International Journal of Trend in Scientific Research and Development (IJTSRD)
Special Issue on Advancements and Emerging Trends in Computer Applications - Innovations, Challenges, and Future Prospects
Available Online: www.ijtsrd.com e-ISSN: 2456 – 6470

Android Mobile Malware Detection using Machine Learning: A Systematic Review

Shiva Majumdar

PG Student, Department of Computer Application, G. H. Raisoni University, Amravati, Maharashtra, India

ABSTRACT

With the increasing use of mobile devices, malware attacks are rising, especially on Android phones, which account for 72.2% of the total market share. Hackers try to attack smartphones with various methods such as credential theft, surveillance, and malicious advertising. Among numerous countermeasures, machine learning (ML)-based methods have proven to be an effective means of detecting these attacks, as they are able to derive a classifier from a set of training examples, thus eliminating the need for an explicit definition of the signatures when developing malware detectors. This paper provides a systematic review of ML-based Android malware detection techniques. It critically evaluates 106 carefully selected articles and highlights their strengths and weaknesses as well as potential improvements. Finally, the ML-based methods for detecting source code vulnerabilities are discussed, because it might be more difficult to add security after the app is deployed. Therefore, this paper aims to enable researchers to acquire in-depth knowledge in the field and to identify potential future research and development directions.

With the swift adoption of Android devices all over the world, it has become a major target for attacks from malware which brings great danger to both users and organizations. With the growing complexity of modern malware, traditional methods of signature-based malware detection are painfully sluggish. It is for this reason that there is recent notice towards the use of ML in the classification and detection of malware.

This systematic review monitors new developments in the detection of Android malware through the use of machine learning techniques. We cover different methodologies, such as static, dynamic, and hybrid analysis, reviewing their advantages and disadvantages. We also focus on research feature extraction methods, classification algorithms, and available datasets. Adverse challenges such as adversarial dataset, model quality, and interpretability are discussed as well.

Our results outline the effectiveness of deep learning models, specifically convolutional neural networks (CNNs) and recurrent neural networks (RNNs), for the purposes of malware detection. Even so, high false positive rates and overhead computation are still open challenges. The review documents the most recent developments, determines under-researched areas, and proposes ways of improving the literature to aid in the development of Android malware detection systems with machine learning techniques.

KEYWORDS: *Android Security, Malware Detection, Code Vulnerability, CNN.*

I. INTRODUCTION

In this technological era, smartphone usage and its associated applications are rapidly increasing due to the convenience and efficiency in various applications and the growing improvement in the hardware and software on smart devices. As of May 2021, its market share was 72.2%. The second highest market share of 26.99% is owned by Apple iOS, while the rest of the 0.81% is shared among Samsung, Kai OS, and other small vendors. Google Play is the official app store for Android-based devices. The number of apps published on it was over 2.9 million as of May 2021. Of these, more than 2.5 million apps are classified as regular apps, while 0.4 million apps are classified as low-quality apps by AppBrain. Android's worldwide popularity makes it a more attractive target for cybercriminals and is more at risk from malware and viruses. Studies have proposed various methods of detecting these attacks, and ML is one of the most prominent techniques among them. This is because ML techniques are able to derive a classifier from a (limited) set of training examples. The use of examples thus avoids the need to explicitly define signatures in developing malware detectors. Defining signatures requires expertise and tedious human involvement and for some attack scenarios explicit rules (signatures) do not exist, but examples can be obtained easily. Numerous industrial and academic research has been carried out on ML-based malware detection on Android, which is the focus of this review paper.

The taxinomical classification of the review is presented in Figure 1. Android users and developers are known to make mistakes that expose them to unnecessary dangers and risks of infecting their devices with malware. Therefore, in addition to malware detection techniques, methods to identify these mistakes are important and covered in this paper (see Figure 1). Detecting malware with ML involves two main phases, which are analyzing Android Application Packages (APKs) to derive a suitable set of features and then training machine and deep learning (DL) methods on derived features to recognize malicious APKs. Hence, a review of the methods available for APK analysis is included, which consists of static, dynamic, and hybrid analysis. Similar to malware detection, vulnerability detection in software code involves two main phases, namely feature generation through code analysis and training ML on derived features to detect vulnerable code segments. Hence, these two aspects are included in the review's taxonomy.

