SEMARAK ILMU PUBLISHING

Journal of Advanced Research in Applied Sciences and Engineering Technology

Journal homepage:
https://semarakilmu.com.my/journals/index.php/applied_sciences_eng_tech/index
ISSN: 2462-1943

Machine Learning Approaches for Malware Classification in Android Platform: A Review

Howida Abubaker[1], Farkhana Muchtar[1,*], Salmah Fattah[2], Asraf Osman Ibrahim Elsayed[2], Carolyn Salimun[2], Hadzariah Ismail[2], Farhan Masud[3]

[1] Faculty of Computing, Universiti Teknologi Malaysia, 81310 Johor Bahru, Johor, Malaysia
[2] Faculty of Computing and Informatics, Universiti Malaysia Sabah, Jalan UMS, 88400 Kota Kinabalu, Sabah, Malaysia
[3] Department of Statistics and Computer Science, University of Veterinary and Animal Sciences, Lahore, Pakistan

ARTICLE INFO

Article history:
Received 27 October 2023
Received in revised form 4 March 2024
Accepted 7 June 2024
Available online 10 July 2024

Keywords:
Android application; Android malware; Machine learning

ABSTRACT

The rapid growth of Android applications has led to a continuous influx of Android malware. Numerous research has been undertaken to tackle that issue. Existing research has indicated that leveraging machine learning is a highly effective and promising approach for Android malware detection. This paper presents a review of Android malware detection methodologies that rely on machine learning. We commence by providing a brief overview of the background context related to Android applications, including insights into the Android system architecture, security mechanisms, and the categorization of Android malware. Subsequently, with machine learning as the central focus, we methodically examine and condense the current state of research, encompassing crucial perspectives such as sample acquisition, data pre-processing, feature selection, machine learning models, algorithms, and the assessment of detection effectiveness. The aim of this review is to equip scholars with a holistic understanding of Android malware detection through the lens of machine learning. It is intended to serve as a foundational resource for future researchers embarking on new endeavours in this field, while also providing overarching guidance for research endeavours within the broader domain.

1. Introduction

The widespread use of mobile phones [1] has recently led to a significant surge in the development of Android malware applications. Consequently, there is a growing interest among researchers in identifying distinctive patterns that differentiate regular applications from malicious ones. Machine learning techniques are leveraged to address issues where the sheer volume of data renders manual analysis impractical for humans. Machine learning classifier algorithms play a crucial role in extracting meaningful features from datasets, which are subsequently utilized for object classification, prediction, and decision-making [2]. That fast growth of malware poses increasingly formidable challenges. Current research trends are shifting towards the adoption of machine learning

* *Corresponding author.*
E-mail address: farkhana@utm.my

https://doi.org/10.37934/araset.48.1.248268

methodologies for the identification and categorization of Android malware applications. This shift is primarily attributed to their effectiveness in keeping pace with the continuous evolution of malware [3,4].

The significance of employing machine learning for malware classification cannot be overstated. Machine learning offers an essential advantage in effectively and efficiently discerning malicious software from benign applications [5]. Its importance lies in its ability to adapt and change alongside the constantly changing landscape of malware [6]. This adaptability enables the development of robust and proactive defence mechanisms that can identify emerging threats and vulnerabilities in real-time, enhancing overall cyber security efforts. Furthermore, machine learning can handle vast datasets and complex patterns that would be arduous for manual analysis, ensuring that malware detection systems remain both accurate and scalable. In essence, the incorporation of machine learning in malware classification is paramount for staying ahead in the ongoing battle against cyber threats [7].

The aim of this paper is to review the current trends of previous papers of using machine learning for malware classification in android operating system by relying on involving different features and focusing on permission features to classify malware apps from non -malicious apps.

The remainder of the paper follows this structure: Section 2 provides an overview of the Background. Section 3 describes the machine learning approaches used for malware detection. Section 4 displays the related works and section 5 concludes the study.

2. Background

2.1 Android Architecture

The Android system consists of multiple layers as shown in Figure 1 below, including the Linux kernel layer, middle layer, and application layer, which work together to pro-vide consistent services and hide the differences between the layers. These layers are essential for the functioning of the Android OS and ensure that the upper layers can access the necessary resources and functionalities. The Linux kernel layer is responsible for interacting with the hardware and managing system resources. The middle layer acts as a bridge between the kernel and the application layer, providing various services and APIs for app development. The application layer is where the user-facing apps and functionalities are implemented. These layers work in harmony to provide a seamless user experience on Android devices [8,9].

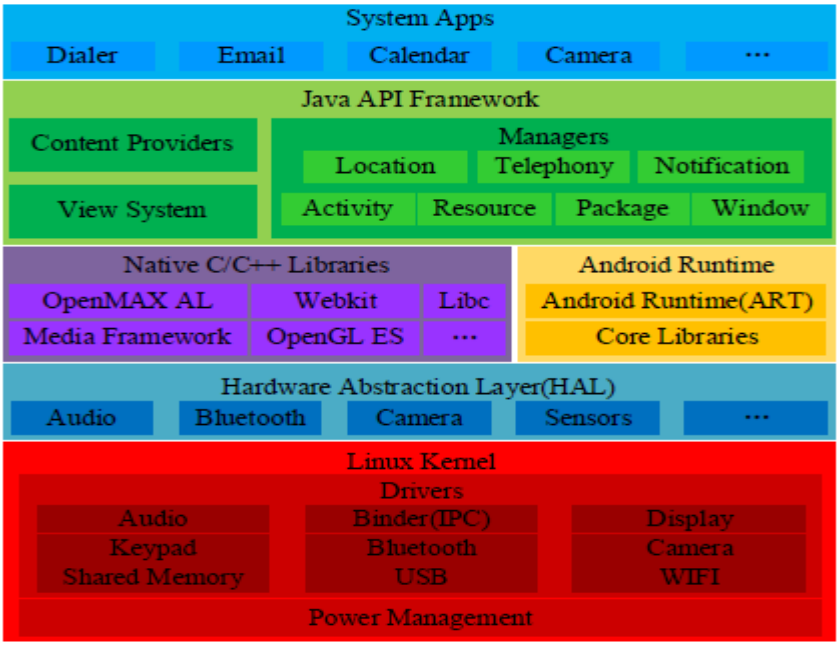

Fig. 1. Android architecture

2.2 Android Applications

Google Play is the most popular official market for downloading Android applications, but it may charge users for downloads. As a result, users often turn to third-party marketplaces that offer free apps. Each Android application has a unique user ID and a set of permissions that are requested at installation time. In previous versions of Android (5.1 and below), permissions were requested at installation time through an ask-on-install (AOI) policy. However, in Android 6.0 and above, permissions are requested at runtime through an ask-on-first-use (AOFU) policy. This updated permission mechanism aims to prevent malware and inform users about the capabilities of installed applications. The permission model used in Android has several advantages in terms of security and user awareness [10,11].

2.3 Android Security Mechanisms

Android security mechanisms are an important aspect of the operating system. The Android framework includes security control structures and a sandbox model to protect against malware and security threats [12]. A comprehensive security assessment of the Android framework has been conducted, identifying high-risk threats and pro-posing defence mechanisms to mitigate them [13]. Various security mechanisms and techniques have been reviewed to ensure the security of Android devices, including authorization and consent-related issues [14]. The analysis of Android's app installation process reveals limitations in update integrity and UID assignment, leading to recommendations for improvements in signing architecture and UID sharing mechanisms [15]. The current security mechanisms in place, such as digital signatures and coarse-grained permissions, are not sufficient to provide fine-grained control over application activities, resulting in privacy leaks [16]. To address this, a service for assessing Android Market applications and a means for mitigating security and privacy threats through automated reverse-engineering and refactoring have been pro-posed.

Furthermore, the security model of the Android operating system is primarily centred on a mandatory access control and sandboxing approach tailored towards applications, effectively limiting

access to local resources through permission constraints [17]. For instance, during the installation of an application, each app is mandated to allocate a unique User ID and a specific set of permissions. This practice acts as a protective measure by restricting access to various functionalities [18-20]. It's worth noting that earlier Android versions (5.1 and below) relied on an ask-on-install (AOI) policy, where users were required to grant permissions during installation. In contrast, the updated permissions mechanism introduced in Android 6.0 follows an ask-on-first-use (AOFU) policy, prompting user authorization at runtime when a particular feature is initially accessed [21].

