Static, Dynamic and Intrinsic Feature Based Android Malware Detection Using Machine Learning: A Technical Review

Bilal Ahmad Mantoo[1], **Zafar Ali Khan N**[2]

[1,2]Department of Computer Science and Engineering, Presidency University, Yelahanka, Bangalore, India.

Abstract: The emergence of smart devices in the market leads to exponential growth of malware in the market posing a significant challenge to smart device users. These malicious programs are designed with advanced techniques to evade existing detection techniques, infiltrate systems, and cause harm to any platform. One such platform is Android, the open-source smartphone operating system which has experienced exponential growth since its inception. However, this progress has been increased by the growing threat of Android malware, which exploits smartphones to carry out malicious acts. These malware employs a plethora of techniques to circumvent detection systems, presenting novel obstacles to reliable detection. Currently, Android malware detection approaches can be broadly classified into two categories, signature- based detection and machine learning-based detection. Signature-based detection relies on patterns or signatures of malware to identify and block malicious software. Nevertheless, this approach is subject to limitations, as it inadequately detects novel or un- known malware variants. To address the limitations of signature-based detection, researchers and anti-malware firms have turned to machine learning-based detection techniques. These methods harness the power of machine learning algorithms to analyze and categorize applications based on their behavioral patterns, intrinsic features, or other distinctive characteristics. By assimilating knowledge from extensive datasets comprising known malware and legitimate applications, machine learning models can identify previously unseen malware by identifying similarities to known malevolent behavior. This study aims to disseminate the current landscape of machine learning-based Android malware detection techniques and undertake a parametric comparison of their efficacy. The objective is to explore a large number of detection methods and elucidate prospective avenues in this domain. By scrutinizing and contrasting these approaches, we can gain profound insights into the strengths and limitations of various machine learning techniques, while identifying potential areas for further research and enhancement.

Keywords: Malicious Programs, Android, Malware, Signature Based Detection, Machine Learning, Behavioral Patterns.

1. INTRODUCTION.

Android is an operating system designed for most of the smart devices like phone, smart television, smart watches etc. Its global market is very high and is in par of the rest of operating system like mac operating system. Its flexibility, customizability, and vast ecosystem of apps have contributed to its popularity among both users and developers. Android provides a rich set of features and capabilities, allowing users to perform various tasks, including communication, web browsing, multimedia consumption, gaming, and productivity and supports wide range of hardware devices. One of the key strengths of Android is its app ecosystem. The Google Play Store offers millions of applications that cater to diverse user needs and preferences. From social networking and entertainment to education and productivity, Android apps cover a broad spectrum of categories. This extensive app ecosystem has fueled innovation and transformed the way people interact with their mobile devices. Smartphones have become indispensable tools in modern life, providing users with a diverse array of functionalities and services readily available at their fingertips. Their portability and versatility have made tasks easier and more accessible, as users can seamlessly access communication, entertainment, productivity, and utility apps from a single device. Moreover, smartphones have replaced numerous traditional gadgets, such as cameras, calculators, and alarm clocks, consolidating multiple functions into a compact and portable form factor [1]. Among the various mobile operating systems, Android dominates the market, boasting a market share of 72.2 percent as of May 2021. As of the latest statistics, Apple's iOS holds a market share of approximately 26.99%, positioning it as the second-largest mobile operating system globally. The remaining 0.81% market share is divided among other players [2,3]

Widespread popularity of android worldwide makes it a prime attraction for cyber criminals, increasing its susceptibility to malware and viruses. As a result, numerous studies have explored different methods to detect these malicious attacks, with machine learning (ML) emerging as a prominent technique [4] which create classifiers from a limited training set, Figure 1 depicts the taxonomical classification of the review. It

E-mail address: bilalbbashir136@gmail.com

highlights the importance of addressing not only malware detection techniques but also the identification of loopholes made by Android developers, which can expose them to unnecessary risks and malware infections. This paper encompasses methods to identify these mistakes alongside malware detection.

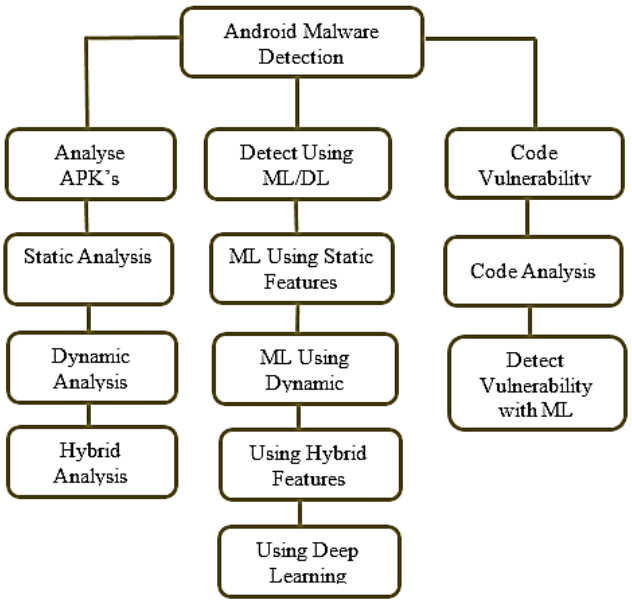

Figure 1. Android malware detection categorizes methods and techniques used.

2 BACKGROUND.

This basic knowledge to learn how the android smart phones is architectured emphasizing its security features, threat vectors that Android devices may face is introduced in this section. Additionally, it introduces the machine learning (ML) process in a manner accessible to readers without a background in ML.

A. ANDROID ARCHITECTURE

The architecture of Android operating system comprises multiple levels, starting with the Linux kernel, which provides essential hardware abstraction and security features. Above this lies the Hardware Abstraction Layer (HAL), facilitating interaction between the Android platform and various hardware components. Native libraries and the Android Runtime (ART or Dalvik) handle application execution and management. The Android framework offers APIs and tools for application development, including components like Activities and Services. At the top layer are the applications themselves, utilizing the framework to provide user functionalities [7]. Key components within the architecture include the System Server, facilitating system-level services, Binder IPC for secure inter-process communication, and the Android Debug Bridge (ADB) for developer interactions. Overall, the Android architecture offers a robust and flexible platform for building diverse mobile applications while abstracting hardware complexities and maintaining a consistent user experience. [8].

B. MALWARE ATTACKS ON ANDROID.

One of the primary threats to Android is malware attacks, which involve malicious applications containing harmful code with the intent to gain unauthorized access and engage in illicit activities that compromise the principles of security. Malware targeting smart devices can be classified based on attacker's goals like fraud and misusing resources. They spread through places like app stores, browsers, networks, and devices, finding different ways to infect and gain access to privileges [9,10,11].

Privilege acquisition methods involve technical exploits and user manipulation, like social engineering. Android malware, a threat to data and functionality on Android devices, comes in multiple ways like Spyware, adware, ransomware, and backdoors [12,13,14,15]. App collusion is another consideration when studying malware, where multiple apps work together to achieve malicious objectives [16,17].