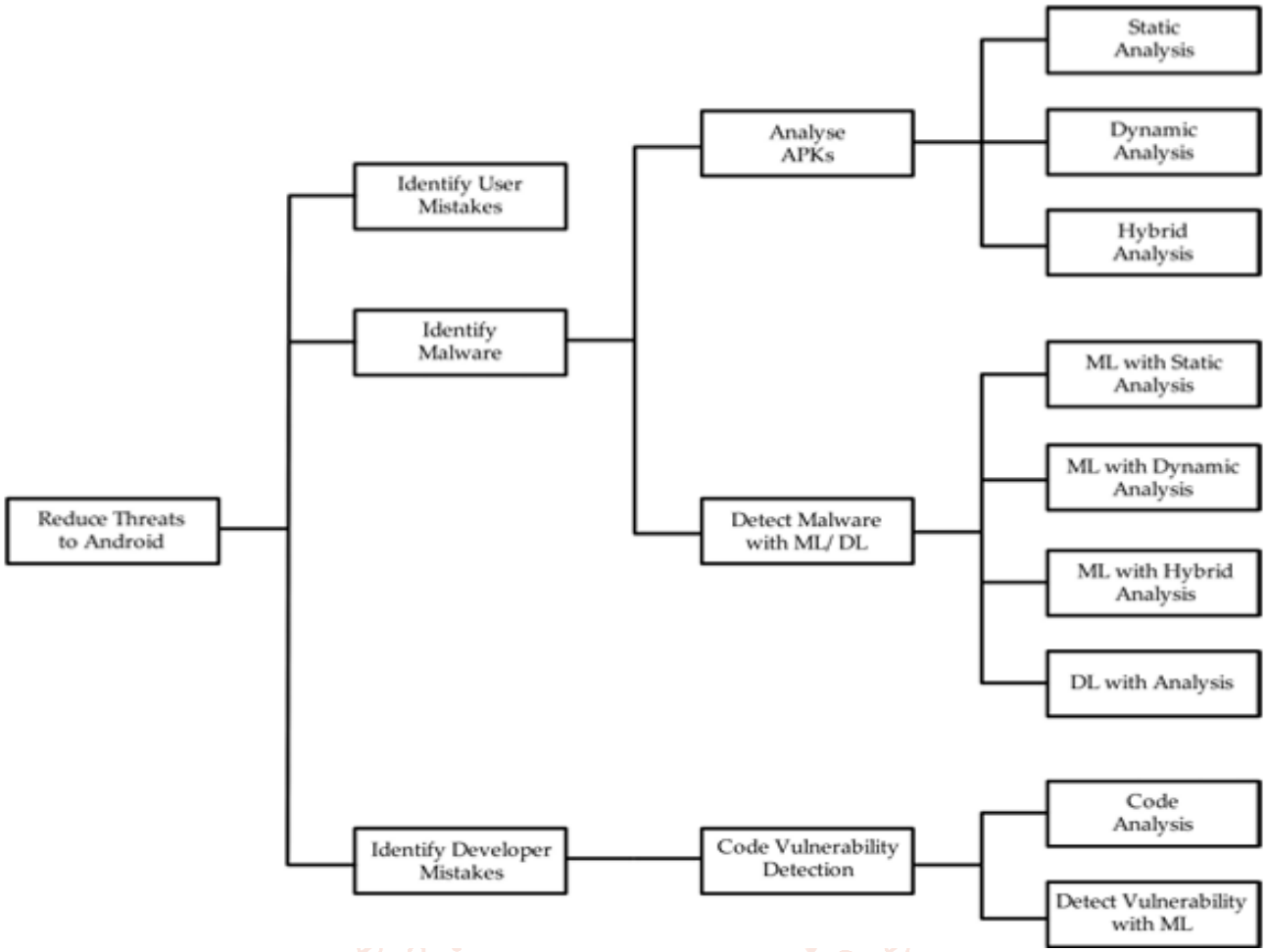

Figure 1.Taxonomy of the review

II. RELATED WORK

Previous reviews in [9,13,17,34–37] discussed various ML-based Android malware detection techniques and ways to improve Android security. The review in systematically reviewed the studies conducted in static analysis techniques used for Android applications from 2011 to 2015. The tools that can be used to perform Android code analysis using static analysis techniques were also summarised. Abstract representation, taint analysis, symbolic execution, program slicing, code instruct mentation, and type/model checking were identified as fundamental analysis methods. Though this review correctly identified the most widely used approach to detect privacy and security related issues, the applicability of static analysis techniques for malware detection was not discussed. Apart from that, it did not take into account the recent research where novel analysis methods and malware detection methods were suggested. The study conducted in provided a good systematic review mainly about static analysis techniques that can be used in Android malware detection. Four methods were identified as characteristic-based, opcode-based, program graph-based and symbolic execution-based. After that, it evaluated the capabilities of static analysis-based Android malware detection methods on those four methods using the existing literature. The paper has identified ML and statistical models as possible methods by which Android malware can be identified. However, ML-based machine learning methods have not been thoroughly reviewed as the main focus is only on the static analysis techniques.

In , a survey was carried out using existing literature upto 2017 to identify malware detection techniques together with their advantages and disadvantages. Under static and dynamic analysis, they have grouped several approaches that can be used to identify Android malware. However, the analysis of this survey was not comprehensive as it focused on a limited number of studies. Based on the previous studies, a systematic review was conducted in. According to it, there are five types of Android malware detection techniques. They are static detection, dynamic detection, hybrid detection, permission-based detection, and emulation-based detection. They also summarised the reviewed work with the model accuracy of malware detection, but the approach of those studies was not discussed. The review conducted in analysed several studies conducted until 2019 related to ML models which can be used to detect Android malware. The malware and APK analysis methods were not discussed in detail since the focus on identifying different ML models was the priority in this review. It is better to analyse the accuracies of the identified ML models. The novel ML/DL and other models which can be used to detect Android malware were also not in the focus of this review. The review in provides a good analysis of static, dynamic, and hybrid detection techniques used in the existing research studies for Android malware detection. Along with that possibility of using machine learning models, several deep learning models are also discussed. However, this study did not comprehensively analyse the model accuracy of the machine learning methods for Android malware detection since this study focused more on discussing different malware detection approaches instead of considering the accuracy of those approaches. Hence, these works differ from our study.

