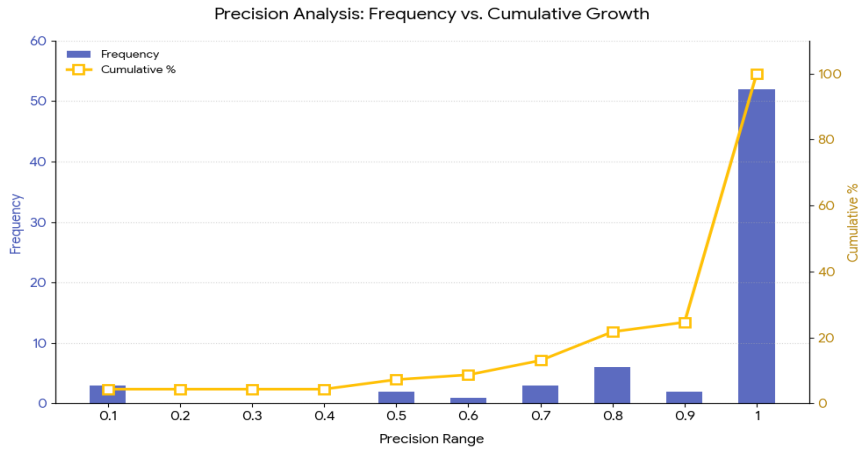

FIG. 9: HISTOGRAM FOR PRESISION MALWARE FAMILIES FOR SVM

Figure 10 shows that around 67% of the families had a recall level between 0.8 and 1.0. Approximately 80% of the families were identified with a recall level of 0.6-1.0.

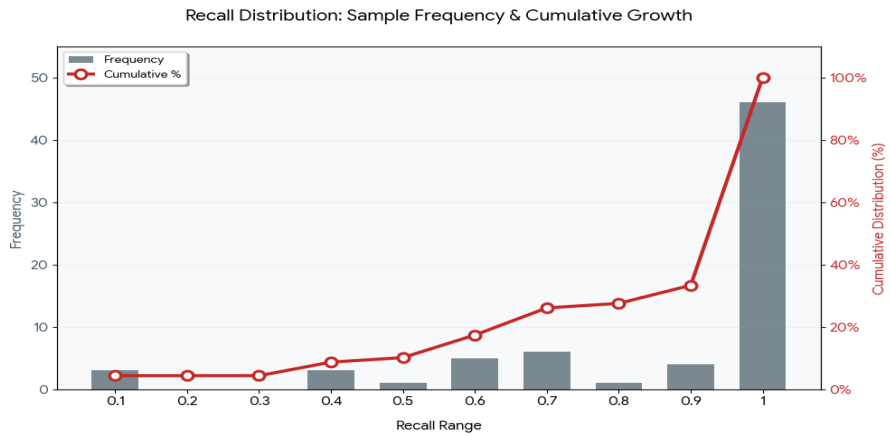

FIG. 10: HISTOGRAM FOR RECALL MALWARE FAMILIES FOR SVM

PART - Figure 11 shows a histogram of the incidence of malware families (71 in total), as well as the precision with which they were categorized using the PART classifier. Approximately 38% of families (including Finspy, Slem bunk, and Fake AV) were categorized with an accuracy level of 0.0-0.1. After investigation, about 45% of the malware families were categorized with an accuracy range of 0.7-1.0.

