Available online at www.qu.edu.iq/journalcm
JOURNAL OF AL-QADISIYAH FOR COMPUTER SCIENCE AND MATHEMATICS
ISSN:2521-3504(online) ISSN:2074-0204(print)

Enhancing Attack Detection in Android Environment Through Software Based on Machine Learning Methods

Shatha hamead Othman [a]*, Huda Abdulaali abdulbaqi [b]*

[a] *Al Mustansiriyah University, College of Science, Computer Department, Baghdad, Iraq, Email: shathaalqaisy20@uomustansiriyah.edu.iq*

[b] *Al Mustansiriyah University, College of Science, Computer Department, Baghdad, Iraq. Email: huda.it@uomustansiriyah.edu.iq*

ARTICLE INFO

Article history:
Received: 29 /10/2024
Rrevised form: 22 /11/2024
Accepted : 12 /2/2025
Available online: 30 /03/2025

Keywords:

Android Security,Mobile Malware,Machine Learning,Attack Detection Systems

ABSTRACT

With the growing use of mobile devices, it is projected that nearly 70% of mobile phone users own an Android smartphone. Due to the completely open-source nature of Android, the Android operating system is vulnerable to various attacks. The smartphone's data has a sensitive nature, making it important to protect from these attacks. Machine learning (ML) approaches have proven to be an efficient methods of detecting these assaults since they can create a classifier from a set of training instances; therefore, it does not need an existing database of harmful signatures. This review paper aims to cover different characteristics involved in Android attack detection systems, such as the Android operating system environment, feature extraction, feature selection, performance measures, supervised, and unsupervised models. They are described based on previous works that employ machine learning for detecting Android malware. Furthermore, this paper intends to help researchers gain expertise in the subject of Android attack detection.

MSC..

https://doi.org/10.29304/jqcsm.2025.17.11969

1. Introduction

Android became known as the most widely used smartphone operating system when it was released in 2008. Around 86% of smartphones in 2019 sold internationally were depend on Android [1]. In the mobile sector, Android OS held a global market share of over 71.9% as of July 2023 [2]. Due to several factors, such as Android applications coarse-grained permission management and the open ecological mode, given the capability to invoke third-party code, several security attack surfaces are presents, which substantially compromise the security of the Android applications. According to the statistics, it was discovered that over 3 million Android apps contained malware. In 2016, implying that a new malware program for Android was detected nearly every 10 seconds [3]. Android-powered smartphones have been the target of more attacks in recent years, partly because of their growing use for banking and business-related operations. Sensitive financial and personal data is now frequently processed by apps as part of social media, communication, and mobile banking .Additionally, there are more possible security

∗Corresponding author

Email addresses:

Communicated by 'sub etitor'

and privacy problems caused by malware[4]. Several solutions have been proposed to evaluate Android security, comprising malware detection, developer reviews, vulnerability detection, and application reinforcement [5]. Different detection strategies for Android malware have been presented to handle these security issues, one of the most efficient methods is to use machine learning for detection [6]. There are a couple primary sources of features in the machine learning-based Android malware detection implementation: both dynamic and static extraction [7].

The static features are taken from Dalvik bytecode, native code, manifest, sound, picture, and inverted APK files. The dynamic features extracted from code execution, pathways, variable value tracking, sensitive function calls, log records and more behaviors that occur during the application's execution by running APK files in the environment [5].

Many research works [8][9][10][11] focused on training machine learning classification algorithms to detect unknown Android malware applications based on known Android malware apps. Machine learning techniques have produced a significant accuracy ratio in detecting malicious programs, contingent upon the quality of extracted features, the dataset, and the training methodologies employed for the models [12]. Machine learning can be divided into three major categories: supervised machine learning [13,14,15], unsupervised machine learning , and reinforcement machine learning [16]. Additionally, each learning type is linked to three fundamental learning methods: clustering, regression, and classifications. Regression is related to reinforcement learning, where the expected result is being graded, ranked or estimated such as linear regression algorithms. The practice of classifying data sets that have been labeled into classes or groups is called classification which is applied in supervised learning such as decision tree algorithms. The method employed in unsupervised learning is called clustering when the datasets are unlabeled. A label is the name of the group or class which the data instances belong to. Many characteristics are used to represent data in machine learning that can be continuous, nominal, or categorical [17].

This review paper is structured as follows: Section 2 provides important background information, discusses the architecture of Android operating system, and Android attacks. In Section 3, this paper covers different key components in Android attack detection system. It reviews feature extraction and code analysis approaches, feature selection, machine learning (ML) categories and performance metrics to evaluate any classification algorithm. This is followed by a discussion and review of past studies in Section 4. Section 5 concludes the paper.