However, several limitations have been identified in the above works, such as not covering recent proposals on ML methods to detect malware, narrow scopes, and lack of critical appraisals of suggested detection methods. The lack of a thorough analysis of ML/DL-based methods was also identified as a limitation of existing works. Android malware detection and Android code vulnerability analysis have a lot in common. ML methods used in one task can be customised for use in the other task. However, as per our understanding, there are no reviews that cover these two areas together. These shortcomings have been addressed in this work and therefore our work is unique.

In, a systematic review on DL-based methods for Android malware defence was discussed. Malware detection, malware family detection, repackaged/fake app detection, adversarial learning attacks and protections, and malicious behaviour analysis were identified as the malware defines objectives in this review together with the usage of DL models. Though they have identified the possible DL models, it is still better to analyse the accuracy and compare it with traditional ML methods and other hybrid approaches.

III. DATA AND SOURCES OF DATA

This section provides a high-level overview of the Android architecture and its built-in security as well as potential threat vectors for Android. It also provides an introduction to the Process as it would be useful for non-ML background readers to understand the contents of this paper.

3.1. Threats to Android

While Android has good built-in security measures, there are several design weaknesses and security flaws that have become threats to its users. Awareness about those threats is also important to perform a proper malware detection and vulnerability analysis. Many research and technical reports have been published related to the Android threats and classified Android threats based on the attack methodology. Social engineering attacks, physical access retrieving attacks, and network attacks are described under the ways of gaining access to the device. For the vulnerabilities and exploitation methods, manin the middle attacks, return to libc attacks, JIT-Spraying attacks, third-party library vulnerabilities, Dalvik vulnerabilities, network architecture vulnerabilities, virtualization vulnerabilities, and Android debug bridges and kernel vulnerabilities are considered. The survey in identified four types of attacks to Android; hardware-based attacks, kernel-based attacks, Hardware Abstraction Layer (HAL) based attacks, and application based attacks. Hardware-based attacks such as Row hammer, Glitch, and Drammer are related to sensors, touch screens, communication media, and DRAM. Kernel-based attacks such as Gooligan, Droid Kungfu, Return-oriented Programming are related to Root Priv ilege, Memory, Boot Loader, and Device Driver. HAL-based attacks such as Return to User and TocTou are related to interfaces for cameras, Bluetooth, Wi-Fi, Global Positioning System (GPS), and Radio. Application-based attacks such as Ad Detect, WuKong, and LibSift are related to third-party libraries, Intra-Library collusion, and privilege escalations.