2.4 Types of Android Malware

Android malware attacks are motivated by the rapid growth of mobile devices and the popularity of the Android operating system. The widespread use of Android devices has made them a prime target for malware authors, who exploit vulnerabilities in the system to gain access to user data and compromise privacy [22-24]. The open nature of the Android platform and the large number of applications available in Android markets make it easier for malware to hide among legitimate apps, posing a serious threat to Android security [25]. Malware authors are also motivated by the potential for financial gain through activities such as stealing confidential information, sending spam, and performing Distributed Denial of Service (DDoS) attacks using botnets [26].To combat these attacks, research efforts have focused on developing effective detection mechanisms and analysing the behaviour of Android mal-ware. And the malware in Android system can be classified into different types based on their behaviour and characteristics. There are many types of malwares as listed below:

i. Virus: A computer virus is a type of malware that attaches itself to another program (e.g., a document) and has the capability to replicate and spread once activated on a system. For example, opening a malicious email attachment unknowingly can lead to the virus infecting the machine. Viruses can cause data damage, gradually consume system resources, and capture keystrokes.
ii. Trojan: This form of malware typically infiltrates a user's device by disguising itself as an email attachment or a free download. Upon download, the malicious code executes its intended purpose, such as gaining unauthorized access to business systems, monitoring users' online activities, or pilfering confidential information. Unusual changes to computer settings are indicative of a Trojan's presence on a device.
iii. Worm: A computer worm, a subset of Trojan horse malware, can self-replicate and spread across systems without human activation once it infiltrates a system. Worms commonly utilize Local Area Networks (LAN) or Internet connections to propagate within a network.
iv. Spyware: Spyware is often described as malicious software designed to infiltrate a computer system, gather information about the user, and transmit it to a third party without consent. Additionally, spyware can refer to legitimate software that tracks user data for commercial purposes like advertising. However, malicious spyware is crafted with the aim of profiting from pilfered data. The surveillance activities of spyware, whether benign or fraudulent, expose users to the risk of data breaches and misuse of personal information. Moreover, the performance of networks and devices is adversely affected by malware, causing a slowdown in typical user activities.
v. Ransomware: Ransomware is a malicious software that can restrict access to computer files by encrypting them. Cyber attackers then demand a ransom in exchange for the decryption key, often compelling businesses to consider paying to swiftly regain access to

their files [103]. Some ransomware variants even include data theft to further pressure victims into complying with the ransom demands.

vi. Rootkit: Rootkit is specifically designed to provide remote access to a system without the user's knowledge. A rootkit can perform various actions on the system, such as uploading files, installing programs, altering system files, or disabling security tools like antivirus software.

vii. Bot malware, categorized as malware, empowers attackers to seize control of a user's system or execute specific tasks without the user's knowledge. In large-scale attacks, bot malware is commonly used to leverage the computing capabilities of the compromised system.

viii. Crypto-malware is another type of malware that grants threat actors the ability to engage in crypto jacking activities. While the fundamental method used by hackers and legitimate crypto miners is similar, crypto-malware exploits another user's devices and processing capacity to generate payment.

One common type is information-stealing malware, which aims to steal sensitive data from users' devices. Another type is malware that launches various malicious attacks to threaten Android users' security. There are also malware samples that undergo different transformations to evade detection by antimalware tools [27]. Additionally, Android malware can exhibit common attack features and evasion techniques, such as code execution and path constraints .These types of malwares can be detected and analysed using a combination of static and dynamic analysis techniques. Besides, that threats, there are threats are caused by exploiting requested permissions. That exploitation can be done in various ways, through unauthorized camera, SMS, call, audio, image, or location access through attacks targeting system calls, permissions, or APIs within the Android device [28-30]. Granting certain permissions, like "send_sms" or "receive_sms," can result in privacy breaches and financial implications for users. For instance, the "send_sms" permission can be misused by an app to send text messages without user consent, potentially incurring unexpected charges or facilitating unauthorized communication with third parties [31]. Moreover, specific permissions hold the potential to introduce integrity threats to the operating system, files, and physical device itself. Permissions such as "change_wifi_state," "install_packages," and "write_external_storage" exemplify this risk. For instance, the "write_external_storage" permission grants an app the capability to write to or modify the external storage of a mobile device. This permission could be exploited by malicious software to damage the device's memory by continually filling it or manipulating files. An instance of malware utilizing this permission is the "Moghava" malware, which replaces user gallery photos with advertising images, causing significant data loss [32]. By understanding the different types of Android malware and their behaviours, researchers and developers can develop effective defence techniques and tools to protect users' devices and data.

3. Machine Learning in Malware Classification

Machine learning techniques have been widely used for malware classification. Traditional methods have been summarized, and machine learning-based approaches have gained attention due to their effectiveness in solving classification problems [33]. Deep learning techniques, such as convolutional neural networks (CNN), have shown superior performance in malware classification, especially when combined with data augmentation techniques [34] .Quantum machine learning algorithms have also been explored for malware classification, with the aim of improving classification accuracy [35] support vector machines and random forests are commonly used machine

learning methods for malware classification, aiming to detect maliciousness or categorize malware by family [36]. Deep learning algorithms have been employed to enhance the performance and accuracy of malware classification, eliminating the need for manual feature engineering [37].

3.1 Machine Learning in Android Malware Detection

The widespread use of mobile phones in recent times has led to a significant increase in the development of Android malware applications. Consequently, researchers have become increasingly interested in identifying patterns that can distinguish between normal and abnormal apps. Machine learning techniques have been employed to address these challenges, especially when dealing with large datasets that are difficult for humans to handle effectively [38,39]. Machine learning methods can be classified based on their learning approach, according to their fundamental principles [66,67]. This widely-accepted classification includes supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. Supervised learning involves the use of labelled datasets to train predictive models. It's commonly used for classification or regression challenges, where the goal is to predict discrete outcomes or continuous variables, respectively. Unlike supervised learning, unsupervised learning doesn't rely on labelled data and aims to uncover the underlying structure or distribution characteristics of datasets, often used for tasks like data clustering or feature dimension reduction. Semi-supervised learning combines aspects of both supervised and unsupervised learning, using both labelled and unlabelled data. This type of machine learning is especially helpful when there's a limited amount of labelled data in a dataset, as it empowers the learner to label unlabelled data by leveraging a model of the data distribution [67].

Reinforcement learning differs from supervised learning in that it operates without labelled data. This process involves a continual loop of prediction and assessment, with input data being fed directly into the model, leading to dynamic adjustments to the model parameters. The refinement of the learning model and training data is achieved through feedback from the environment, which enables the update of model parameters. This machine-learning approach is commonly applied to dynamic systems and robot control scenarios [68,102].

In this context, machine learning classifiers are used to extract informative features from datasets, enabling the classification of objects, prediction of outcomes, and decision-making processes.

The features used for three types:

i. Static Analysis
   Static analysis involves examining the characteristics of an Android application without executing it. Machine learning models are trained on static features extracted from the app's code, manifest files, and permissions [61]. Common features include:
   - Permissions: Android apps request various permissions to access device resources. Malicious apps often request excessive or suspicious permissions.
   - API Calls: Analysis of the APIs called by an application can reveal its behaviour. Malware may exhibit unusual or malicious API call patterns.
   - Code Structure: The structure of the app's code, including the presence of obfuscation techniques, can be indicative of malware.
   - Intents: Intents define communication between Android components. Suspicious or malicious intent usage can signal malware.

Static analysis provides a high detection rate while consuming fewer resources than dynamic analysis. However, it is limited in its ability to capture the dynamic execution behaviour of malware and is significantly affected by techniques such as obfuscation and packing. As such, while it can be an effective tool in identifying and analysing potential security threats, its limitations must be acknowledged and alternative methods should be employed when necessary [61,62].

ii. Dynamic Analysis

Dynamic analysis involves running an Android application in a controlled environment and monitoring its behaviour [62]. Machine learning models are trained on features extracted during the app's execution. Key dynamic features include:

- System Calls: Recording system calls made during the app's execution can reveal malicious activities, such as file manipulation, network communication, and privilege escalation.
- API Calls: Monitoring API calls during runtime can detect suspicious behaviour, such as unauthorized access to sensitive resources.
- Network Traffic: Analysing network traffic generated by the app can identify communication with malicious servers or domains.