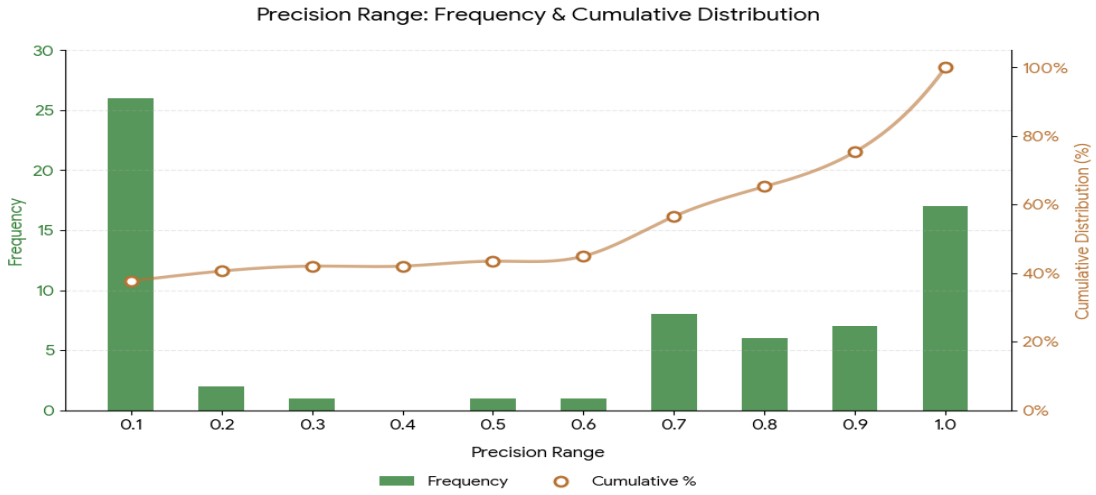

FIG. 11: HISTOGRAM FOR PRESISION MALWARE FAMILIES FOR PART

Figure 12 shows that around 38% of families had a recall level between 0.0 and 0.1. Approximately 57% of families were identified as having a recall level between 0.5 and 1.

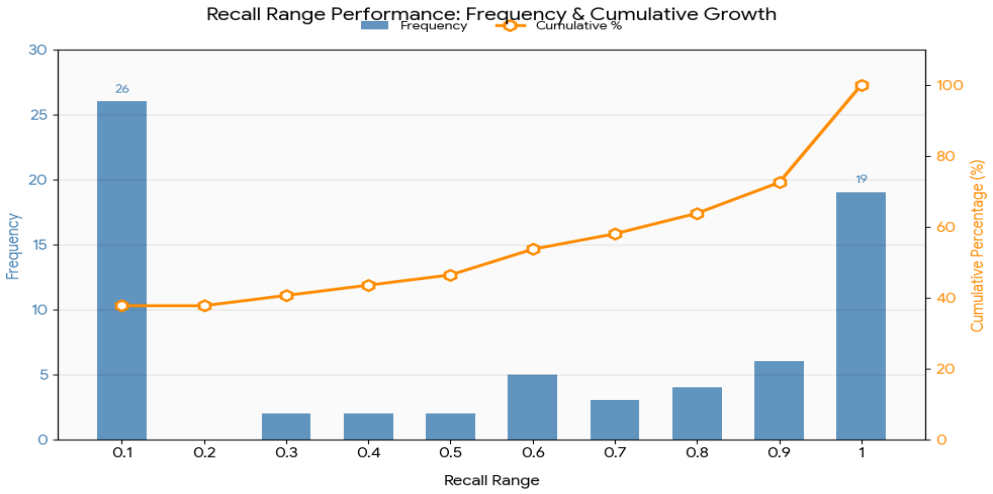

FIG. 12: HISTOGRAM FOR RECALL MALWARE FAMILIES FOR PART

RIDOR- Figure 13 shows a histogram of the incidence of malware families (71 in total), as well as the precision with which they were categorized using the RIDOR classifier. Approximately 75% of families (e.g., Andro RAT, Droid Kung Fu, Gold Dream) were categorized with a precision of 0.9-1.0. After investigation, about 78% of the malware families were categorized with an accuracy range of 0.8-1.0.