Android applications are easily penetrable with proper knowledge of Android pro gramming if suitable security mechanisms are not in place. In addition, Android mar ketplaces such as Google Play are not following extensive security protocols when new apps are published. For example, the Android game known as Angry Bird was hacked and the hacker managed to get into its APK file and embed a malicious code that sent text messages unknowingly by the user. The cost was 15 GPB to the user per message. More than a thousand users were affected.

3.2. Machine Learning Process

ML is a branch of artificial intelligence that focuses on developing applications by learning from data without explicitly programming how the learned tasks are performed. The traditional ML methods make predictions based on past data. ML process lifecycle consists of multiple sequential steps. They are data extraction, data preprocessing, feature selection, model training, model evaluation, and model deployment. Supervised learning, unsupervised learning, semi supervised learning, reinforcement learning, and deep learning are the different subcategories of ML. The supervised learning approach uses a labelled dataset to train the model to solve classification and regression problems depend on the output variable type (continuous or discreet). Unsupervised learning is used to identify the internal structures (clusters), the characteristics of a dataset, and a labelled dataset is not required to train the model. A mix of both supervised and unsupervised learning techniques are applied in semi supervised learning and used in a case of limited labelled data in the used dataset. The learning model and the data used for training are inferred. The model parameters are updated with the received feedback from the environment in reinforcement learning where no training data is involved. This ML method proceeds as prediction and evaluation cycles. DL is defined as learning and improving by analysing algorithms on their own. It works with models such as artificial neural networks (ANN)and consists of a higher or deeper number of processing layers.

3.3. Android Architecture

Android is built on top of the Linux Kernel. Linux is chosen because it is open source, verifies the pathway evidence, provides drivers and mechanisms for networking, and manages virtual memory, device power, and security . Android has a layered architecture . The layers are arranged from bottom to top. On top of the Linux Kernal Layer, the Hardware Abstraction Layer, Native C/C++ Libraries and Android Runtime, Java Application Programming Interface (API) Framework, and System Apps are stacked on top of each. Each layer is responsible for a particular task. For example, the Java API Framework provides Java libraries to perform a location awareness application-related activity such as identifying the latitude and the longitude. Android-based applications and some system services use the Android Runtime (ART). Dalvik was the runtime environment used before the ART. Both ART and Dalvik were created for the Android applications-related projects. The ART executes the Dalvik Executable (DEX) format and the bytecode specification . The other aspects are mem Ory management and power management since the Android-based applications run on battery-powered devices with limited memory. Therefore, the Android operating system is designed in a way that any resource can be well managed . For instance, the Android OS will automatically suspend the application in memory if an application is not in use at the moment. This state is known as the

running state of the application life cycle. By doing this, it can preserve the power that can be utilised when the application reopens. Otherwise, the applications are kept idle until they are closed . However, there are possibilities of malware attacks to exploit some vulnerabilities in the applications developed by various users, because the Google Play Store will not detect some vulnerabilities when publishing applications in the Play Store as in Apple App Store.

IV. RESEARCH METHODOLOGY

Android was first released in 2008. A few years later, the security concerns were discussed with the increasing popularity of Android applications. More attention was received towards applying ML for software security in the last five years because many researchers continuously identify and propose novel ML-based methods. This review was conducted according to the Preferred Reporting Items for Systematic reviews and Meta-Analysis (PRISMA) model . Based on the objective of this study, first we formulated several research questions. Next, a search strategy was defined to identify the conducted studies which can be used to answer our research questions. The database usage and inclusion and exclusion criteria were also defined at this stage. The study selection criteria were defined to identify the studies aiming to answer the formulated research questions as the third stage. The fourth stage is defined as data extraction and synthesis, which describes the usage of the collected studies to analyse for providing answers to the research questions. We reviewed threats to the validity of the review and the mechanism to reduce the bias and other factors that could have influenced the outcomes of this study as the last step of the review process.

4.1. Data Extraction and Synthesis

We extracted data from 9 studies to answer the RQ1, which is about the existing literature reviews related to Android malware detection using ML/DL models and Android vulnerability analysis. To map with RQ2, related studies were identified related to Android code/APK analysing techniques that can be used to analyse malware. The count for those studies was 22. To answer the RQ3 about ML/DL based techniques which can be used to detect malware, we extracted data from 18 different studies. Data from 36 research studies were extracted to find answers for the RQ4, which is about detection model accuracy, strengths, and weaknesses. The remaining 21 papers about Android source code vulnerability analysis and detection methods were used to answer the RQ5.