Dynamic analysis often requires emulator or sandbox environments to execute apps safely and record their behaviour. Dynamic analysis offers a significant advantage in terms of its robustness to obfuscation and shelling techniques. However, this technique usually requires a higher level of resource consumption and may encounter challenges in traversing all execution paths comprehensively [62].

iii. Hybrid Analysis

This method combines the static and dynamic analysis to overcome some of the limitations or weaknesses associated with each individual technique. By combining these two approaches, the hybrid analysis can provide a more comprehensive understanding of the malware's purpose, potential effects, and capabilities [63]. Additionally, hybrid analysis can help in rectifying mislabelling issues in malware detection, improving the performance of downstream applications such as malware classification [64]. Overall, the use of hybrid analysis in malware detection allows for more accurate and efficient identification of malicious software, enhancing system security.

This research primarily relies on supervised learning, which involves predicting outputs based on labelled inputs with various features. In the domain of malware classification, machine learning models are trained on datasets containing labelled examples of malware and non-malware applications for binary classification or different types or families of malware for multiclass classification. These models learn to identify distinguishing features between the classes, allowing them to classify new and previously unseen examples as malicious or non-malicious, or categorize them into specific malware families with a certain level of accuracy [40].

4. Related Works

In this section, we present previous studies that have utilized machine learning approaches for Android malware classification. Malware detection is considered a classification problem, as highlighted by [41]. Various classifier algorithms, including Support Vector Machines (SVM), K-Nearest Neighbors, Decision Trees (DT), Logistic Regression (LR), and Naive Bayes (NB), have been

commonly used in security re-search, particularly for behaviour-based or anomaly detection methods [42].

The literature on this topic can be categorized along five dimensions, as proposed by [43]. The first dimension pertains to the dataset used, the second to the types of features employed (static, dynamic, or hybrid), the third to feature selection methods, the fourth to feature weighting schemes, and the fifth to classification algorithms used to differentiate between suspicious and non-suspicious apps.

This literature review focus on classifying Android applications based on permission features through using machine learning approaches, we will emphasize studies that classify Android apps based on permissions and other features using machine learning methods.

Many studies in this domain employ binary classification, with real samples of malware and benign apps. Static features are extracted before app execution, while dynamic features are extracted afterward. Hybrid features encompass both static and dynamic attributes. For instance, [44] conducted a study where they extracted static attributes like opcodes, methods, and strings from a dataset containing 612 malignant apps and 758 benign apps. They employed three different feature selection techniques, namely information gain, correlation, and Goodman Kruskal's methods. To assess the performance of their approach, they utilized six classifier algorithms: Adaboost, Naïve Bayes, Ibk, J48, Random Forest, and SMO. They also examined various subsets of features, ranging from 100 to 1000 features, to investigate how the length of the feature set impacted accuracy. The results indicated that the Adaboost classifier outperformed the other classifiers, achieving an accuracy rate of 88.75% when using a subset of 600 features. The primary focus of their research was on feature extraction, and they did not evaluate the importance of individual features by assigning weights to them.

In the study of [45], a dataset containing 400 applications, equally divided into 200 malware and 200 non-malware apps was collected. Their approach involved using permissions and source code as static attributes to discern Android malware applications from non-malware ones. They executed four experiments in which they applied both classification and clustering algorithms using these features to differentiate malware from non-malware applications. The outcomes of their experiments demonstrated that the accuracy rate in the classification task surpassed that of the clustering task.

The researchers in [46] conducted an analysis of permissions in Android apps by creating a tool named APK Auditor. They gathered a substantial dataset of approximately 6,909 malware-infected apps from various sources, including the Drebin dataset, contagio, and the Android Malware Genome Project. In addition, they collected clean applications from the Google Play Store, totalling 1,853 clean apps. Their system functioned by evaluating apps on a server and assigning scores to the permissions requested by these apps.

Dataset comprising 1,000 malware applications and 1,000 normal applications was gathered by [47] with dynamic features. The normal apps were sourced from the Google Play Store in the year 2014 and were cross-verified using the Virus Total service. Meanwhile, the malicious apps were acquired from the Drebin dataset. The dynamic features encompassed sequences of system calls, involving the extraction of a sequence of 750 system calls from each app. They applied machine learning techniques to automatically acquire associations within these sequences, essentially creating a "fingerprint" for the malware. Subsequently, they utilized these fingerprints to detect malware and achieved a remarkable detection accuracy of 97%.

A framework was designed by [48] for the identification of Android malware applications. Their framework utilized system calls as features and was tested on a dataset containing 200 apps. In the preprocessing stage, they eliminated system calls that were not relevant to the task, while retaining

those that displayed correlations. How-ever, reducing the number of system calls proved to be challenging due to the lack of clear categories and factors associated with these calls, as highlighted by [48].

The researchers in [49] used manifest and opcode features, such as hardware, filtered intents, opcodes, permissions, application components, and strings, to detect malicious apps. They employed feature selection techniques, including entropy-based category coverage difference (ECCD) and Weighted Mutual Information (WI). Three datasets were created for their experiments, and Random Forest (RF), Rotation Forest (RF), and Support Vector Machine (SVM) classifiers were used. The results indicated that weighted mutual information outperformed ECCD. However, their features extracted from manifest files were represented as Boolean vectors, while op-codes and strings were represented as frequency of attribute vectors.

The study done by [50] collected a dataset of 107,327 benign apps and 8,701 mal-ware apps, yielding a large number of features, of which 34,630 were selected as the most relevant and related features. Support Vector Machine (SVM) served as the feature selection method to enhance classification performance. Various classifiers, including SVM, CART, K-NN, and NB, were employed to distinguish malicious apps from benign ones.

Another work done by [51] collected 1,227 malicious applications and 1,189 benign applications, using 196 system calls as dynamic features. They employed a Back-propagation Neural Network to train their dataset. The results indicated the highest F-score rate of 0.982, a true positive rate (TPR) of 0.977, and a false positive rate (FPR) of 0.013. Additionally, their model achieved an accuracy rate of 0.7. Notably, they used only dynamic features for app classification.

A tool called SWORD (Semantic Aware Dynamic Malware Detection) was introduced by [52] to classify mobile applications as either benign or malicious based on their usage of system calls. They assembled a dataset comprising 2,000 applications, evenly split between benign and malicious (1,000 of each). SWORD monitored app behaviour during runtime to collect system calls and constructed a Sequential System-Call Graph (SSG) using Markov chains. They derived typical program behaviour paths using the Asymptotic Equipartition Property (AEP) on these graphs, forming the basis for their classification model. ALBF metric was applied to each path, and supervised learning was employed for model training. Their results demonstrated that the pro-posed model achieved an accuracy rate of 94.2%. However, SWORD's high over-head, caused by injecting a substantial number of systems calls into malicious paths, posed limitations and impacted learning performance.

A forensic tool named FAMOUS (Forensic Analysis of Mobile devices using Scoring of application permissions) was proposed by [53] for scanning and providing descriptive reports on installed applications on attached devices. They collected a dataset comprising 5,553 malware apps and 5,818 non-malware apps, focusing on static permissions. Random Forest (RF), Naïve Bayes (NB), Decision Tree (DT), and Sup-port Vector Machine (SVM) classifiers were utilized, and a scoring engine was employed to assign weight values to permissions based on their frequency in malware and non-malware apps.

The study conducted by [54] developed EnDroid to differentiate between malware and non-malware Android apps. They collected two datasets, M1 (8,806 benign apps and 5,213 suspicious apps) and M2 (5,000 benign apps and 5,000 suspicious apps). They focused on ten dynamic features, such as cryptographic operations, network operations, file operations, dexclass load, information leaks, sent SMS, phone calls, service starts, system calls, and receiver actions. Chi-square feature selection was used to remove redundant features, and a combination of classifiers, including Decision Tree, Extremely Randomized Trees, Random Forest, Linear SVM, and Boosted Trees, with Logistic Regression as a meta-classifier, was employed. However, their study solely utilized dynamic features and did not investigate permissions requested by apps.