C. APPLICATIONS OF MACHINE LEARNING.

Artificial Intelligence is a science of making intelligent systems in an artificial way, one such branch of this area is Machine Learning (ML). This technique learns from the data and makes machine learns without any natural support like human brain. It excels in scenarios where rigid algorithms are impractical, leveraging pattern recognition to automate processes. ML's adaptability and data-driven approach empower it to tackle complex tasks across various domains [9]. Machine Learning (ML) can be used various areas like voice assistants, self-driving cars. However, the field faces a challenge due to a shortage of skilled professionals. According to Statista, 82% of enterprises globally demand ML skills, but only 12% acknowledge sufficient supply. Addressing this talent gap requires understanding ML applications, empowering aspiring professionals to acquire the necessary skills and thrive in the field [28].

Machine learning can be categorized into several types, each serving different purposes and employing distinct techniques like Supervised method where the model learns from the labeled data, Unsupervised learning where the model learns from the data on its own by finding the patterns in it, semi supervised learning where the algorithm uses labelled as well as unlabeled data and then we have reinforcement learning where the model leans through the hit and trail method [18,19,20] refer Figure 3.

Figure 2 illustrate the exact working of the machine learning model going through the different phases like data processing, training and prediction. The prediction done by model should be accurate with least possible false positives. Such models are then deployed and are used in real life.

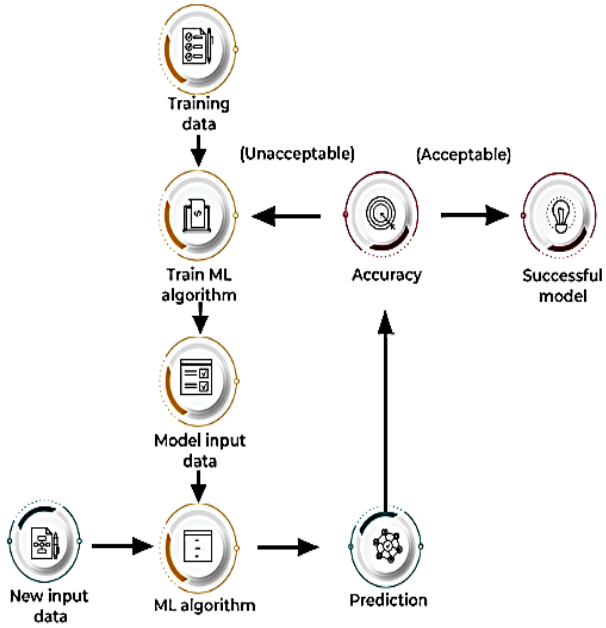

Figure 2: Working flow diagram of Machine Learning Model [78]

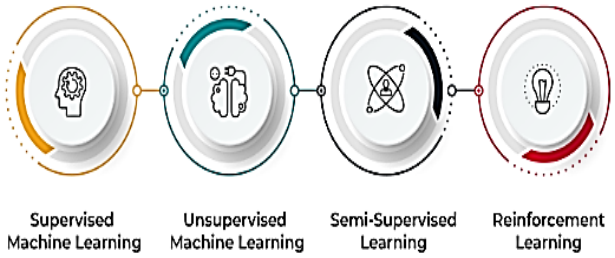

Figure 3: Machine Learning Used in different Literature [78].

3.METHODOLOGY.

Android made its debut in 2008, and as its popularity surged, so did the accompanying concerns over security in Android applications [2]. Researchers have continuously proposed innovative ML-based methodologies to tackle these concerns [9]. To find the study's objectives, a series of research questions were formulated (Section 3.1). A meticulous search approach was devised, establishing rigorous criteria for database usage and study inclusion/exclusion. The process involved strategic search methodologies, careful selection criteria definition, systematic data extraction, synthesis, and critical assessment of biases and validity threats to comprehensively address research questions.

A. RESEARCH QUESTION.

This systematic review endeavors to provide comprehensive insights into the below mentioned inquiries:

Research Question1: What are the extant literature that have explored AI based models in the detection of malware in Android platform?

Research Question2: Cutting-edge methodologies for code and APK analysis that can be harnessed in malware analysis.

Research Question3: Which ML/DL- based approaches demonstrate efficacy in detecting malware within the Androidecosystem?

Research Question4: What is the accuracy of the proposed models and their inherent strengths and limitations?

Research Question5: Which techniques are optimal for scrutinizing Android source code to unveil vulnerabilities?

By scrutinizing relevant studies, this review aims to provide profound insights and address these research questions, offering a comprehensive understanding of the existing literature. Moreover, it will shed light on the efficacy, accuracy, strengths, and limitations of ML/DL-based models.

B. DATA EXTRACTION.

To answer Research Question 1 regarding the data collection for malware detection using machine learning, data was extracted from 9 relevant studies. For Research Question 2, exploring Android code/APK analyzing techniques applicable for malware analysis, data from 22 related studies was identified and extracted. Research Question3, investigating ML/DL-based techniques suitable for malware detection, involved examination of 18 different studies with relevant data extraction. For Research Question4, which delves into detection model accuracy, strengths, and weaknesses, data was gathered from 36 research studies. Lastly, to address Research Question5, pertaining to Android source code data from 21 papers was utilized. By systematically extracting data from a total of 106 studies, comprehensive insights and answers were aimed to be provided for each research question, covering a vast area of topics related to code analysis, ML/DL techniques, model accuracy, strengths, weaknesses, and vulnerability analysis.

4. LITERATURE REVIEW.

Android Permissions provides a comprehensive analysis of permission and its implications for security. It discusses the challenges and limitations of the permission system, including the use of overly broadpermissions and the potential for user confusion [21]. Crowdroid highlight drawbacks of traditional approaches and focuses the importance of including the instrumentation framework,

system monitoring, and behavior analysis modules [22]. In this study [22] the issue of repackaging, where malicious actors modify legitimate applications to introduce malicious behaviors or bypass security checks. The authors propose a novel detection technique called "Kirin" to identify repackaged applications [23]. Focusing on the problem using machine learning algorithms as an effective approach to identify malware using artificial intelligence techniques [24].

The DREBIN [42] based analysis used a dataset of over 120,000 applications; the results showed that DREBIN achieved high accuracy [25]. The AndroDTector approach leverages ensemble learning by training multiple base classifiers on different subsets of the dataset. They compared the detection accuracy of AndroDTector [43] with several other popular machine learning algorithms and observed significant improvements [26]. By considering multiple indicators of malicious behavior machine learning-based bases techniques effectively predicts well [27, 28]. Effective malware detection approach by leveraging machine learning techniques and focusing on the analysis of permissions requested by Android applications. The paper begins by first extract the permission requests from Android applications then used various machine learning algorithms, to train classification models based on these permissions [29]. Emphasize the importance of considering both the permissions made by a app can provide a valuable insight into its behavior and potential malicious activities. In their proposed method, the authors first get the requested permissions and API's from Android applications. The study construct feature vectors based on these extracted in- formation. To classify applications and build prediction models study used labeled datasets [30]. The authors [31] conducted experiments using real world Android applications. The study results demonstrated that their method achieved high detection rates and effectively identified malicious applications. The analysis of permission patterns proves exceptionally well results in improving Android malware detection accuracy and reliability [32, 34]. Furthermore, the combined analysis of both permissions and API 's significantly enhances accuracy, compared to using either feature alone [33, 35].