4.2. Research Questions

This systematic review aims to answer the following research questions.
RQ1: What are the existing reviews conducted in ML/DL based models to detect Android malware and source code vulnerabilities?
RQ2: What are code/APK analysing methods that can be used in malware analysis?
RQ3: What are the ML/DL based methods that can be used to detect malware in Android?
RQ4: What are the accuracy, strengths, and limitations of the proposed models related to Android malware detection?
RQ5: Which techniques can be used to analyse Android source code to detect vulnerabilities?

4.3. Search Strategy

The search strategy involves the outline of the most relevant bibliographic sources and search terms. In this review, we have used several top research repositories as main sources to identify studies. They were ACM Digital Libraries, IEEEXplore Digital Library, Science Direct, Web of Science and Springer Link. Google Scholar, and Research Gate were also used to identify research studies published in some quality venues. The search string that we used to browse through research repositories contained the following search terms: (“android malware”) OR (“malware detection”) OR (“machine learning”) OR (“deep learning”) OR (“static analysis”) OR (“dynamic analysis”) OR (“hybrid analysis”) OR(“malware analysis”) OR (“android vulnerability analysis”) OR (“ML based malware detection”) OR (“DL based malware detection”).

4.4. Threats to Validity of the Review

This review was conducted in a systematic approach explained above. We tried to minimise the bias and the other factors affecting the review study. Though we have conducted our review comprehensively, still there can be good papers which were not reviewed in this study since they are not available in the research repositories that we used. The period we were considering for the paper selection is from 2016 to May 2021, as the use of ML techniques for malware detection has increased significantly during this period due to recent advances in artificial intelligence. Therefore, if comprehensive studies were conducted before that, those studies were not captured in our work. When searching for the papers we considered the research papers written in the English language. Because of this limitation, our work may have overlooked some important works written in other languages such as Chinese, German, and Spanish.

4.5. Study Selection Criteria

Since mobile malware detection using ML techniques related trends increased from 2016, we limit our review to study related work from 2016 to May 2021. Initially through the research database search in the top research repositories, 109 research papers and from another sources 11 research papers were identified. From these 120 papers, 5 were excluded because of duplicate entries and another 5 were excluded because they were not available in public from those 110 articles. Due to data analysis issues and experiment issues in the given context, 4 articles were excluded though the full text is available. The remaining 106 articles were reviewed in this study. We performed the snowballing process, considering all the references presented in the retrieved papers and evaluating all the papers referencing the retrieved ones, which resulted in two additional relevant paper. We applied the same process as for the retrieved papers. The snowballing search was conducted in March 2021. Figure 2 shows a summary of the paper selection method for this systematic review.