A framework was designed by [55] to distinguish between malicious and clean apps by examining permission, metadata, and sensitive API call features. They collected a dataset containing 8,177 applications from the Google Play Store and AndroZoo. Information gain (IG) served as the feature selection method, resulting in 20 sensitive API calls and 62 permission features. Decision Tree-J48 (DT), Random Forest (RF), Naïve Bayes (NB), and Support Vector Machine (SVM) classifiers were employed for assessment, and the features were represented in binary vectors.

The authors in the study of [56] introduced the Multilevel DroidFusion model for Android malware detection. They used four datasets, including Malgenome-215, Drebin-215, McAfee-350, and McAfee-100, which contained varying numbers of clean and malware instances. Information gain (IG) was employed for feature ranking, and DroidFusion was evaluated using different datasets with one or more base classifiers. However, the focus of their study was primarily on static features.

The experiment don by [57] used a dataset containing 9,419 suspicious apps and 6,070 non-suspicious apps to identify malware. Static permissions were the sole features used in their study, and they employed various machine learning classifiers, including Naïve Bayes (NB), Bayesian Network (BN), J48, Random Trees (RT), Random Forest (RF), and k-Nearest Neighbors (K-NN).

However, many studies employed hybrid features to improve classification tasks. For instance, [58] proposed by a hybrid approach was that incorporated static features (permissions and intents) and dynamic features (data leakages, cryptographic API calls, and network manipulation) to identify malware apps from non-malware apps. Feature selection methods, Information Gain (IG) and Principal Component Analysis (PCA), were used to select distinctive features. Classifiers included RF, NB, GB, and DT.

The work conducted by [59] collected 8,000 applications (4,000 malware and 4,000 benign) and proposed a hybrid feature-based approach using static features (permissions and API calls) and dynamic features (system calls). Their hybrid approach out-performed static and dynamic features alone, but the representation of features was not specified. Another study employed hybrid features made by [64] introduced a model called Tree Augmented Naive Bayes (TAN) for Android malware detection, incorporating hybrid features comprising permissions, API calls, and system calls.

In summary, many existing studies have explored various feature sets and machine learning algorithms for Android malware classification, including static features (permissions, intents, and API calls), dynamic features (system calls), and hybrid features combining both static and dynamic aspects. These studies have used a range of datasets and feature selection techniques, with varying degrees of accuracy in distinguishing between benign and malicious apps. However, not all studies have investigated dynamic permissions or provided comprehensive details about their methodologies. Table 1 below summarized some of the previous works in this study related to android applications classification based on using machine learning methods.

Table 1
Summary of machine learnings technique used in selected references related to Android malware detection

Study	Class label	Features used	Type of features	Feature Selection methods	Feature weighting	Machine Learning classifier	Advantages	Disadvantages
[42]	310926 non-malware apps and 4868 malwares	Permissions	Static features	CorrCoef ,Mutual Informati on, and T-test	N/A	SVM, DT, and RF	Reduced number of features by using different types of feature selection methods.	Features are not assigned weights and the most top important patterns of permissions are not identified.

[105]	200 clean and 200 malware apps	Permissions and source code	Static features	N/A	N/A	Not mentioned	Classifying apps by exploring static permissions.	Using all features to build classifier models will increase complexity overhead.
[106]	850 benign apps and 620 malware apps	Permissions and intent-filters	Static features	Information gain(IG)	N/A	ID3 and J48 classifiers	Using a smaller number of features in classifying apps.	IG selects features based on the relations to class not to classifier.
[107]	200 clean and 200 malicious apps	Permissions and source code	Static features	N/A	N/A	C4.5 decision trees, RF, NB, and SVM with SMO JRip	Utilizing permission features in differentiating between malware and non-malware.	Feeding classifiers with all features will lead to lower learning process.
2017 [99]	11,000 Android apps	Permissions	Hybrid permissions features	N/A	N/A	NB, J48, RF, Simple Logistic and k-star	Exploring permissions at installation & run time.	Using all features in learning task will increase complexity overhead.
[101]	5553 malware and 5818 benign apps	Permissions	Static features	N/A	Frequency method	RF, DT, NB, and SVM	Classifying android apps based on permission features.	Assigning weights to features using the frequency method is more expensive because there are many permission features are requested by apps.
[55]	8177 Android apps (benign) and AndroZoo (malware)	Permission, sensitive API calls, and metadata	Static features	Information gain (IG)	N/A	NB, SVM, DT-J48, and RF.	Decreasing computational overhead by selecting a small number of significant features instead of using all features.	Representing the extracted features in binary values makes all features are equal in importance which is not.

[77]	4000 malware and 4000 benign samples	Permissions and system calls	Hybrid Features	TF-IDF	Not mentioned	NB, RF, XGBoost, GC Forest	Utilizing hybrid features in classifying apps is more efficient.	Selecting important features using TF-IDF will increase complexity overhead due to many requested permissions.
[100]	5774 malware apps and 500 normal apps	(permissions) and intents and (API calls, cryptographic data leakages, & network manipulation)	Hybrid Features	Information gain (IG) and (PCA).	Not mentioned	NB, RF, GB and DT	Feeding machine learning classifiers with a smaller number of features using (IG) and PCA.	IG selects features based on their relations to the class label not to the classifier which is less effective than the method that selects features based on the classifier.
[107]	6070 benign apps and 9419 malware apps	Permission Features	Static Features	Not mentioned	TF-IDF	(NB), (BN), J48, (RT), (RF) and K- (K-NN)	Extracting the most significant permission features using TF-IDF.	Using TF-IDF in extracting the significant permissions will increase computational overhead
[108]	1650 malware apps and 1650 good ware apps	API calls, permissions and system calls	Hybrid Features	Not declared	Not declared	A Tree Augmented Naive Bayes (TAN)	Using hybrid features makes the classification process more effective.	Feeding classifiers with all features helps in lowering learning process & increasing complexity cost overhead.

Android malware detection datasets have been reviewed in several papers. Table 2 shows some datasets used in the previous studies.

Table 2
Summary of some Datasets used in the previous study

Study	Dataset(s) Used
[69]	Derbin
[70]	Malgenome and Contagio project
[71]	Contagiodump
[72]	AMD Projects
[73]	Drebin, Androzoo

4.1 Feature Selection

Numerous studies have utilized feature selection to decrease the number of features used in a model. This involves selecting features that are strongly correlated with class labels, which can help reduce the computational cost of the model [74]. However, when addressing real-world problems, it's not always possible to identify all the relevant features at the outset. As a result, a large number of features are often collected to gain a comprehensive understanding of the domain. Unfortunately, many of these features may be irrelevant to the target class [74]. Using a learning model with a dataset containing redundant and unimportant attributes can lead to issues, as highlighted by Dash and Liu in 1997.

Different types of features selection approaches have been used by previous studies; the Table 3 below summarize some of these studies.

Table 3
Summary of feature selection methods in selected references related to Android malware detection based on machine learning approaches

Reference	Feature Selection approach used
[75]	Information Gain (IG)
[76]	Chi-Square
[77]	Wrapper method
[78]	TF-IDF,cosine similarity
[79]	Genetic search (GS)

Feature selection is an essential process in data analysis, as it helps to identify a subset of features that are relevant to a specific target variable. One popular approach to feature selection is the use of Information Gain (IG) technique. This method has been widely adopted in various studies to classify malware apps from non-malware apps as done by [75]. The IG technique evaluates the relevance of a feature by measuring the reduction in entropy that occurs when the feature is added to the subset of features related to the target variable. Its effectiveness lies in its ability to select a set of features that maximizes the mutual information between the features and the target variable, while minimizing redundancy among the selected features.

Chi-Square is a widely used feature selection method utilized by researchers in the classification of Android malware applications as used in the study of [76]. It calculates the chi-square statistics between each feature variable and the class object, analysing the correlation between the two. If the class object is found to be independent of the feature variable, the said feature is not deemed significant.