2. Background

Android, globally being one of the most widely used mobile operating systems, so it is a main target for cybercriminals. Android attack refers to cyberattacks targeting devices running on the Android operating system. This section covers the architecture and potential Android attack.

2.1- Architecture of Android

Android is constructed on the top of the Linux Kernel as shown in Fig. 1 . Linux is selected since it is free and open source validates the path evidence, offers networking protocols and drivers, controls virtual memory, security, and device power [18]. Android uses a layered architecture [19]. The layers are organized from bottom to top, the Hardware Abstraction Layer, Native Libraries, Android Runtime, Java API Framework, and System Apps are all stacked on top of the Linux Kernal Layer. Each layer is accountable for a specific job. Android-based applications and various services used in the system make use of the Android Runtime (ART). Dalvik was the runtime environment that preceded the Android Runtime (ART). ART and Dalvik were created for Android application-related projects. The ART uses the Dalvik Executable (DEX) format and the bytecode specifications [20]. The remaining components are memory management and power management. As Android-based applications operate on battery- powered devices with limited memory. Consequently, Android OS is prepared such that any resource may be effectively managed [18].

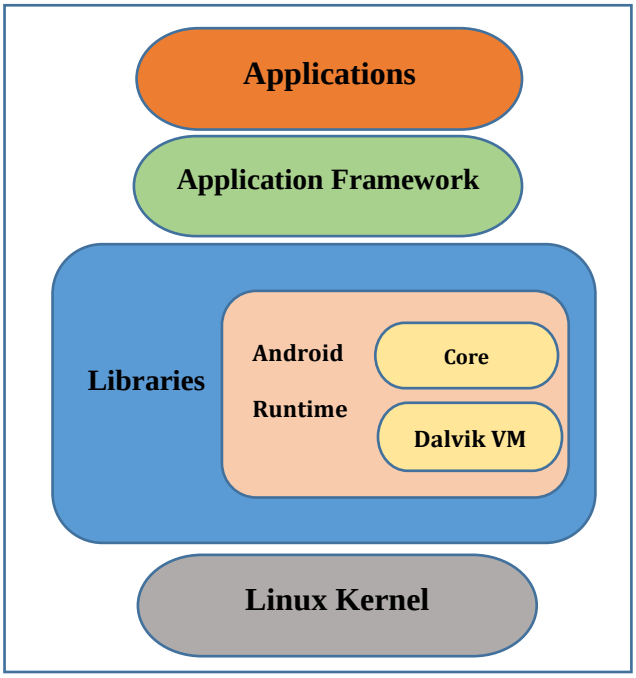

Fig. 1 - android OS architecture

2.2 Android-specific attacks

Malicious programs might target the devices in several ways. In addition to generally recognized attacks, Android specific attacks comprise [21]:

- **Privacy leakage**. It occurs when the device user allows risky permissions that provide access to personal information and sensitive data to a malware application with no user's knowledge.
- **Colluding attacks**. This can happen when lots of applications that the same developer has signed are set up and share UID. These applications, coupled with the shared UIDs, share harmful permissions wanted by a specific app, allowing another application to access the same data without requesting for permission.
- **Premium Text (SMS)**. Is a well-known attack in which an individual unknowingly signs up for a premium SMS service, which can result in financial loss because these SMS are more expensive than standard text messages [22].

Harmful applications try to take advantage of vulnerabilities and offer a variety of threats to Android system. In the fight against attackers, ML can be quite useful. This paper in the next section, reviews key attack detection system concepts.

3. Attack detection system

Android attack detection system involves different key components to identify and detect potential security threats. Here are the basic concepts of how to approach building such a system:

3.1 feature extraction

Based on the feature extraction method, Android attack detection can be classified as static, dynamic, or hybrid analysis. It uses static, dynamic, or both features. as shown in Fig. 2 .

3.1.1 Static analysis

Static analysis involves evaluating Android files and obtaining information such as opcode sequences, requested permissions, API calls, and so on. Static detection is frequently utilized in detecting Android malware, among various additional features due to it is simple to extract [23].

Ayman Elshenawy et al [24] suggested a successful machine learning model for detecting Android malware according to static malware analysis of Android application features before installation on devices. The suggested approach is applied to two independent datasets, DREBIN and MALGENOME. A machine learning classifier was created by them using multi-level feature selection techniques like PCA and IG. The classifier that was implemented provides around an 89% decrease from the starting set of features for the defined dataset. Furthermore, they improved the performance of the suggested classifier and the RF classifier's experimental results showed the best performance.