5. MACHINE LEARNING APPLICATION IN ANDROID PLATFORM

In Android, we find malware mainly by checking if apps match known signature or by observing what they do (behavior-based). [39]. Signature-based detection quickly compares an app's binary code to known malware patterns in a database, making it simple, efficient, and with low false positives. But it struggles with new variants of Android malware. In contrary to this behavior based analysis, widely used, spots Android malware by analyzing how apps behave using machine learning and data science techniques, rather than just looking at specific code patterns. Within this method, researchers have extensively explored traditional machine learning [40,41]. Behavior based detection methods are widely preferred due to their capability to identify previously unknown malware by analyzing behavioral patterns exhibited by applications. Leveraging ML and DL techniques, these approaches learn and recognize malicious behaviors, enabling them to adapt and effectively detect new and evolving threats within the Android ecosystem [44,45]. These studies extensively employed various datasets to perform experiments and train models [42,43,46,47,48,49,50].

A. ANDROID MALWARE DETECTION USING STATIC FEATURES

Static analysis, widely used in Android malware detection, doesn't require installing suspicious apps or using the device's runtime environment, making it a preferred method and can identify potential indicators of malicious behavior. This includes analyzing permissions, and other static characteristics of the application [51].

Using Manifest Files: Manifest file analysis is indeed a widely used technique in malware detection. This approach involves analyzing the Manifest file to find application's permissions, and configurations. One notable model that leverages this technique is SigPID [52]. SigPID presents an Android permission-based malware detection mechanism. It utilizes the Android Manifest file to extract and analyze the declared permissions. By comparing the permissions against a known set of malicious permission patterns, SigPID can identify potential instances of malware. As part of the process, a dataset in binary format consisting of permissions was utilized. This dataset was created by combining malware and benign apps obtained from Google Play.

In [53], approach involved employing a static analyzer to extract the code-level information from APK files. The AndroZoo repository was utilized as the dataset for training the malware detection model. The proposed method holds promise for effective malware detection in Android applications by combining manifest permission analysis and code-level extraction. In the model validation process [54], the researchers explored the potential of employing reduced dimension vector generation for malware detection. The proposed work focused on utilizing machine learning (ML) models in combination with permission.

In [55], a model proposed static features of Android apps for malware classification using permissions, intents, URLs, emails, and IPs. APK files were decompiled to

access app structure. Random Forest (RF) showed high precision and recall (0.98) for permissions, while Naïve Bayes (NB) performed well for intents. RF and AdaBoost (AB) exhibited comparable precision and recall (0.97) for network-based features. refer Table 2.

Table 2: Manifest based analysis in Android Malware detection.

Study	Detection Approach	Dataset Used	Algorithms used	Accuracy	Limitations
[52]	Developing three level data purring method	Google Play Store	NB,DT,SVM	90%	Considers only permission for feature analysis
[53]	Permission Analysis using ML algorithms	AndroZoo, AppChina	RF,SVM,NB,K-means	81.5%	Lacks some other static features
[54]	Permission based using Linear regression	AMD, APKPure	Linear Regression, Knn, RF	96%	Hyper parameter tuning missing
[55]	Manifest and intents	Drebin, Google Play	RF,NB,AB	RF-98,NB-92%,AB-97%	API, opcode missing

Code Based Analysis: Code-based analysis in Android refers to the examination and evaluation of the actual code of Android applications (APK files) to identify potential security threats, vulnerabilities, or malicious behavior. This approach involves analyzing the bytecode, machine code, or source code of Android applications rather than focusing solely on their external behaviors or permissions. By examining the API's in the operand sequences of these apps, the model aimed to identify patterns and characteristics indicative of malware behavior. This analysis of API calls allowed for the differentiation between benign and malicious applications [56].

In the MaMaDroid [57] model, the API calls executed the Android apps by converting API calls into a Markov chain, the model could capture the probabilistic relationships between consecutive API calls within an app. This allowed for an effective use of the behavior of apps and facilitated the identification of potential patterns and anomalies associated with malware. [58] by abstracting and examining the API calls, the model aimed to identify unique patterns and behaviors that could distinguish between benign and malicious applications as shown in Table 2. TFDroid model was introduced as machine learning based malware detection approach that combined topics and sensitive data flow analysis. The model achieved an impressive accuracy of 93.7 percent in detecting malware. To analyze apps, the static analysis tool FlowDroid was utilized, by examining the data flow, the model aimed to identify potential sensitive data leakage and other suspicious behaviors indicative of malware. The TFDroid model's integration and sensitive data flow analysis, along with the use of SVM as the classifier, contributed to its high accuracy in detecting Android malware [59].

C. DYNAMIC ANALYSIS USING MACHINE LEARNING

Dynamic methods are used to find maicious software (malware) by letting the application run in a live environment. In a study [60], a technique for spotting malware on Android devices was introduced. This technique looks at how the app interacts with the internet, like talking to remote servers or moving data around. It watches for anything that seems suspicious or bad. By using machine learning, the system can understand how both safe and harmful apps behave when they're running. This helps it spot potential malware based on how the app acts online while it's running

By integrating machine learning and dynamic analysis in the network-based approach, the accuracy and speed of detecting Android malware are improved. This is because the system can identify new types of malware that haven't been seen before by analyzing how apps behave in real-time. This means it can respond quickly to new threats as they emerge. This approach offers a valuable tool in safeguarding Android devices against evolving malware threats in today's inter- connected digital landscape. In the network-based Android malware detection approach presented in [61], the features extraction module focused on extracting various network-related features used by the applications. These features were instrumental in analyzing the communication behavior of the apps during runtime

The extracted features included:

Domain Name System (DNS) based features: These features captured the domain names accessed by the application, providing insights into the app's communication with external servers.

HyperText Transfer Protocol (HTTP) based features: These features examined the HTTP requests and responses made by the application, revealing potential malicious activities or data transfers.

Origin destination based features: These features looked into the source and destination of network communication,

providing information on the app's interactions with different entities.

Transmission Control Protocol (TCP) based features: These features analyzed the TCP connections established by the application, helping to identifysuspicious network behavior.

Table 3: Code based analysis using machine learning.

Study	Detection Approach	Dataset Used	Algorithms used	Accuracy	Strengths	Limitations
[39]	Developing three level data purring method	Drebin	SVM,KNN,RF	87.5%	Allows abstraction of opcode sequence	Malware samples collected from few researches
[56]	Permission Analysis using ML algorithms	Drebin	NB,J45,DT	90.5%	Trained with different data set	Less malware samples used
[57]	API Calls to Markov Model Chains	AMD	NLP, SVM, KNN, NB,	86%	Efficiency	Samples obtained from leas areas
[58]	API calls and Permissions, call graphs	AMD	RF,NB,AB	92%	Analysed features individually	Less features used

By integrating various network-based features, the model becomes adept at capturing the communication patterns and actions of an application while it's running. This comprehensive approach enables the detection of Android malware based on their network activities, thereby boosting the security of Android devices against potential threats.