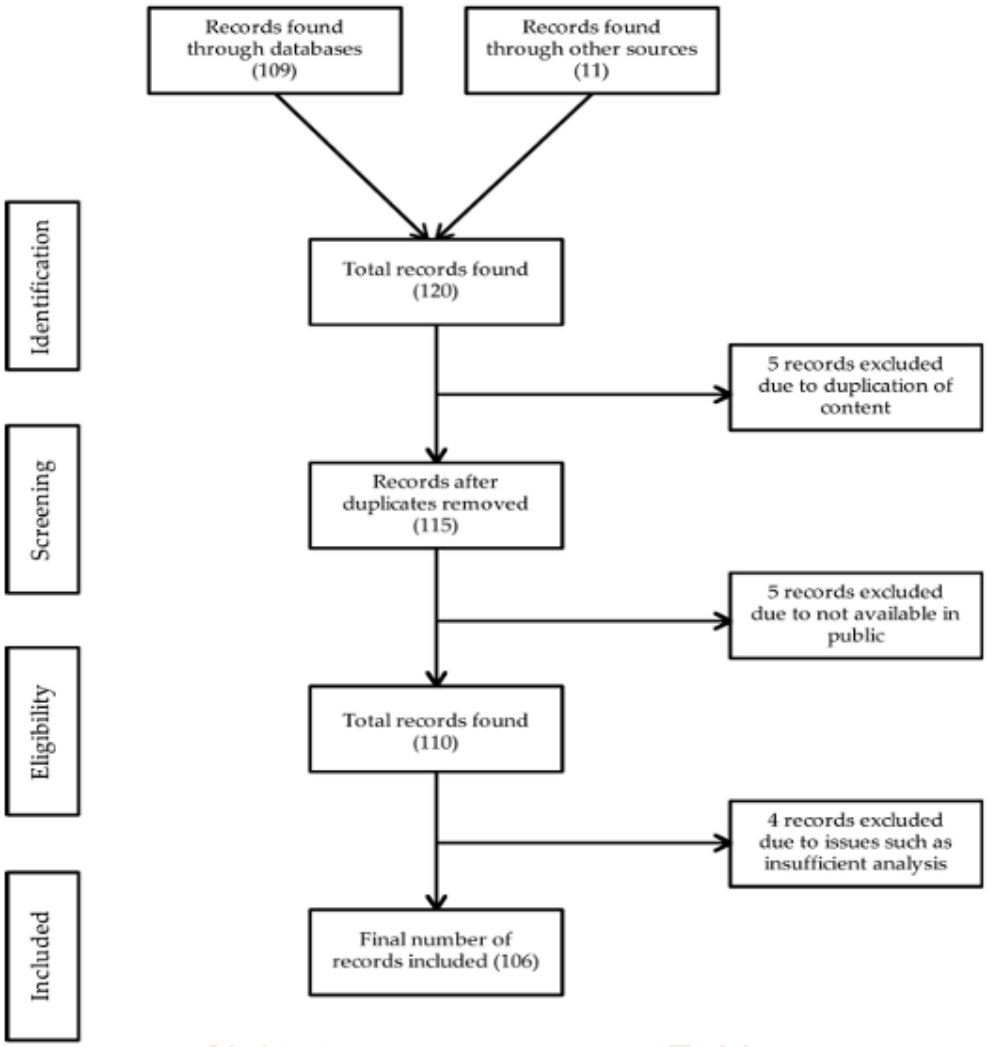

Figure 2. PRISMA method: collection of papers for the review.

V. RESULTS AND DISCUSSION

Based on the reviewed studies in ML/DL based methods to detect malware, it is identified that 65% of studies related to malware detection techniques used static analysis, 15%useddynamicanalysis, and the remaining 20% followed the hybrid analysis technique. This is illustrated in Figure 3. This high attractiveness of static analysis may be due to the various advantages associated with it over dynamic analysis, such as ability to detect more vulnerabilities, localising vulnerabilities, and offering cost benefits.

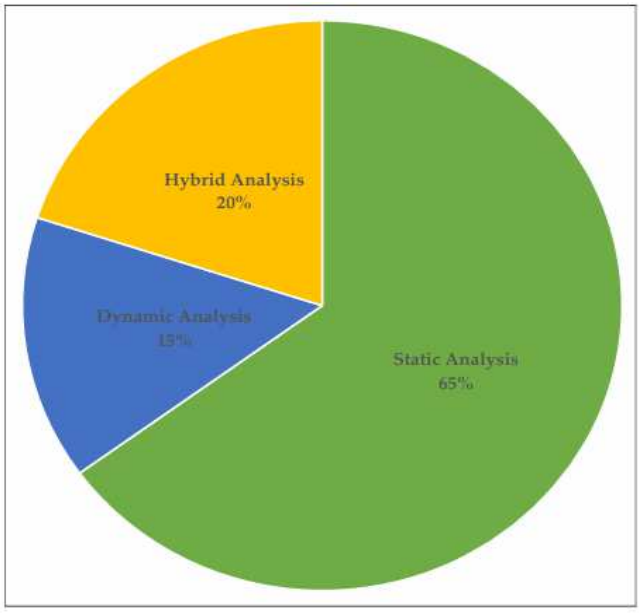

Figure 3. Malware analysis techniques used in the reviewed studies.

Many ML/DL based malware detection studies used the code analysis method as the feature extraction method. Apart from that, manifest analysis and system call analysis methods are the other widely used methods. Illustrates those feature extraction methods used in the reviewed studies. It is possible to detect a substantial amount of mal ware after analysing decompiled source codes rather than analysing permissions or other features. That may be the reason for the high usage of code analysis in malware detection.