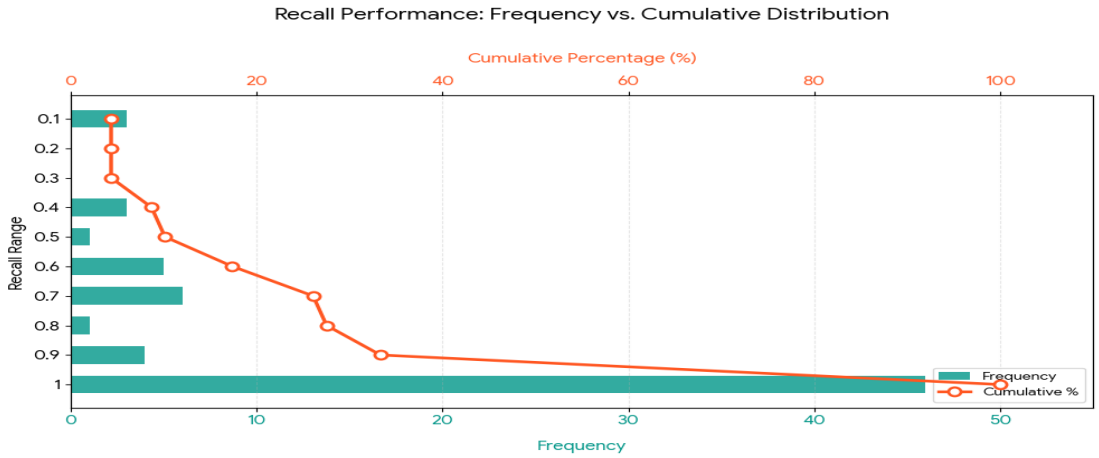

FIG. 13: HISTOGRAM FOR PRESISION MALWARE FAMILIES FOR RIDOR

Figure 14 shows a recall level of 0.9-1.0 for about 67% of all families. Approximately 80% of the families were identified with a recall level of 0.6-1.0.

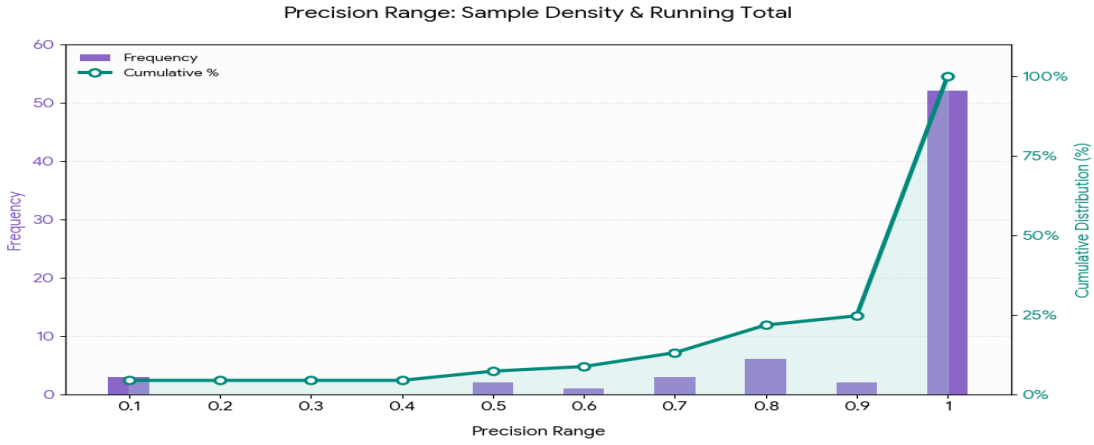

FIG. 14: HISTOGRAM FOR RECALL MALWARE FAMILIES FOR RIDOR

According to the results, MLP is the best method for the dataset, with an accuracy of 90.69%, while RIDOR has the lowest accuracy (82.61%). Figure 15 illustrates the accuracy of various separate classifiers.