The wrapper method is a technique used for search problems in feature selection. It presents different groups of collections and designs analytical models to evaluate a group of features. The accuracy of the model determines the score assigned to the collection group. In a study conducted

by [77] has relied on the wrapper approach as a feature selection method to obtain significant features that classify malware apps from non-malware.

The study done by [78] proposed a low-cost and high-efficient method for detecting Android malware using static analysis and the Rotation Forest (RF) model and TF-IDF to select the important features. It achieves a high accuracy of 88.26% with 88.40% sensitivity at the precision of 88.16%. The proposed method improves the accuracy by 3.33% compared to the Support Vector Machine (SVM) model.

4.2 Type of Features

In this section, the chosen features from diverse machine learning algorithms within the domain of Android malware detection are outlined and assessed. These features can be categorized into three groups: static features, dynamic features, and hybrid features, contingent on whether they are obtained through the execution of an Android application [79]. The techniques employed for analysing these three feature types are termed static analysis, dynamic analysis, and hybrid analysis, correspondingly as explained previously.

4.2.1 Static features

Static features are characteristics that are obtained by analysing the source code or other related information associated with an application. This analytical approach is known as static analysis and is denoted by [80]. In the case of Android applications, the APK file is the primary object of scrutiny as it serves as the installation package for Android applications. Decompiling APK files reveals files such as AndroidManifest.xml and small files, among others. Analysing these files uncovers a range of static features, including permissions, API calls, Dalvik opcodes, and other components. Table 4 summarizes some static features used in machine learning-based Android malware detection.

Table 4
Summary of some static features used in selected references

Study	features
[81]	API call
[82]	Permission, API call, System event, URL
[83]	Opcode sequence
[84]	Description of function, Data flow, Permission

4.2.2 Dynamic features

In the realm of Android applications, the behaviours exhibited during real-world usage or simulated testing - known as dynamic features [85] - are analysed through a process called dynamic analysis. In the case of Android apps, this includes system calls, API calls, network traffic, and CPU data. Many studies have utilized dynamic analysis techniques for Android malware detection, including [86,87] which employ dynamic taint analysis, and [88], which integrates Dalvik opcode with graph theory. Some dynamic features used in machine learning-based Android malware detection are displayed in Table 5 below.

Table 5
Summary of some dynamic features used in selected references

Study	Features
[89]	Frequency of system calls
[90]	CPU, Memory, Network traffic
[91]	Network traffic: DNS, HTTP, TCP, Origin- destination.
[92]	Method call, Inter-component communication (ICC) intent

4.2.3 Hybrid features

Hybrid features involves the fusion of static and dynamic components, forming a comprehensive analytical approach [93]. While static analysis prioritizes individual aspects, dynamic analysis covers a wider scope. By combining the benefits of both static and dynamic analyses, hybrid analysis can detect specific threats in specific scenarios. Some hybrid features used in machine learning-based Android malware detection are displayed in Table 6 below.

Table 6
Summary of some dynamic features used in selected references

Study	features
[94]	Static: Permission Dynamic: Behaviour (System function, Sensitive permission, Sensitive API)
[95]	Static: Permission, Intent, Hardware feature, Software features, IP address, Advertisement module, System security setting. Dynamic: Behaviour (Sensitive API, System service, IP address)
[96]	Static: Permission, API, Intent, Components, Hardware Dynamic: Behaviour (System call)
[97]	Static: Permission, API, Intent, Min_sdk. Dynamic: Behaviour (Service startup, File operation, SMS and phone event, Sensitive data leakage, Network data transmission, etc.)

As observed from the previous studies that majority of research relies on utilizing static permissions, which are obtained at the time of installation without executing the applications. Some studies incorporate additional features alongside permissions, such as APIs, as demonstrated by [98]. In contrast, the investigation conducted by [99] employs hybrid permissions, combining static and dynamic permission features, for identifying Android malware through machine learning classifiers. Static permissions are extracted at installation time, while dynamic permissions are extracted during runtime after the execution of the apps. They extracted 123 dynamic permissions, categorized them into safe and unsafe based on the Android developer website, and assessed their dataset using five machine learning classifiers (Naive Bayes, Decision Tree, Random Forest, Simple Logistic, and k-star) through cross-validation and a dataset split of 66%.

The paper provides a succinct overview of the broader context of Android applications while concentrating on crucial elements of machine learning, including type of features and feature selection. Table 7 shows the differences among those studies.

Table 7
The differences among those studies

Study	Type of features	Features selection
[42]	Static	✓
[48]	Dynamic	✓
[49]	Static	✓
[100]	Hybrid	✓
[101]	Hybrid	✓

5. Conclusions

In this review, we investigate into the literature on various types of malwares and their potential risks on the Android operating system. We also examine the machine learning techniques utilized in detecting malware within the Android system. Our findings reveal that many studies utilize a combination of static, dynamic, and hybrid features to differentiate between benign and malicious apps. Additionally, permission features are commonly used as static features. Moving forward, we recommend further exploration into machine learning methods that analyse combined permissions requested at installation and run time for more accurate malware classification.

Acknowledgement
This research was not funded by any grant.

References
[1] Amilah, Anis. "The Usability of Mobile Experiment Application in Science Subject for Secondary Student." *life* 16, no. 1 (2019): 34-41.
[2] Tahtaci, Burak, and Beyzanur Canbay. "Android malware detection using machine learning." In *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)*, pp. 1-6. IEEE, 2020. https://doi.org/10.1109/ASYU50717.2020.9259834
[3] Gibert Llauradó, Daniel, Carles Mateu Piñol, and Jordi Planes Cid. "The rise of machine learning for detection and classification of malware: Research developments, trends and challenge." *Journal of Network and Computer Applications, 2020, vol. 153, 102526* (2020). https://doi.org/10.1016/j.jnca.2019.102526
[4] Ucci, Daniele, Leonardo Aniello, and Roberto Baldoni. "Survey of machine learning techniques for malware analysis." *Computers & Security* 81 (2019): 123-147. https://doi.org/10.1016/j.cose.2018.11.001
[5] Chen, Yizheng, Zhoujie Ding, and David Wagner. "Continuous Learning for Android Malware Detection." *arXiv preprint arXiv:2302.04332* (2023).
[6] Singh, Mahendra Pratap, and Heena Kausar Khan. "Malware Detection in Android Applications Using Machine Learning." In *2023 International Conference on Advances in Electronics, Communication, Computing and Intelligent Information Systems (ICAECIS)*, pp. 105-110. IEEE, 2023. https://doi.org/10.1109/ICAECIS58353.2023.10170311
[7] Hammood, Layth, İbrahim Alper Doğru, and Kazım Kılıç. "Machine Learning-Based Adaptive Genetic Algorithm for Android Malware Detection in Auto-Driving Vehicles." *Applied Sciences* 13, no. 9 (2023): 5403. https://doi.org/10.3390/app13095403
[6] Atacak, İsmail, Kazım Kılıç, and İbrahim Alper Doğru. "Android malware detection using hybrid ANFIS architecture with low computational cost convolutional layers." *PeerJ Computer Science* 8 (2022): e1092. https://doi.org/10.7717/peerj-cs.1092
[8] Xie, Nannan, Zhaowei Qin, and Xiaoqiang Di. "GA-StackingMD: Android Malware Detection Method Based on Genetic Algorithm Optimized Stacking." *Applied Sciences* 13, no. 4 (2023): 2629. https://doi.org/10.3390/app13042629
[9] Hu, M. "An Architecture Design Method Based on Android System." (2019).
[10] Majethiya, Raj J., and Monika Shah. "Comparative analysis of detecting over-claim permissions from android apps." In *2023 International Conference on Intelligent Systems, Advanced Computing and Communication (ISACC)*, pp. 1-8. IEEE, 2023. https://doi.org/10.1109/ISACC56298.2023.10084321