In another study [62], the 6th Sense model was introduced as a dynamic analysis-based method to identify Android malware, leveraging the sensors present in mobile devices. This model utilized Markov Chain, Naive Bayes (NB), algorithms to achieve malware detection by monitoring and analyzing changes in sensor data. The sensors in the mobile device, like the accelerometer, gyroscope, and GPS, provided crucial contextual information about the device's surroundings and user interactions. By employing Markov Chain, NB, and LMT, the model could effectively analyze and interpret the sensor data. The detection system learned from the dynamic sensor data patterns of both benign and malicious apps, enabling it to recognize anomalies or suspicious behavior associated with Android malware refer Table 3.

The framework in [63] is designed to effectively detect malware on devices by monitoring and analyzing various system services. Service Monitor focuses on host-based detection, meaning it operates directly on the device itself, without relying on external services or cloud-based analysis. This approach ensures that the malware detection process is efficient and independent of internet connectivity. By continuously monitoring system services, Service Monitor can identify any suspicious or abnormal behavior indicative of malware. The framework leverages various techniques and algorithmsto analyze the activities of system services in real-time, enabling it to swiftly recognize potential malware activities. The lightweight nature of Service Monitor ensures that it does not significantly impact the device's performance or consume excessive resources, making it a practical and unobtrusive solution for Android malware detection.

D.HYBRID ANALYSIS WITH MACHINE LEARNING

Hybrid analysis as the name suggest is the combination of two or more methods or techniques to bolster the effectiveness of malware detection. In hybrid analysis, the application undergoes examination both statically (without execution) and dynamically (while executing the app within a controlled environment) [64]. Static analysis concentrates on extracting features from the app's code, permissions, and other attributes to identify potential signs of malware. Dynamic analysis, monitors its interactions with the system, network, and other applications in real time. By amalgamating insights from both static and dynamic analyses, the hybrid approach can enhance the accuracy and capture a more comprehensive understanding of real-world behavior exhibited by Android apps.

Machine learning algorithms are often applied to the extracted features and observed behaviors to create robust and adaptive models for Android malware detection. The hybrid analysis approach is a valuable addition to the arsenal of security measures used to protect Android devices from evolving malware threat [65] refer table 4.

A novel method for scanning malware in android platform was introduced, employing a deep Convolutional Neural Network (CNN). The process began by disassembling Android apps to obtain their Smali code, from which the raw opcode sequence representing low-level instructions was extracted. Static analyzers were then used to derive meaningful features and patterns from this opcode sequence. These features served as input to the deep CNN for classification [70]. In

a related study [71], an experimental deep learning based using permissions achieved impressive results with 99.9% accuracy. The model were trained on the dataset to discern the nuanced differences between benign and malicious apps [72].

Table 4: Hybrid analysis using machine learning.

Study	Detection Approach	Dataset Used	Algorithms used	Accuracy	Strengths	Limitations
[64]	Extracting the DNS, HTTP, TCP, Origin based features of the network used by apps	Genome	KNN,RF,DT ,LR	98.5%	Allows abstraction of opcode sequence	Malware samples collected from few researches
[65]	Manifest analysis for permissions and system call analysis	Drebin	RF, J.48, NB, Simple Logistic, NPolyKernel	Static-96% Dynamic-88%	Compared with different ML algorithm	Uses Monkey runner
[66]	Manifest analysis for permissions, code analysis for API calls and System call analysis	MalGenome, Kaggle	SVM, LR, KNN, RF	Static-81% Dynamic-93%	Dynamic Analysis works better	Didn't work better in hybrid.

Table 2: Dynamic analysis using machine learning.

Study	Detection Approach	Dataset Used	ML algorithms used	Model Accuracy	Strengths	Limitations
[60]	Finds TCP,HTTP,DNS based features	Genome	KNN,RF,DT ,LR	98.5%	Allows abstraction of opcode sequence	Malware samples collected from few researches
[62]	Using Markov Chain-based detection technique	Google Play	NB, LMT, Markov Chain	95%	Efficient by using Sensor data	Battery consumption issues not discussed
[63]	System services on host based detection	AndroZoo, Drebin	RF,KNN,SV M	96.7%	Works directly on device, not relying on external devices.	Signature based verification was missing

D. DEEP LEARNING BASED ANDROID MALWARE DETECTION.

Deep learning based methods have shown great potential for detecting Android malware. The model presented in [68] introduces the Deep Refiner tool, which utilizes a semantic-based deep learning approach with LSTM networks to detect Android malware. This two-layer detection and validation process contribute to a powerful and reliable malware detection system for enhancing the security of Android devices. The MOCDroid model introduced a multi objective evolutionary classifier that leveraged clustering and third-party call group behaviors to detect Android malware. This innovative approach offered a powerful and effective solution for enhancing the security of Android devices against evolving malware threats. The use of multi objective optimization enabled MOCDroid to explore multiple potential solutions and find the best trade-offs among conflicting objectives, ensuring the classifier's robustness and adaptability to various types of Android malware [69].

A new and updated technique for separating malware from benign android application was proposed, leveraging a deep learning method. This method involved analyzing the features from the Smali program of Android apps using static analyzers. Initially, the Android apps were disassembled to obtain their Smali code. From this code, the raw opcode sequence, which represents the low-level instructions defining the app's behavior, was extracted. Static analyzers were then applied to this raw opcode sequence to derive meaningful features and patterns. These features were utilized as input for the deep Convolutional Neural Network (CNN) to classify the apps [70,71,72].

6. DETECTING CODE VULNERABILITIES THROUGH MACHINE LEARNING.

Hackers not only develop malware but also search for weaknesses in current applications to execute malicious activities. The discovery of vulnerabilities in Android source code is essential. Such vulnerabilities can arise from errors during the design, development, or configuration phases, making them susceptible to exploitation and compromising the security of the system. Code vulnerability detection can be carried out through two primary methods. The first approach involves reverse- engineering the APK files of the application. This entails deconstructing the compiled code to analyze its inner workings and potential security weaknesses. The second method involves identifying and addressing security flaws during the design and development stages of the application. This proactive approach aims to implement robust coding practices, security measures, and rigorous testing to prevent vulnerabilities from being introduced in the first place. By combining both these methods, developers and security experts can significantly enhance the overall security posture of Android applications, reducing the risk of exploitation and ensuring a safer user experience [72].

A. STATIC, DYNAMIC, AND HYBRID SOURCE CODE ANALYSIS.

Manifest file analysis is indeed a widely used technique in malware detection. This approach involves analyzing the Manifest file to find application's permissions, and configurations. Instead, the code is transformed into a more abstract representation [73], to identify potential properties and issues. Static analysis allows for early detection of certain vulnerabilities and can provide insights into the code's structure and logic. However, it may not capture all runtime behaviors and interactions. Dynamic analysis, on the other hand, involves running the program and observing its behavior in real-time. This method allows for the detection of runtime-specific issues, such as memory leaks or unexpected program behaviors [74]. A hybrid analysis approach combines elements of both these methods aiming to leverage the strengths of each method. By using static analysis to catch design-level vulnerabilities and dynamic analysis to observe runtime behaviors, developers and security experts can obtain a more code's security posture and potential risks [75].