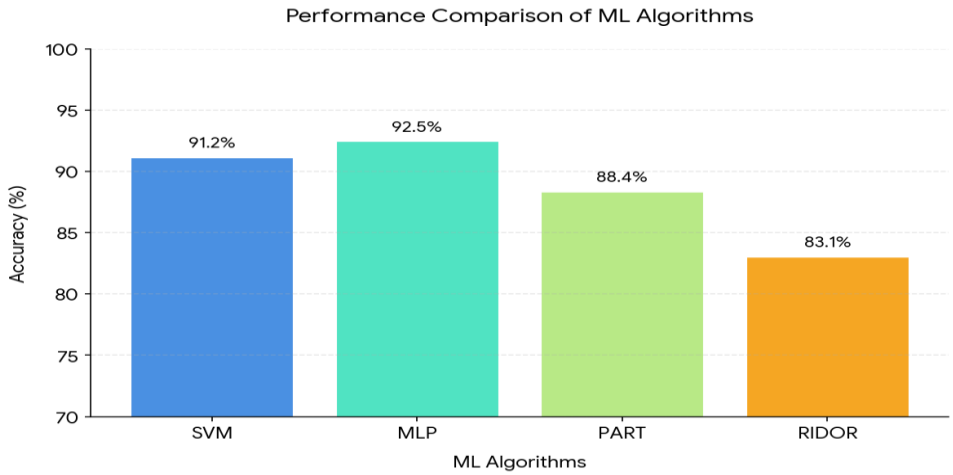

FIG. 15: ACCURACY FOR INDIVIDUAL CLASSIFIERS

RESULTS OF PARALLEL CLASSIFIERS

This section was carried out utilizing a parallel model in which the findings (obtained from individual classification) were used to various ensemble approaches. Because this is a multiclass classification issue, just two parameters (maximum voting and averaging probability) were used to assess the effectiveness of the ensemble strategy. These probabilities are derived from individual classifiers. There were 71 malware families discovered. The two ensemble approaches utilized are as follows:

Maximum Vote (MaxVote): The maximum vote ensemble approach determines the predicted class by taking the maximum vote from many prediction classifiers, and the class with the greatest mode value is chosen as the final predicted class.

Averaging Probabilities (AvgProb): The averaging ensemble approach uses the average of the classifiers' estimated probability to anticipate the ensemble technique's result.

According to the findings obtained using various ensemble strategies, the MVote ensemble methodology is more accurate (93.90%), whereas the Avg Prob ensemble strategy has an accuracy of 80.63%, as seen in Figure 16.