By using the feature extraction methods, permissions, API calls, system calls, and opcodes are the most widely extracted features. This is illustrated in along with the other extracted features in the reviewed studies. Many hybrid analysis methods extracted permissions as the feature to perform static analysis. It is easy to analyse permissions when comparing with the other features too. These could be reasons for the high usage of permissions as the extracted feature. Services and network protocols have low usage in feature extractions. The reason for this maybe it is comparatively not easy to analyse those features.

The datasets used in ML/DL based Android malware detection studies to train the algorithms are illustrated in. Drebin was the most widely used dataset in Android Malware Detection, and it was used in 18 reviewed studies. Google Play, Mal Genome, and AMD datasets are the other widely used datasets. The reason for the highest usage of the Drebin dataset may be because it provides a comprehensive labelled dataset. Since Google Play is the official app store of Android, it may be a reason to have high usage for the dataset from Google.The majority of the studies used hybrid analysis and static analysis as the source code analysis techniques In vulnerability detection in Android, as illustrated in Figure4. To perform a highly accurate vulnerability analysis, the source code should be analysed and executed too . Therefore, this may be the reason to have hybrid analysis and static analysis as the widely used source code analysis methods to detect vulnerabilities.

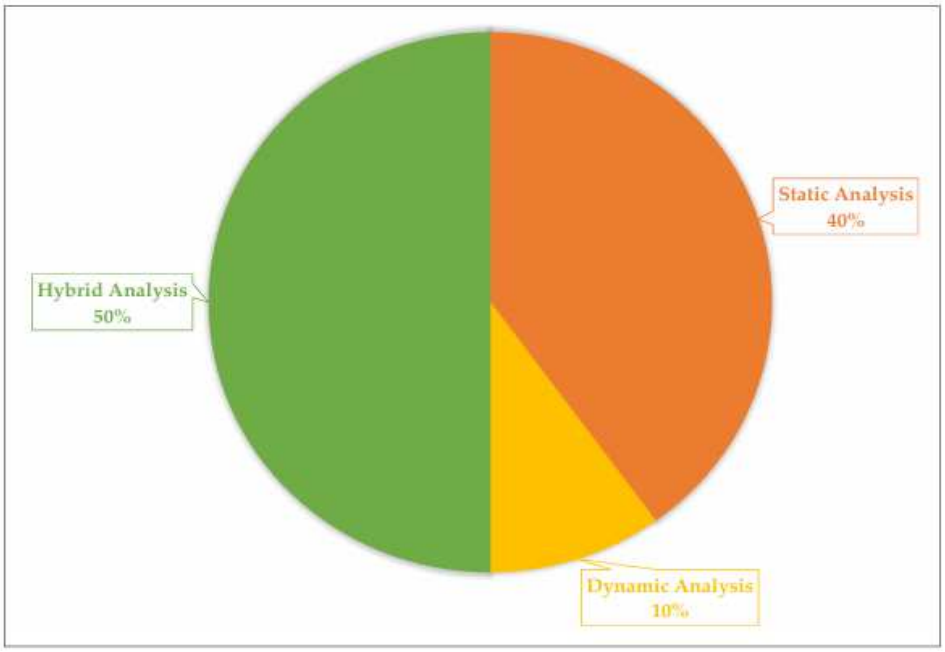

Figure 4. Android source code vulnerability analysis methods.

VI. CONCLUSIONS

Any smartphone is potentially vulnerable to security breaches, but Android devices are more lucrative for attackers. This is due to its open-source nature and the larger market share compared to other operating systems for mobile devices. This paper discussed the Android architecture and its security model, as well as potential threat vectors for the Android operating system. Based upon the available literature, a systematic review of the state-of-the-art ML-based Android malware detection techniques was carried out, covering the latest research from 2016 to 2021. It discussed the available ML and DL models and their performance in Android malware detection, code and APK analysis methods, feature analysis and extraction methods, and strengths and limitations of the proposed methods. Malware aside, if a developer makes a mistake, it is easier for a hacker to find and exploit these vulnerabilities. Therefore, methods for the detection of source code vulnerabilities using ML were discussed. The work identified the potential gaps in previous research and possible future research directions to enhance the security of Android OS. Both Android malware and its detection techniques are evolving. Therefore, we believe that similar future reviews are necessary to cover these emerging threats and their detection methods. As per our findings in this paper, since DL methods have proven to be more accurate than traditional ML models, it will be beneficial to the research community if more comprehensive systematic reviews can be performed by focusing only on DL-based malware detection on Android. The possibility of using reinforcement learning to identify source code vulnerabilities is another area of interest in which systematic reviews and studies can be carried out.