The research conducted in [76] involved an online experiment with participation from Android developers. During the experiment, the developers were provided with vulnerable code samples that included issues like hard-coded credentials. After analyzing the results of the experiment, the researchers concluded that there is a need to assist developersin developing more secure applications. Automated code vulnerability detection tools can act as a safety net, helping developers catch potential security issues early in the development process. By providing real-time feedback and suggestions, these tools empower developers to write more secure code and reduce the chances of introducing vulnerabilities This study highlights the importance of integrating such automated tools into the development workflow to enhance theoverall security of Android applications. The tools are designed to detect and address various coding issues and bad practices in Android applications include Android Linters. They perform static analysis on the source code, often based on generating Abstract Syntax Trees (AST) or Universal Abstract Syntax Trees (UAST) from the written source code [77]. The process starts with the Android Linter parsing the source code and creating an AST or UAST representation. This abstract representation allows the tool to analyze the code's structure and identify potential problems, such as code smells, anti-patterns, performance bottlenecks, or security vulnerabilities. By leveraging Android Linters, developers can receive valuable feedback during the development process, enabling them to make improvements and adhere to best practices. It aids in maintaining code quality, enhancing the overall performance, and promoting secure coding practices in Android applications.

7. RESULTS AND DISCUSSION.

The distribution of techniques used in the android malware detection reviewed in this study is as follows:

Static Analysis: Approximately 75 percent of the studies utilized static analysis techniques which allows for early detection of potential malware characteristics

Dynamic Analysis: Around 10 percent of the studies reviewed employed dynamic analysis techniques for malware detection. This approach allows researchers to monitor interactions with the system, network, and other applications as the code runs, enabling the detection of suspicious or harmful behavior as it occurs. While less common than static analysis, dynamic analysis provides better results in the protected environments of applications and can complement other detection methods in identifying and mitigating malware threats.

Hybrid Analysis: Approximately 15 percent of the studies adopted the hybrid analysis technique. These numbers indicate that hybrid analysis remains at top in ML/DL based detection, but dynamic and hybrid analysis techniques are also gaining attention for their complementary benefits in identifying and combating malware threats. Researchers and security experts often

employ a combination of these techniques to achieve more robust and effective malware detection systems.

Figure 9 illustrates the distribution of techniques, with the majority of studies (75 percent) employing static analysis, followed by 10 percent using dynamic analysis and 15 percent adopting the hybrid analysis. In ML/DL feature extraction methods play a crucial role in preparing the data for analysis. Within this, the top features that are considered in most of the studies as impactful for their accuracy are show in Figure 4(for API calls), Figure 7 (for permissions) and Figure 8 (for system calls). Within hybrid analysis approaches, permissions emerge as a frequently chosen feature and for static analysis as well. The simplicity of permission analysis compared to other features contributes to its popularity. The prevalence of permissions as a commonly extracted feature could be attributed to several factors. Conversely, services and network protocols see limited usage in feature extraction, potentially due to their complexity, making analysis more challenging refer Table 5 for top features used.

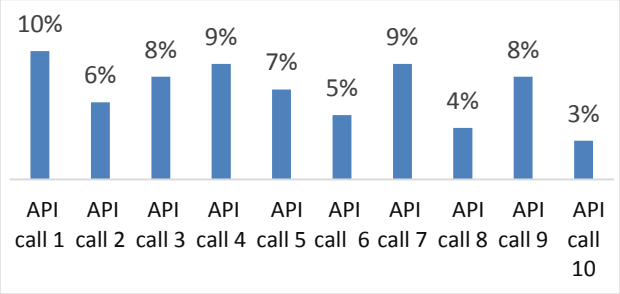

Figure. 4. API calls used in most of the research work.

In the domain of machine learning analysis for Android malware detection, Random Forest (RF), Support Vector Machine (SVM), and Naive Bayes (NB) are extensively explored models. This is largely attributed to their favorable characteristic of requiring minimal computational resources. Figure 6 depicts the algorithm employed in machine learning and deep learning oriented studies for the purpose of training algorithms aimed at Android malware detection. Conversely, models such as Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM), and AdaBoost (AB) are less commonly used, likely due to their higher computational demands. Additionally, the growing interest in deep learning-based models has contributed to this trend.

Among the reviewed studies, the Drebin (123453B,5560M dataset stood out as the most frequently employed dataset for Android malware detection, with over 3593 downloads. Additionally, the MalGenome with 2023 downloads (1260M,2539B) and AMD(10854,4354M,6500B) datasets also saw significant usage. Furthermore, the recently introduced new data set MH-100k-dataset (101975 M,) [79] is also used in most studies with over 226 downloads as of April, 2024 Figure 5. Among all these data sets Drebin is widely used one, the primary rationale behind Drebin's popularity could be attributed to its provision of a comprehensive labeled dataset. Moreover, the substantial utilization of the Google Play dataset might be attributed to the fact that it originates from Android's official app store.

Table 5: Top Most features used in static and dynamic analysis

Permission	**API calls**	**System calls**
Internet	Ljava/lang/String Builder;.:()V api call 1	Clock_get_time
Read_phone_state	Ljava/lang/StringBuilder;.append:(I) Ljava/lang/StringBuilder; API call 2	Read
Send_sms	Ljava/util/ArrayList;.:()V API call 3	Ioct
Write_external_storage	Ljava/util/Iterator;.hasNext:()Z API call 4	Epoll_pwait
Access_network_state	Ljava/util/Iterator;.next:()Ljava/lang/Object; API call 5	Rt_sigprocmask
Recieve_SMS	Landroid/content/BroadcastReceiver;.:()V API call 6	Getuid32
Recieve_boot_completed	Landroid/content/Context;.getSystemService:(Ljava/langt; API call 7	Recvfrom
Read_sms	Landroid/content/Intent;.(Landroid/content/Context;Ljava/V API call 8	Futex
Write_WIFI	Landroid/net/Uri;.parse:(Ljava/lang/String;)Landroid/net/Uri; API call 9	Gettimeofday
Access_Location	Ljava/lang/System;.currentTimeMillis:()J API call 10	write

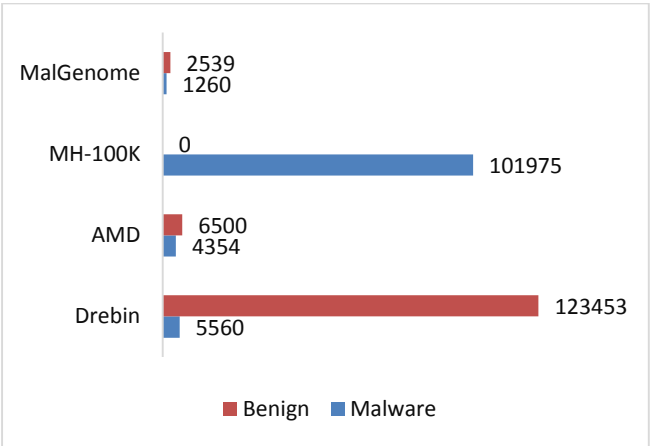

Figure 5. Datasets used in the study along with their count of samples.