[11] Tao, Guanhong, Zibin Zheng, Ziying Guo, and Michael R. Lyu. "MalPat: Mining patterns of malicious and benign Android apps via permission-related APIs." *IEEE Transactions on Reliability* 67, no. 1 (2017): 355-369. https://doi.org/10.1109/TR.2017.2778147
[12] Mayrhofer, René, Jeffrey Vander Stoep, Chad Brubaker, and Nick Kralevich. "The android platform security model." *ACM Transactions on Privacy and Security (TOPS)* 24, no. 3 (2021): 1-35. https://doi.org/10.1145/3448609
[13] Yu, Lifang, Tianchang Yang, and Shaozhang Niu. "Secure Access to Data Transmission for Inter-Component Communication." (2016).
[14] Faruki, Parvez, Ammar Bharmal, Vijay Laxmi, Vijay Ganmoor, Manoj Singh Gaur, Mauro Conti, and Muttukrishnan Rajarajan. "Android security: a survey of issues, malware penetration, and defenses." *IEEE communications surveys & tutorials* 17, no. 2 (2014): 998-1022. https://doi.org/10.1109/COMST.2014.2386139
[15] Sun, Lin, ShuTao Huang, YunWu Wang, and MeiMei Huo. "Application policy security mechanisms of Android system." In *2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems*, pp. 1722-1725. IEEE, 2012. https://doi.org/10.1109/HPCC.2012.258
[16] Yi-yang, Fu, and Dan-ping Zhou. "Android's Security Mechanism Analysis." In *The 26th National Symposium on computer security*, pp. 23-25. 2011.
[17] Arora, A., Das, A. K., Rawat, D., & Chavan, V. "Security Model of the Android Operating System: A Mandatory Access Control and Sandboxing Approach." (2019).
[18] Bhandari, S., Sharma, N., & Verma, A. "Allocation of Unique User IDs and Permission Sets During Android Application Installation for Resource Access Control." (2017).
[19] Kumar, P., Singh, R., Gupta, S., *et al.,* "Android Security: Restricting Access to Local Resources Through Permission Constraints During App Installation." (2018).
[20] Wang, L., Chen, H., Zhang, Q., *et al.,* "Protective Measures in the Android Operating System: Allocation of Unique User IDs and Specific Permissions During Application Installation." (2019).
[21] Gao, Z., Li, M., Wang, S., *et al.,* "Evolution of Android Permissions Mechanism: From Ask-on-Install (AOI) to Ask-on-First-Use (AOFU) Policy for User Authorization." (2020).
[22] Almomani, Iman, Mohanned Ahmed, and Walid El-Shafai. "Android malware analysis in a nutshell." *PloS one* 17, no. 7 (2022): e0270647. https://doi.org/10.1371/journal.pone.0270647
[23] Sadananda, L., Bolwar, A., & Musthafa, A. "Review on Analysis of Different Malware Types in Android System." *Journal of Emerging Technologies and Innovative Research.* (2019).
[24] Sabbah, Ahmed, Adel Taweel, and Samer Zein. "Android Malware Detection: A Literature Review." In *Inernational Conference on Ubiquitous Security*, pp. 263-278. Singapore: Springer Nature Singapore, 2022. https://doi.org/10.1007/978-981-99-0272-9_18
[25] Khemani, Shreya, Darshil Jain, and Gaurav Prasad. "Android malware detection techniques." In *Emerging Research in Computing, Information, Communication and Applications: ERCICA 2018, Volume 2*, pp. 449-457. Springer Singapore, 2019. https://doi.org/10.1007/978-981-13-6001-5_36
[26] Jiang, Xuxian, and Yajin Zhou. *Android malware*. Springer, 2013. https://doi.org/10.1007/978-1-4614-7394-7
[27] Chavan, Neeraj, Fabio Di Troia, and Mark Stamp. "A comparative analysis of android malware." *arXiv preprint arXiv:1904.00735* (2019). https://doi.org/10.5220/0007701506640673
[28] Saudi, M., Suliman, A., Alzahrani, F., *et al.,* "Android Malware and Attack Techniques: A Comprehensive Review." (2017).
[29] Alenezi, M., Almomani, A. "Analysis of Android Malware Attack Features and Evasion Techniques." (2018).
[30] Ghasempour, Z., Selamat, A., Ibrahim, S., *et al.,* "Detecting Android Malware Using a Combination of Static and Dynamic Analysis Techniques." (2020).
[31] Garg, Shivi, and Niyati Baliyan. "Comparative analysis of Android and iOS from security viewpoint." *Computer Science Review* 40 (2021): 100372. https://doi.org/10.1016/j.cosrev.2021.100372
[32] Deypir, M. "Privacy and Financial Implications of Unauthorized SMS Access Through Android Permissions." (2019).
[33] Dini, G., Monteleone, S., Falzarano, S., *et al.,* "Integrity Threats in Android Due to Specific Permissions: A Case Study." (2018).
[34] Wurdi, I., Amin, M. B. "A Survey of Malware Classification Techniques. Journal of Computer Virology and Hacking Techniques." (2017).
[35] Yerima, S. Y., Sezer, S., McWilliams, G. "Towards the Development of Quantum Machine Learning for Malware Classification." *Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC)* (pp. 977-984). (2016).
[36] Kolosnjaji, Bojan, Apostolis Zarras, George Webster, and Claudia Eckert. "Deep learning for classification of malware system call sequences." In *AI 2016: Advances in Artificial Intelligence: 29th Australasian Joint Conference, Hobart, TAS, Australia, December 5-8, 2016, Proceedings 29*, pp. 137-149. Springer International Publishing, 2016. https://doi.org/10.1007/978-3-319-50127-7_11

[37] Rajab, M. A., Debbabi, M. "Deep Learning for Malware Classification Using End-to-End Autoencoders." *2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE)* pp. 1198-1205, (2018).
[38] Le, D. X., Phung, D., Venkatesh, S. "Context-Aware Adaptive Android Malware Classification." *2018 IEEE Conference on Computational Intelligence in Cyber Security (CICS)* pp. 1-8, (2018).
[39] Liu, Kaijun, Shengwei Xu, Guoai Xu, Miao Zhang, Dawei Sun, and Haifeng Liu. "A review of android malware detection approaches based on machine learning." *IEEE Access* 8 (2020): 124579-124607. https://doi.org/10.1109/ACCESS.2020.3006143
[40] Pendlebury, R. A., Lim, K. L., Tiong, W. W. "A Survey of Machine Learning Algorithms for Malware Classification." *Information Systems,* 75, 41-60, (2018).
[41] Singh, Mahendra Pratap, and Heena Kausar Khan. "Malware Detection in Android Applications Using Machine Learning." In *2023 International Conference on Advances in Electronics, Communication, Computing and Intelligent Information Systems (ICAECIS)*, pp. 105-110. IEEE, 2023. https://doi.org/10.1109/ICAECIS58353.2023.10170311
[42] Wang, W., Zhang, Y., Cao, Z., Guan, H. "Permission-Combination-Based Android Malware Detection." *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* pp. 3191-3196, (2014).
[43] Alazab, M. "A Systematic Review on Android Malware Detection Based on Machine Learning Techniques." *Journal of Cyber Security Technology,* 4(1), 47-62, (2020).
[44] Deepa, K. S., Eswaran, K. "Detection of Android Malware Using Data Mining Techniques." *Procedia Computer Science,* 47, 43-50, (2015).
[45] Damshenas, M., Dousti, S. "A Novel Approach for Detection of Android Malware Using Minimal Dynamic Features." *2015 12th International ISC (Iranian Society of Cryptology) Conference on Information Security and Cryptology (ISCISC)* pp. 199-204, (2015).
[46] Talha, M., Borisyuk, F. "Malware Detection on Android: An Investigation into Machine Learning Algorithms." *2015 7th Computer Science and Electronic Engineering Conference (CEEC)* pp. 140-145, (2015).
[47] Canfora, G., Medvet, E., Mercaldo, F., Visaggio, C. A. "Android Malware Detection Through Structural and Behavioral Analysis of Application Components." *IEEE Transactions on Software Engineering,* 41(11), 1127-1147, (2015).
[48] Amamra, A., Challal, Y. "Detection of Android Malware Using Models of Supervised Learning." *2016 12th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)* pp. 35-42, (2016).
[49] Varsha, V., & Mahalakshmi, M. "Android Malware Classification Using Machine Learning Techniques." *Procedia Computer Science,* 89, 97-101, (2016).
[50] Wang, S., Li, X., Zhang, J., & Zhang, J. "Android Malware Detection Based on Ensemble Learning Methods." *Future Generation Computer Systems,* 89, 358-369, (2018).
[51] Xiao, Z., Yuan, J., & Liao, Q. "A Two-Stage Neural Network Model for Android Malware Detection." *2017 13th International Conference on Computational Intelligence and Security (CIS)* pp. 37-41, (2017).
[52] Sugunan, Krishna, T. Gireesh Kumar, and K. A. Dhanya. "Static and dynamic analysis for android malware detection." In *Advances in Big Data and Cloud Computing*, pp. 147-155. Springer Singapore, 2018. https://doi.org/10.1007/978-981-10-7200-0_13
[53] Kumar, N., Sood, K., Kumar, V., & Sharma, A. "An Empirical Analysis of Android Malware Classification Using Permissions." *2018 IEEE International Conference on Consumer Electronics (ICCE)* pp. 1-6, (2018).
[54] Feng, L., Duan, Y., Liu, Z., & Zhang, J. "Android Malware Detection Based on Feature Selection Using Hybrid Features." *2018 14th International Conference on Computational Intelligence and Security (CIS)* pp. 26-30, (2018).
[55] Aminordin, A. Z. M. I., FAIZAL MA, and R. O. B. I. A. H. Yusof. "Android malware classification base on application category using static code analysis." *J. Theor. Appl. Inf. Technol* 96, no. 11 (2018).
[56] Yerima, Suleiman Y., and Sakir Sezer. "Droidfusion: A novel multilevel classifier fusion approach for android malware detection." *IEEE transactions on cybernetics* 49, no. 2 (2018): 453-466. https://doi.org/10.1109/TCYB.2017.2777960
[57] Yuan, S., Lu, L., & Xue, Y. "A Comparative Study of Machine Learning Algorithms for Android Malware Detection Using Static Features." *2020 IEEE International Conference on Big Data (Big Data)* pp. 4461-4468, (2020).
[58] Arshad, M., & Raza, A. "Detection of Android Malware Using Static and Dynamic Analysis." *2018 International Conference on Frontiers of Information Technology (FIT)* pp. 281-286, (2018).
[59] Fang, B., & Xiao, Y. "A Hybrid Approach to Android Malware Detection Using Static and Dynamic Analysis." *2019 IEEE International Conference on Data Mining Workshops (ICDMW)* pp. 738-745, (2019).
[60] Chandy, Jeff. "Review on Malware, Types, and its Analysis."
[61] Dhalaria, Meghna, and Ekta Gandotra. "A hybrid approach for android malware detection and family classification." (2020). https://doi.org/10.9781/ijimai.2020.09.001