VII. REFERENCES

[1] Number of Mobile Phone Users World wide from 2016 to 2023(In Billions). Available online: https://www.statista.com/statistics/ 330695/number-of-smartphone-users-worldwide/ (accessed on 19 May 2021).

[2] Mobile Operating System Market Share Worldwide. Available online: https://gs.statcounter.com/os-

market-share/mobile/ worldwide/ (accessed on 19 May 2021).

[3] Number of Android Applications on the Google Play Store. Available online: https://www.appbrain.com/stats/number-of android-apps/ (accessed on 19 May 2021).

[4] Gibert, D.; Mateu, C.; Planes, J. The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. J. Netw. Comput. Appl. 2020, 153, 102526.

[5] Khan, J.; Shahzad, S. Android Architecture and Related Security Risks. Asian J. Technol. Manag. Res. [ISSN: 2249–0892] 2015, 5, 14–18. Available online: http://www.ajtmr.com/papers/Vol5Issue2/Vol5Iss2_ P4.pdf (accessed on 19 May 2021).

[6] Platform Architecture. Available online: https://developer.android.com/guide/platform (accessed on 19 May 2021).

[7] Android Runtime (ART) and Dalvik. Available online: https://source.android.com/devices/tech/dalvik (accessed on 19 May2021).

[8] Cai, H.; Ryder, B.G. Understanding Android application programming and security: A dynamic study. In Proceedings of the 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME), Shanghai, China, 17–22 September 2017; pp. 364–375.

[9] Liu, K.; Xu, S.; Xu, G.; Zhang, M.; Sun, D.; Liu, H. A Review of Android Malware Detection Approaches Based on Machine Learning. IEEE Access 2020, 8, 124579–124607.

[10] Gilski, P.; Stefanski, J. Android os: A review. Tem J. 2015, 4, 116. Available online: https://www.temjournal.com/content/41/14/ temjournal4114.pdf (accessed on 19 May 2021).

[11] Privacy in Android 11 | Android Developers. Available online: https://developer.android.com/about/versions/11/p rivacy (accessed on 19 May 2021).

[12] Garg, S.; Baliyan, N. Comparative analysis of Android and iOS from security viewpoint. Comput. Sci. Rev. 2021, 40, 100372.

[13] Odusami, M.; Abayomi-Alli, O.; Misra, S.; Shobayo, O.; Damasevicius, R.; Maskeliunas, R. Android malware detection: A survey. In International Conference on Applied Informatics; Springer: Cham, Switzerland, 2018; pp. 255–266.

[14] Bhat, P.; Dutta, K. A survey on various threats and current state of security in android platform. ACM Comput. Surv. (CSUR) 2019, 52, 1–35.

[15] Tam,K.; Feizollah, A.; Anuar, N.B.; Salleh, R.; Cavallaro, L. The evolution of android malware and android analysis techniques. ACMComput. Surv. (CSUR) 2017, 49, 1–41.

[16] Li, L.; Li, D.; Bissyandé, T.F.; Klein, J.; Le Traon, Y.; Lo, D.; Cavallaro, L. Understanding android app piggybacking: A systematic study of malicious code grafting. IEEE Trans. Inf. Forensics Secur. 2017, 12, 1269–1284.

[17] Ashawa, M.A.; Morris, S. Analysis of Android malware detection techniques: A systematic review. Int. J. Cyber-Secur. Digit. Forensics 2019, 8, 177–187. [CrossRef]

[18] Suarez-Tangil, G.; Tapiador, J.E.; Peris-Lopez, P.; Ribagorda, A. Evolution, detection and analysis of malware for smart devices. IEEE Commun. Surv. Tutor. 2013, 16, 961–987.

[19] Mos,A.; Chowdhury, M.M. Mobile Security: A Look into Android. In Proceedings of the 2020 IEEE International Conference on Electro Information Technology (EIT), Chicago, IL, USA, 31 July–1 August 2020; pp. 638–642

[20] Faruki, P.; Bharmal, A.; Laxmi, V.; Ganmoor, V.; Gaur, M.S.; Conti, M.; Rajarajan, M. Android security: A survey of issues, malware penetration, and defenses. IEEE Commun. Surv. Tutor. 2014, 17, 998–1022.