Regarding vulnerability detection within Android systems, a predominant number of studies tend to favor

hybrid analysis and static analysis techniques for source code examination. To achieve a high level of accuracy in vulnerability assessment, it is imperative to thoroughly scrutinize the source code through both analysis and execution. This rationale underscores the prevalence of static, dynamic and hybrid techniques as widely favored methods for source code examination, effectively enhancing the detection of vulnerabilities.

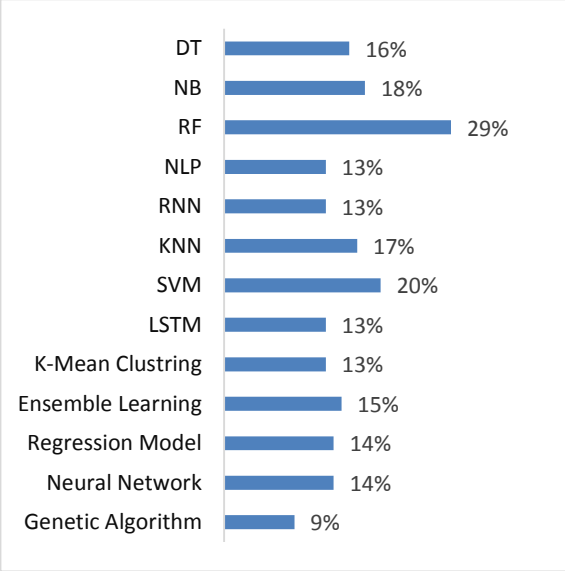

Figure 6. Machine Learning Algorithms Used in Android Malware Detection.

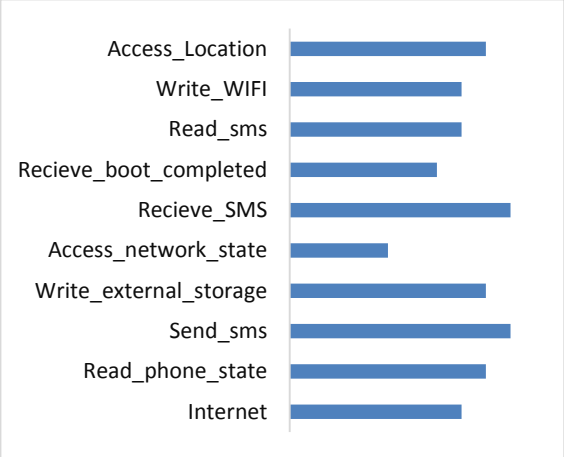

Figure 7. Top Ten Permissions Used in most Literature.

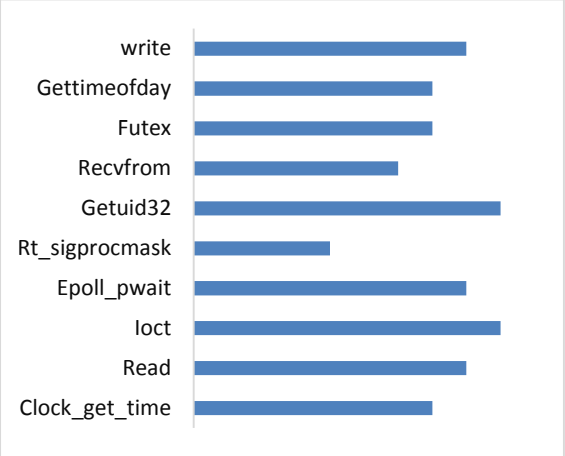

Figure 8. Top Ten System Calls Used in Most of the Literature.

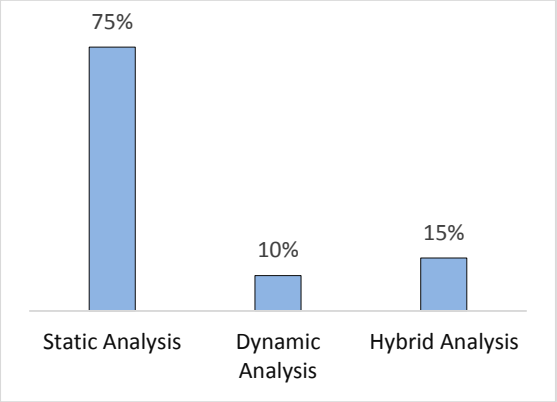

Figure 9. Android analysis used in different of Android Malware Detection.

8. CONCLUSION AND FUTURE WORK.

This research delves into the architecture and security model of Android devices, shedding light on their vulnerability to security breaches. It emphasizes Android's susceptibility to attacks and dominant position in the mobile operating system market. Drawing from recent literature, the paper conducts a comprehensive review of cutting-edge machine learning (ML) techniques for detecting Android malware, spanning from 2016 to 2023.The review covers a wide range of topics, including the utilization of ML and deep learning (DL) models, analysis methods for code and APK files, as well as techniques for feature extraction and analysis. It provides a balanced assessment of the effectiveness and limitations of these approaches. Furthermore, the paper recognizes the importance of identifying coding errors that can lead to vulnerabilities exploited by hackers. It explores ML-

based strategies for detecting such vulnerabilities within source code. Identifying gaps in current research, the study suggests future directions for enhancing Android OS security. It underscores the need for ongoing reviews in response to the evolving nature of Android malware and the strategies employed to combat it. Additionally, the paper advocates for specialized systematic reviews focusing solely on DL-based malware detection in the Android context, given the superior accuracy demonstrated by DL methods compared to traditional ML models. Moreover, it suggests exploring the potential of other learning techniques like adaptive and reinforcement learning as a promising avenue for future research and systematic reviews.

REFERENCES.

[1] Number of Mobile Phone Users Worldwide from 2016 to 2023 (In Billions Available on https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

[2] 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).

[3] 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

[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] 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] Platfrom Architecture Available online: https://developer.android.com/guide/platform (accessed on 7 July 2023).

[7] 7. Android Runrimw (ART) and Dalvik Available online: https://source.android.com/devices/tech/dalvik accessed (on 7 July 2023).

[8] 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] 9. 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 Secure. 2017, 12, 1269–1284.

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

[11] 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.

[12] 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.

[13] 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.

[14] Android Security & Privacy 2018 Year in Review. Available online: https://source.android.com/security/reports/Google_Android_Security_2018_Report_Final.pdf (accessed on 19 May 2021)

[15] Kalutarage, H.K.; Nguyen, H.N.; Shaikh, S.A. Towards a threat assessment framework for apps collusion. Telecommun. Syst. 2017, 66, 417–430.

[16] Asavoae, I.M.; Blasco, J.; Chen, T.M.; Kalutarage, H.K.; Muttik, I.; Nguyen, H.N.; Roggenbach, M.; Shaikh, S.A. Towards automated android app collusion detection. arXiv 2016, arXiv:1603.02308.