[62] Ding, Chao, Nurbol Luktarhan, Bei Lu, and Wenhui Zhang. "A hybrid analysis-based approach to android malware family classification." *Entropy* 23, no. 8 (2021): 1009. https://doi.org/10.3390/e23081009
[63] Yang, Wang, Mingzhe Gao, Ligeng Chen, Zhengxuan Liu, and Lingyun Ying. "RecMaL: Rectify the malware family label via hybrid analysis." *Computers & Security* 128 (2023): 103177. https://doi.org/10.1016/j.cose.2023.103177
[64] Dugyala, Raman, N. Hanuman Reddy, V. Uma Maheswari, Gouse Baig Mohammad, Fayadh Alenezi, and Kemal Polat. "Analysis of malware detection and signature generation using a novel hybrid approach." *Mathematical Problems in Engineering* 2022 (2022): 1-13. https://doi.org/10.1155/2022/5852412
[65] Qingyang, Ling. "Machine learning algorithms review." *Applied and Computational Engineering,* 4(1):91-98, (2023). https://doi.org/10.54254/2755-2721/4/20230355
[66] Oppong, Stephen Opoku. "Predicting Students' Performance Using Machine Learning Algorithms: A Review." *Asian Journal of Research in Computer Science* 16, no. 3 (2023): 128-148. https://doi.org/10.9734/ajrcos/2023/v16i3351
[67] Henry, Knipe. "Overview of Machine Learning." *International Journal of Advanced Research in Science, Communication and Technology,* 76-79, (2022). https://doi.org/10.48175/IJARSCT-3885
[68] Louis, H., Kauffman. "Reinforcement Learning." 350-370, (2023). https://doi.org/10.1017/9781108755610.013
[69] Morales-Ortega, Salvador, Ponciano Jorge Escamilla-Ambrosio, Abraham Rodriguez-Mota, and Lilian D. Coronado-De-Alba. "Native malware detection in smartphones with android os using static analysis, feature selection and ensemble classifiers." In *2016 11th International Conference on Malicious and Unwanted Software (MALWARE)*, pp. 1-8. IEEE, 2016. https://doi.org/10.1109/MALWARE.2016.7888731
[70] Shahriar, Hossain, Mahbubul Islam, and Victor Clincy. "Android malware detection using permission analysis." In *SoutheastCon 2017*, pp. 1-6. IEEE, 2017. https://doi.org/10.1109/SECON.2017.7925347
[71] Aswini, A. M., and P. Vinod. "Droid permission miner: Mining prominent permissions for Android malware analysis." In *The Fifth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2014)*, pp. 81-86. IEEE, 2014. https://doi.org/10.1109/ICADIWT.2014.6814679
[72] Ilham, Soussi, Ghadi Abderrahim, and Boudhir Anouar Abdelhakim. "Permission based malware detection in android devices." In *Proceedings of the 3rd International Conference on Smart City Applications*, pp. 1-6. 2018. https://doi.org/10.1145/3286606.3286860
[73] Mohamad Arif, Juliza, Mohd Faizal Ab Razak, Suryanti Awang, Sharfah Ratibah Tuan Mat, Nor Syahidatul Nadiah Ismail, and Ahmad Firdaus. "A static analysis approach for Android permission-based malware detection systems." *PloS one* 16, no. 9 (2021): e0257968. https://doi.org/10.1371/journal.pone.0257968
[74] Wang, Huanran, Weizhe Zhang, and Hui He. "You are what the permissions told me! Android malware detection based on hybrid tactics." *Journal of Information Security and Applications* 66 (2022): 103159. https://doi.org/10.1016/j.jisa.2022.103159
[75] Alazab, Moutaz, Mamoun Alazab, Andrii Shalaginov, Abdelwadood Mesleh, and Albara Awajan. "Intelligent mobile malware detection using permission requests and API calls." *Future Generation Computer Systems* 107 (2020): 509-521. https://doi.org/10.1016/j.future.2020.02.002
[76] Effrosynidis, Dimitrios, and Avi Arampatzis. "An evaluation of feature selection methods for environmental data." *Ecological Informatics* 61 (2021): 101224. https://doi.org/10.1016/j.ecoinf.2021.101224
[77] Fang, Yong, Yangchen Gao, F. A. N. Jing, and L. E. I. Zhang. "Android malware familial classification based on dex file section features." *IEEE Access* 8 (2020): 10614-10627. https://doi.org/10.1109/ACCESS.2020.2965646
[78] Zhu, Hui-Juan, Zhu-Hong You, Ze-Xuan Zhu, Wei-Lei Shi, Xing Chen, and Li Cheng. "DroidDet: effective and robust detection of android malware using static analysis along with rotation forest model." *Neurocomputing* 272 (2018): 638-646. https://doi.org/10.1016/j.neucom.2017.07.030
[79] Qamar, Attia, Ahmad Karim, and Victor Chang. "Mobile malware attacks: Review, taxonomy & future directions." *Future Generation Computer Systems* 97 (2019): 887-909. https://doi.org/10.1016/j.future.2019.03.007
[80] Amro, Bela. "Malware detection techniques for mobile devices." *arXiv preprint arXiv:1801.02837* (2018). https://doi.org/10.2139/ssrn.3430317
[81] Shen, Feng, Justin Del Vecchio, Aziz Mohaisen, Steven Y. Ko, and Lukasz Ziarek. "Android malware detection using complex-flows." *IEEE Transactions on Mobile Computing* 18, no. 6 (2018): 1231-1245. https://doi.org/10.1109/TMC.2018.2861405
[82] Zhu, Hui-Juan, Zhu-Hong You, Ze-Xuan Zhu, Wei-Lei Shi, Xing Chen, and Li Cheng. "DroidDet: effective and robust detection of android malware using static analysis along with rotation forest model." *Neurocomputing* 272 (2018): 638-646. https://doi.org/10.1016/j.neucom.2017.07.030
[83] Chen, Tieming, Qingyu Mao, Yimin Yang, Mingqi Lv, and Jianming Zhu. "Tinydroid: a lightweight and efficient model for android malware detection and classification." *Mobile information systems* 2018 (2018). https://doi.org/10.1155/2018/4157156