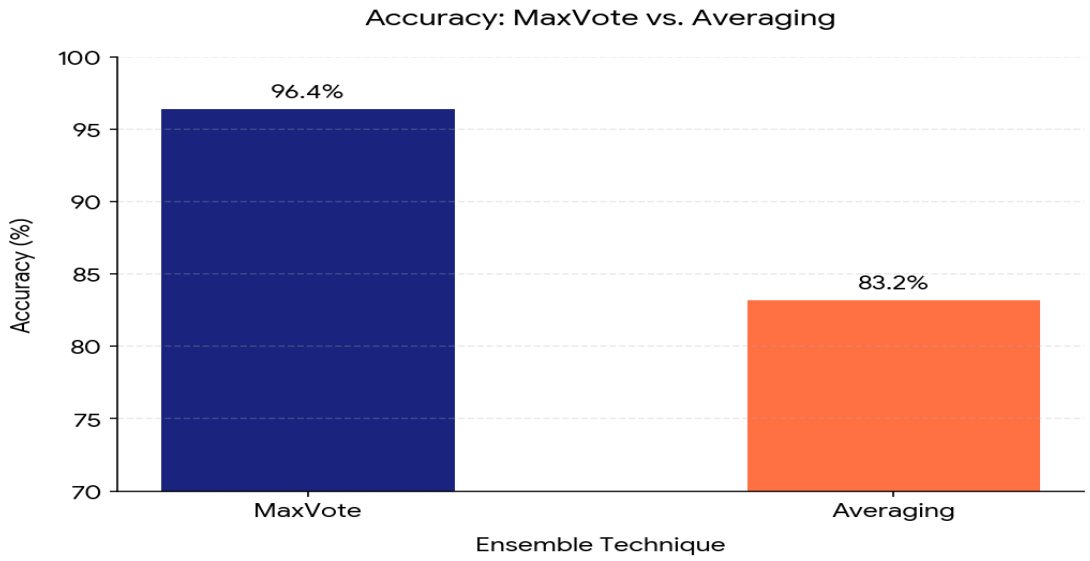

FIG. 16: ACCURACY FOR ENSEMBLE CLASSIFIERS

CONCLUSION

Software applications are used in a variety of fields, including finance, education, health care, transportation, and a wide range of smart devices, such as mobile phones and tablets. Mobile applications have grown popular owing to their ease. Android mobile OS is a popular mobile platform for smart devices. As the app industry grows, so do security worries about smart devices. Malicious software or malware samples have increased significantly throughout the years. Every day, new types of malware are created to carry out malicious acts such as stealing personal information, leaking data, and so on. Therefore, it is necessary to develop resilient, scalable, and economical solutions that can manage three problems: (i) Malware detection - differentiating harmful from benign apps; (ii) Malware family classification - categorizing malware samples into recognized families. (iii) Malware vulnerability mapping entails mapping malware to exploitable vulnerabilities and analyzing those vulnerabilities at the Android architecture level. The approach suggested in the study employs individual classifiers, specifically MLP, SVM, PART, and RIDOR, and combines them to provide a more accurate and efficient result. This technology not only identifies malware in the Android OS with 98.27% accuracy, but it also addresses the gaps and limitations of earlier approaches. This technique takes use of each classifier's skills and strengths while minimizing the impact of its limitations. As a result, the whole process is more resilient and produces fewer errors. Android malware is identified using ensemble parallel classifiers in machine learning. The aforementioned work produces feature vectors including the APK's features, including permissions, libraries utilized,

services, broadcast receivers, and version numbers. In addition to the static attributes, the dataset includes the class of malware family that will be predicted for a certain APK.

REFERENCES

1. Skybox Security, "Vulnerability and threat trends report 2020," Available Online: https://www.skyboxsecurity.com/trends-report/. Accessed: 9 Nov, 2020
2. Statcounter GlobalStats, "Mobile Operating System Market Share Worldwide," Available Online: https://gs.statcounter.com/os-market-share/mobile/worldwide/. Accessed: 5 May 2020
3. Surendran, R., Thomas, T., & Emmanuel, S., "GSDroid: Graph Signal Based Compact Feature Representation for Android Malware Detection," Expert Systems with Applications, v.159, p.113581, 2020.
4. Qamar, A., Karim, A., & Chang, V., "Mobile Malware Attacks: Review, Taxonomy & Future Directions," Future Generation Computer Systems, v. 97, pp. 887-909, 2019.
5. Lee, S., (2013) "Assessment of malicious applications using permissions and enhanced user interfaces on Android," In IEEE International Conference on Intelligence and SecurityInformatics, pp. 270-270, 2013.
6. Shrestha, B., Ma, D., Zhu, Y., Li, H., & Saxena, N. (2015), "Tap-wave-rub: Lightweight human interaction approach to curb emerging smartphone malware," IEEE Transactions on Information Forensics and Security, v. 10, n. 11, pp. 2270-2283, 2015.
7. Cai, H., Meng, N., Ryder, B., & Yao, D. D. (2016), "Droidcat: Unified dynamic detection of Android malware," Informal Report: Virginia Polytechnic Institute & State University, USA, 2016.
8. Karbab, E. B., Debbabi, M., Alrabaee, S., & Mouheb, D. (2016), "DySign: dynamic fingerprinting for the automatic detection of Android malware," In 11th International Conference on Malicious and Unwanted Software, pp. 1-8, 2016.
9. Cho, T., & Seo, S. H., (2016) "A Strengthened Android Signature Management Method," KSII Transactions on Internet & Information Systems, v. 9, n. 3, 2016.
10. Tam, K., Feizollah, A., Anuar, N. B., Salleh, R., & Cavallaro, L. (2017), "The Evolution of Android Malware and Android Analysis Techniques," ACM Computing Surveys, v. 49, n. 4, pp. 1-41, 2017.
11. Wang, S., Yan, Q., Chen, Z., Yang, B., Zhao, C., & Conti, M. (2017), "Detecting Android malware leveraging text semantics of network flows," IEEE Transactions on Information Forensics and Security, v. 13, n. 5, pp. 1096-1109, 2017.
12. Shen, F., Del Vecchio, J., Mohaisen, A., Ko, S. Y., & Ziarek, L. (2018), "Android malware detection using complex-flows," IEEE Transactions on Mobile Computing, v. 18, n. 6, pp. 1231-1245, 2018.
13. Garcia, J., Hammad, M., & Malek, S. (2018), "Lightweight, obfuscation-resilient detection and family identification of Android malware," ACM Transactions on Software Engineering and Methodology, v. 26, n. 3, pp. 1-29, 2018.
14. Zhang, L. (2018), "Smartphone App Security: Vulnerabilities and Implementations," Ph.D. Thesis, University of Michigan-Dearborn, USA, 2018.
15. Talal, M., Zaidan, A.A., Zaidan, B.B., Albahri, O.S., Alsalem, M.A., Albahri, A.S., Alamoodi, A.H., Kiah, M.L.M., Jumaah, F.M. and Alaa, M. (2019), "Comprehensive review and analysis of anti-malware apps for smartphones," Telecommunication Systems, v. 72, n. 2,pp.285-337, 2019.