[17] Asăvoae, I.M.; Blasco, J.; Chen, T.M.; Kalutarage, H.K.; Muttik, I.; Nguyen, H.N.; Roggenbach, M.; Shaikh, S.A. Detecting malicious collusion between mobile software applications: The Android case. In Data Analytics and Decision Support for Cybersecurity; Springer: Cham, Switzerland, 2017; pp. 55–97.

[18] Van Engelen, J.E.; Hoos, H.H. A survey on semi-supervised learning. Mach. Learn. 2020, 109, 373–440.

[19] Alauthman, M.; Aslam, N.; Al-Kasassbeh, M.; Khan, S.; Al-Qerem, A.; Choo, K.K.R. An efficient reinforcement learning-based Botnet detection approach. J. Netw. Comput. Appl. 2020, 150, 102479.

[20] Shrestha, A.; Mahmood, A. Review of deep learning algorithms and architectures. IEEE Access 2019, 7, 53040–53065.

[21] Felt, A. P., Chin, E., Hanna, S., Song, D., & Wagner, D. (2011). Android permissions demystified. In Proceedings of the 18th ACM conference on Computer and communications security (CCS).

[22] Burguera, I., Zurutuza, U., &Nadjm-Tehrani, S. (2011). Crowdroid: behavior-based malware detection system for android. In Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices (SPSM)

[23] Zhou, Y., Zhou, W., Jiang, X., &Ning, P. (2012). Detecting repackaged smartphone applications in third-party android marketplaces. In Proceedings of the 19th Annual Network & Distributed System Security Symposium (NDSS).

[24] Wang, Y., Zhang, K., & Jin, H. (2013). Effective android malware detection through machine learning techniques. In Proceedings of the 2013 IEEE 11th International Conference on Dependable, Autonomic and Secure Computing (DASC).

[25] Arp, D., Spreitzen barth, M., Hübner, M., Gascon, H., & Rieck, K. (2014). DREBIN: Effective and explainable detection of android malware in your pocket. In Proceedings of the 2014 Network and Distributed System Security Symposium (NDSS).

[26] Chakraborty, T., & Das, D. (2015). AndroDTector: An efficient android malware detection technique using ensemble learning methods. Journal of Information Security and Applications, 24, 45-60.

[27] Hsiao, Hsiu-Yu, et al. "Android malware detection based on permission and API calls." International Conference on Information Networking (ICOIN). IEEE, 2015.

[28] Zhou, Yajin, et al. "Permission-based android malware detection using machine learning techniques." International Journal of Information Security. Springer, 2015.

[29] Zeki, Sahar, and Adel Ammar. "A novel permission-based approach for android malware detection." International Journal of Advanced Computer Science and Applications (IJACSA) 6.1 (2015).

[30] Deng, L., & Yang, Y. (2017). An Android malware detection method based on permissions and API calls. In Proceedings of the 2017 International Conference on Progress in Informatics and Computing (PIC).

[31] Zhang, Xinjie, et al. "Android malware detection based on permission and system call analysis." International Conference on Cloud Computing and Security (ICCCS). IEEE, 2015.

[32] Araujo, Mario, et al. "Malware detection in android applications using permission patterns." International Symposium on Engineering Secure Software and Systems. Springer, 2015.

[33] Zou, Deqing, et al. "A hybrid approach for android malware detection using permissions and api calls." Proceedings of the