[84] Lou, Songhao, Shaoyin Cheng, Jingjing Huang, and Fan Jiang. "TFDroid: Android malware detection by topics and sensitive data flows using machine learning techniques." In *2019 IEEE 2Nd international conference on information and computer technologies (ICICT)*, pp. 30-36. IEEE, 2019. https://doi.org/10.1109/INFOCT.2019.8711179

[85] Jannat, Umme Sumaya, Syed Md Hasnayeen, Mirza Kamrul Bashar Shuhan, and Md Sadek Ferdous. "Analysis and detection of malware in Android applications using machine learning." In *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, pp. 1-7. IEEE, 2019. https://doi.org/10.1109/ECACE.2019.8679493

[86] Enck, William, Peter Gilbert, Seungyeop Han, Vasant Tendulkar, Byung-Gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N. Sheth. "Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones." *ACM Transactions on Computer Systems (TOCS)* 32, no. 2 (2014): 1-29. https://doi.org/10.1145/2619091

[87] Shankar, Venkatesh Gauri, Gaurav Somani, Manoj Singh Gaur, Vijay Laxmi, and Mauro Conti. "AndroTaint: An efficient android malware detection framework using dynamic taint analysis." *2017 ISEA Asia security and privacy (ISEASP)* (2017): 1-13. https://doi.org/10.1109/ISEASP.2017.7976989

[88] Zhang, Jixin, Zheng Qin, Kehuan Zhang, Hui Yin, and Jingfu Zou. "Dalvik opcode graph based android malware variants detection using global topology features." *IEEE Access* 6 (2018): 51964-51974. https://doi.org/10.1109/ACCESS.2018.2870534

[89] Bhatia, Taniya, and Rishabh Kaushal. "Malware detection in android based on dynamic analysis." In *2017 International conference on cyber security and protection of digital services (Cyber security)*, pp. 1-6. IEEE, 2017. https://doi.org/10.1109/CyberSecPODS.2017.8074847

[90] Bhatia, Taniya, and Rishabh Kaushal. "Malware detection in android based on dynamic analysis." In *2017 International conference on cyber security and protection of digital services (Cyber security)*, pp. 1-6. IEEE, 2017. https://doi.org/10.1109/CyberSecPODS.2017.8074847

[91] Garg, Shree, Sateesh K. Peddoju, and Anil K. Sarje. "Network-based detection of Android malicious apps." *International Journal of Information Security* 16 (2017): 385-400. https://doi.org/10.1007/s10207-016-0343-z

[92] Cai, Haipeng, Na Meng, Barbara Ryder, and Daphne Yao. "Droidcat: Effective android malware detection and categorization via app-level profiling." *IEEE Transactions on Information Forensics and Security* 14, no. 6 (2018): 1455-1470. https://doi.org/10.1109/TIFS.2018.2879302

[93] BalaGanesh, D., Amlan Chakrabarti, and Divya Midhunchakkaravarthy. "Smart devices threats, vulnerabilities and malware detection approaches: a survey." *European Journal of Engineering and Technology Research* 3, no. 2 (2018): 7-12. https://doi.org/10.24018/ejeng.2018.3.2.302

[94] Wei, Linfeng, Weiqi Luo, Jian Weng, Yanjun Zhong, Xiaoqian Zhang, and Zheng Yan. "Machine learning-based malicious application detection of android." *IEEE Access* 5 (2017): 25591-25601. https://doi.org/10.1109/ACCESS.2017.2771470

[95] Saracino, Andrea, Daniele Sgandurra, Gianluca Dini, and Fabio Martinelli. "Madam: Effective and efficient behavior-based android malware detection and prevention." *IEEE Transactions on Dependable and Secure Computing* 15, no. 1 (2016): 83-97. https://doi.org/10.1109/TDSC.2016.2536605

[96] Arshad, Saba, Munam A. Shah, Abdul Wahid, Amjad Mehmood, Houbing Song, and Hongnian Yu. "SAMADroid: a novel 3-level hybrid malware detection model for android operating system." *IEEE Access* 6 (2018): 4321-4339. https://doi.org/10.1109/ACCESS.2018.2792941

[97] Jannat, Umme Sumaya, Syed Md Hasnayeen, Mirza Kamrul Bashar Shuhan, and Md Sadek Ferdous. "Analysis and detection of malware in Android applications using machine learning." In *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, pp. 1-7. IEEE, 2019. https://doi.org/10.1109/ECACE.2019.8679493

[98] Wang, Zhen, Kai Li, Yan Hu, Akira Fukuda, and Weiqiang Kong. "Multilevel permission extraction in android applications for malware detection." In *2019 international conference on computer, information and telecommunication systems (CITS)*, pp. 1-5. IEEE, 2019. https://doi.org/10.1109/CITS.2019.8862060

[99] Mahindru, Arvind, and Paramvir Singh. "Dynamic permissions based android malware detection using machine learning techniques." In *Proceedings of the 10th innovations in software engineering conference*, pp. 202-210. 2017. https://doi.org/10.1145/3021460.3021485

[100] Hussain, Syed Jawad, Usman Ahmed, Humera Liaquat, Shiba Mir, N. Z. Jhanjhi, and Mamoona Humayun. "IMIAD: intelligent malware identification for android platform." In *2019 International Conference on Computer and Information Sciences (ICCIS)*, pp. 1-6. IEEE, 2019. https://doi.org/10.1109/ICCISci.2019.8716471

[101] Kumar, Ajit, K. S. Kuppusamy, and Gnanasekaran Aghila. "FAMOUS: Forensic Analysis of MObile devices Using Scoring of application permissions." *Future Generation Computer Systems* 83 (2018): 158-172. https://doi.org/10.1016/j.future.2018.02.001

[102] Mustafa, Wan Azani, Haniza Yazid, Aimi Salihah Abdul-Nasir, and Mastura Jaafar. "An illumination normalization on face images: A comparative." *computer* 35, no. 1 (2017): 1-9.
[103] Chin, Thoo Ai, and Liu Min. "The Effect of Supply Chain Risk Management Practices on Resilience and Performance: A Systematic Literature Review." *Journal of Advanced Research in Technology and Innovation Management* 1, no. 1 (2021): 41-53.
[104] Damshenas, Mohsen, Ali Dehghantanha, Kim-Kwang Raymond Choo, and Ramlan Mahmud. "M0droid: An android behavioral-based malware detection model." *Journal of Information Privacy and Security* 11, no. 3 (2015): 141-157. https://doi.org/10.1080/15536548.2015.1073510
[105] Verma, Sushma, and S. K. Muttoo. "An Android Malware Detection Framework-based on Permissions and Intents." *Defence Science Journal* 66, no. 6 (2016). https://doi.org/10.14429/dsj.66.10803
[106] Milosevic, Nikola, Ali Dehghantanha, and Kim-Kwang Raymond Choo. "Machine learning aided Android malware classification." *Computers & Electrical Engineering* 61 (2017): 266-274. https://doi.org/10.1016/j.compeleceng.2017.02.013
[107] Yuan, Hongli, Yongchuan Tang, Wenjuan Sun, and Li Liu. "A detection method for android application security based on TF-IDF and machine learning." *Plos one* 15, no. 9 (2020): e0238694. https://doi.org/10.1371/journal.pone.0238694
[108] Surendran, Roopak, Tony Thomas, and Sabu Emmanuel. "A TAN based hybrid model for android malware detection." *Journal of Information Security and Applications* 54 (2020): 102483. https://doi.org/10.1016/j.jisa.2020.102483