16. Karbab, E. B., & Debbabi, M. (2019), "MalDy: Portable, data-driven malware detection using natural language processing and machine learning techniques on behavioral analysis reports," Digital Investigation, v. 28, pp. S77-S87, 2019.
17. Peynirci, G., Eminağaoğlu, M., & Karabulut, K. (2020), "Feature Selection for Malware Detection on the Android Platform Based on Differences of IDF Values," Journal of Computer Science and Technology, v. 35, n. 4, pp. 946-962, 2020.
18. Dhalaria, M., & Gandotra, E. (2021). Android malware detection techniques: A literature review. Recent Patents on Engineering, 15(2), 225-245.
19. Kouliaridis, V., & Kambourakis, G. (2021). A comprehensive survey on machine learning techniques for android malware detection. Information, 12(5), 185.
20. Yadav, C. S., & Gupta, S. (2022). A review on malware analysis for iot and android system. SN Computer Science, 4(2), 118.
21. Elsersy, W. F., Feizollah, A., & Anuar, N. B. (2022). The rise of obfuscated Android malware and impacts on detection methods. PeerJ Computer Science, 8, e907.
22. Tang, L., Chen, X., Wen, S., Li, L., Grobler, M., & Xiang, Y. (2023). Demystifying the evolution of android malware variants. IEEE Transactions on Dependable and Secure Computing, 21(4), 3324-3341.
23. Faruki, P., Bhan, R., Jain, V., Bhatia, S., El Madhoun, N., & Pamula, R. (2023). A survey and evaluation of android-based malware evasion techniques and detection frameworks. Information, 14(7), 374.
24. Sharma, M., & Kaul, A. (2024). A review of detecting malware in android devices based on machine learning techniques. Expert Systems, 41(1), e13482.
25. Mohd Saudi, M., Husainiamer, M. A., Ahmad, A., & Idris, M. Y. I. (2024). iOS mobile malware analysis: a state-of-the-art. Journal of Computer Virology and Hacking Techniques, 20(4), 533-562.
26. Dahiya, A., Singh, S., & Shrivastava, G. (2025). Android malware analysis and detection: A systematic review. Expert Systems, 42(1), e13488.
27. Almarri, S., Bodokhi, A., & Frikha, M. (2025). A Review of the Recent Trends in Mobile Malware Evolution, Detection, and Analysis. IEEE Access.
28. Chimeleze, C. U., Al-Sharafi, M. A., & Jamil, N. (2026). A comprehensive review of Android malware: trends, behaviors, taxonomies, and future direction. PeerJ Computer Science, 12, e3312.
29. Google playstoreapps. Available Online: https://play.google.com/store/apps/. Accessed: 15 July 2023.
30. Wandoujiaapps. Available Online: http://www.wandoujia.com/apps/. Accessed: 15 July 2023.
31. Wei, F., Li, Y., Roy, S., Ou, X., & Zhou, W. (2017), "Deep ground truth analysis of current Android malware," In International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, pp. 252-276, 2017.
32. Allix, K., Bissyandé, T. F., Klein, J., & Le Traon, Y. (2016), "Androzoo: Collecting millions of Android apps for the research community," In IEEE/ACM 13th Working Conference on Mining Software Repositories, pp. 468-471, 2016.
33. Yan, S., Khan, F.N., Mavromatis, A., Gkounis, D., Fan, Q., Ntavou, F., Nikolovgenis, K., Meng, F., Salas, E.H., Guo, C. and Lu, C. (2017), "Field trial of machine-learning-assisted and SDNbased optical network planning with network-scale monitoring database," In European Conference on Optical Communication, pp. 1-3, 2017.