13th International Conference on Security and Privacy in Communication Systems. ACM, 2016.
[34] Zhang, Yiming, et al. "Android malware detection based on permission patterns." IEEE International Conference on Web Services (ICWS). IEEE, 2016.
[35] Wang, Shuai, et al. "Detecting android malware using permission and api calls." International Conference on Information Security Practice and Experience. Springer, 2016.
[36] Chen, T.; Mao, Q.; Yang, Y.; Lv, M.; Zhu, J. Tiny Droid: A lightweight and efficient model for Android malware detection and classification. Mob. Inf. Syst. 2018, 2018.
[37] Nisa, M.; Shah, J.H.; Kanwal, S.; Raza, M.; Khan, M.A.; Damaševičius, R.; Blažauskas, T. Hybrid malware classification method using segmentation-based fractal texture analysis and deep convolution neural network features. Appl. Sci. 2020, 10, 4966.
[38] Amin, M.; Shah, B.; Sharif, A.; Ali, T.; Kim, K.l.; Anwar, S. Android malware detection through generative adversarial networks. Trans. Emerg. Telecommun. Technol. 2019, e3675.
[39] Arp, D.; Spreitzenbarth, M.; Hubner, M.; Gascon, H.; Rieck, K.; Siemens, C. Drebin: Effective and explainable detection of android malware in your pocket. In Proceedings of the 2014 Network and Distributed System Security Symposium, San Diego, CA, USA, 23–26 February 2014.
[40] Google Play. Available online: https://play.google.com/ (accessed on 13 May 2023).
[41] AndroZoo. Available online: https://androzoo.uni.lu/ (accessed on 19 May 2023).
[42] AppChina. Available online: https://tracxn.com/d/companies/appchina.com (accessed on 19 May 2023).
[43] Tencent. Available online: https://www.pcmgr-global.com/ (accessed on 20 May 2023).
[44] Zhou, Y.; Jiang, X. Dissecting android malware: Characterization and evolution. In Proceedings of the 2012 IEEE Symposium on Security and Privacy, San Francisco, CA, USA, 20–23 May 2012; pp. 95–109.
[45] VirusShare. Available online: https://virusshare.com/ (accessed on 19 May 2023).
[46] Intel Security/MacAfee. Available online: https://steppa.ca/portfolio-view/malware-threat-intel-datasets/ (accessed on 19 May 2023).
[47] Chen, K.; Wang, P.; Lee, Y.; Wang, X.; Zhang, N.; Huang, H.; Zou, W.; Liu, P. Finding unknown malice in 10 s: Mass vetting for new threats at the google-play scale. In Proceedings of the 24th USENIX Security Symposium (USENIX Security 15), Redmond, WA, USA, 7–8 May 2015; pp. 659–674.
[48] Android Malware Dataset. Available online: http://amd.arguslab.org/ (accessed on 11 May 2023).
[49] Maggi, F.; Valdi, A.; Zanero, S. Andrototal: A flexible, scalable toolbox and service for testing mobile malware detectors. In Proceedings of the Third ACM Workshop on Security and Privacy in Smartphones & Mobile Devices, Berlin, Germany, 8 November 2013; pp. 49–54.
[50] Wandoujia App Market. Available online: https://www.wandoujia.com/apps (accessed on 19 May 2021).
[51] Lubuva, H.; Huang, Q.; Msonde, G.C. A review of static malware detection for Android apps permission based on deep learning. Int. J. Computer. Network. Appl. 2019, 6, 80–91.
[52] Li, J.; Sun, L.; Yan, Q.; Li, Z.; Srisa-An, W.; Ye, H. Significant permission identification for machine-learning-based android malware detection. IEEE Trans. Ind. Inform. 2018, 14, 3216–3225.
[53] Mcdonald, J.; Herron, N.; Glisson, W.; Benton, R. Machine Learning-Based Android Malware Detection Using Manifest Permissions. In Proceedings of the 54th Hawaii International Conference on System Sciences, Maui, HI, USA, 5–8 January 2021; p. 6976.
[54] Şahin, D.Ö.; Kural, O.E.; Akleylek, S.; Kılıç, E. A novel permission-based Android malware detection system using feature selection based on linear regression. Neural Computer. Appl. 2021, 1–16.
[55] Nawaz, A. Feature Engineering based on Hybrid Features for Malware Detection over Android Framework. Turk. J. Computer. Math. Educ. (TURCOMAT) 2021, 12, 2856–2864.
[56] Zhang, P.; Cheng, S.; Lou, S.; Jiang, F. A novel Android malware detection approach using operand sequences. In Proceedings of the 2018 Third International Conference on Security of Smart Cities, Industrial Control System and Communications (SSIC), Shanghai, China, 18–19 October 2018; pp. 1–5.
[57] Onwuzurike, L.; Mariconti, E.; Androids, P.; Cristofaro, E.D.; Ross, G.; Stringhini, G. MaMaDroid: Detecting Android malware by building Markov chains of behavioral models (extended version). ACM Trans. Priv. Secure. (TOPS) 2019, 22, 1–34.
[58] Zhang, H.; Luo, S.; Zhang, Y.; Pan, L. An efficient Android malware detection system based on method-level behavioral semantic analysis. IEEE Access 2019, 7, 69246–69256.
[59] Lou, S.; Cheng, S.; Huang, J.; Jiang, F. TFDroid: Android malware detection by topics and sensitive data flows using machine learning techniques. In Proceedings of the 2019 IEEE 2nd International Conference on Information and Computer Technologies (ICICT), Kahului, HI, USA, 14–17 March 2019; pp. 30–36.
[60] Garg, S.; Peddoju, S.K.; Sarje, A.K. Network-based detection of Android malicious apps. Int. J. Inf. Secure. 2017, 16, 385–400.
[61] Salehi, M.; Amini, M.; Crispo, B. Detecting malicious applications using system services request behavior. In Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, Houston, TX, USA, 12–14 November 2019; pp. 200–209.
[62] Jannat, U.S.; Hasnayeen, S.M.; Shuhan, M.K.B.; Ferdous, M.S. Analysis and detection of malware in Android applications using machine learning. In Proceedings of the 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE), Cox's Bazar, Bangladesh, 7–9 February 2019; pp. 1–7.
[63] Salehi, M.; Amini, M.; Crispo, B. Detecting malicious applications using system services request behavior. In Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, Houston, TX, USA, 12–14 November 2019; pp. 200–209.
[64] Leeds, M.; Keffeler, M.; Atkison, T. A comparison of features for android malware detection. In Proceedings of the SouthEast Conference, Kennesaw, GA, USA, 13–15 April 2017; pp. 63–68.
[65] Jannat, U.S.; Hasnayeen, S.M.; Shuhan, M.K.B.; Ferdous, M.S. Analysis and detection of malware in Android applications using machine learning. In Proceedings of the 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE), Cox'sBazar, Bangladesh, 7–9 February 2019; pp. 1–7.
[66] Kapratwar, A.; Di Troia, F.; Stamp, M. Static and Dynamic Analysis of Android Malware; ICISSP: Porto, Portugal, 2017; pp. 653–662.
[67] Jannat, U.S.; Hasnayeen, S.M.; Shuhan, M.K.B.; Ferdous, M.S. Analysis and detection of malware in Android applications using machine learning. In Proceedings of the 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE), Cox'sBazar, Bangladesh, 7–9 February 2019; pp. 1–7.
[68] Mahindru, A.; Sangal, A. MLDroid—Framework for Android malware detection using machine learning techniques. Neural Comput. Appl. 2021, 33, 5183–5240.
[69] Xu, K.; Li, Y.; Deng, R.H.; Chen, K. Deeprefiner: Multi-layer android malware detection system applying deep neural networks. In Proceedings of the 2018 IEEE European Symposium on Security and Privacy (EuroS&P), London, UK,

24–26 April 2018; pp. 473–487.
[70] Vu, L.N.; Jung, S. AdMat: A CNN-on-Matrix Approach to Android Malware Detection and Classification. IEEE Access 2021, 9, 39680–39694. McLaughlin, N.; Martinez del Rincon, J.; Kang, B.; Yerima, S.; Miller, P.; Sezer, S.; Safaei, Y.; Trickel, E.; Zhao, Z.; Doupé, A.; et al. Deep android malware detection. In Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy, Scottsdale, AZ, USA, 22–24 March 2017; pp. 301–308.
[71] Amin, M.; Tanveer, T.A.; Tehseen, M.; Khan, M.; Khan, F.A.; Anwar, S. Static malware detection and attribution in android byte-code through an end-to-end deep system. Future Gener. Comput. Syst. 2020, 102, 112–126.
[72] Acar, Y.; Stransky, C.; Wermke, D.; Weir, C.; Mazurek, M.L.; Fahl, S. Developers need support, too: A survey of security advice for software developers. In Proceedings of the 2017 IEEE Cyber security Development (SecDev), Cambridge, MA, USA, 24–26 September 2017; pp. 22–26.
[73] Palomba, F.; Di Nucci, D.; Panichella, A.; Zaidman, A.; De Lucia, A. Lightweight detection of android-specific code smells: The adoctor project. In Proceedings of the 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER), Klagenfurt, Austria, 20–24 February 2017; pp. 487–491.
[74] Pustogarov, I.; Wu, Q.; Lie, D. Ex-vivo dynamic analysis framework for Android device drivers. In Proceedings of the 2020 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 18–21 May 2020; pp. 1088–1105.
[75] Amin, A.; Eldessouki, A.; Magdy, M.T.; Abdeen, N.; Hindy, H.; Hegazy, I. AndroShield: Automated android applications vulnerability detection, a hybrid static and dynamic analysis approach. Information 2019, 10, 326.
[76] Tahaei, M.; Vaniea, K.; Beznosov, K.; Wolters, M.K. Security Notifications in Static Analysis Tools: Developers' Attitudes, Comprehension, and Ability to Act on Them. In Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, Yokohama, Japan, 8–13 May 2021; pp. 1–17.
[77] Goaër, O.L. Enforcing green code with Android lint. In Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering Workshops, Melbourne, VIC, Australia, 21–25 September 2020; pp. 85–90.
[78] https://www.spiceworks.com/tech/artificialintelligence/articles/what-is-ml/ accessed online on 5 april,2024.
[79] Hendrio B, Vanderson R, Lucas B, Eduardo S, Diego K, Eduardo F, Android malware detection with MH-100K: An innovative dataset for advanced research. Data in Brief 51 2023

Bilal Ahmad Mantoo has received his B Tech Degree from department of computer Science and Engineering University of Kashmir Jammu and Kashmir, India in Dec.2015 and M Tech Degree from University of Punjab in 2019 and is Perusing his Ph. D degree from Presidency University, Bangalore, India. His research area is Data Science, Cyber Security related cutting edge topic