Ahmed S. Shatnawi et al [25] present a static base classification method for identifying malware depend on Android permission and API calls depending on three Machine Learning algorithms, K-nearest neighbors (KNN), Support Vector Machines (SVM), and Naive Bayes (NB) with the new Android malware dataset (CIC InvesAndMal2019). They show that when compared to other classifiers, the SVM classifier performed best, achieving an accuracy rate of 83% when using API call features and an average of 94% when using permission features. This classifier aimed for high malware detection while supporting research on mobile information security.

Munam Ali Shah et al [26] use Machine Learning methods and Reverse Engineered Android applications' features to detect vulnerabilities present in smartphone applications. Firstly, they propose a model that merges the biggest datasets available of malware samples with more innovative static feature sets than ordinary methods. Secondly, they have employed machine learning with ensemble learning. Their proposed model has a 96% accuracy rate for false positives with a 0.3 false positive rate in the specified environment. The study also found that strong and ensemble learning algorithms perform better when doing classifications with high-dimensional data.

Vasileios Syrris and Dimitris Geneiatakis [27] studied the powerful supervised machine learning techniques that make use of Drebin data set static analysis data. They evaluate six classification techniques under different configurations. Their experiments show that SVM can achieve accuracy of nearly 99% and the model physical size supplied by SVM is 390 MB size. Hence, their study shows that about 1000 features from the Drebin feature set are adequate to achieve great classification performance using the classification models. Moreover, they provide and defend several methods for lowering the dimensionality of space.

3.1.2 Dynamic analysis

Dynamic analysis is a technique that runs the program in a sandbox environment and follows the behavior of the application's API call sequence, system calls, networks traffic, and CPU data to observe the data flow while the program is running, thus showing the true state of the program processing more near to the actual status. However, it is not extensively utilized in a large number of resources and the slow detection while running the program [21].

Bilal Majeed Abdulridha Al Latteef and Faris Mutar Mahdi Aledam [28] introduce an effective machine learning based strategy for malware detection on smartphones. This model selects and infers features using both static and dynamic analytic techniques. Next, it extracts features making dimensionality reduction using sampling and Principal Component Analysis (PCA) without compromising accuracy, using real-world and synthetic datasets. They evaluate different metrics and criteria for analysis and machine learning schemes to assess the effectiveness of malware detection methods. Moreover, their model shows a cost-effective solution for malware detection in smartphone operating systems, notably apps that are malicious or recompiled.

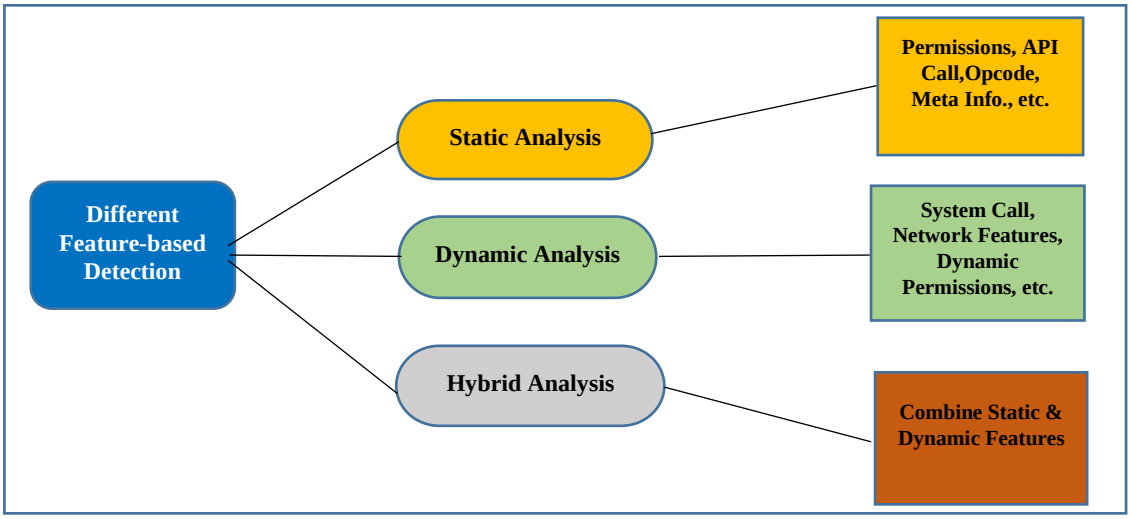

Fig. 2 - static, dynamic and hybrid analysis

3.1.3 Hybrid analysis

Is a blend of dynamic and static analysis, which can improve the accuracy and efficiency of Android malware detection [21].

R. Srinivasan et al [29] provided a detailed examination of Android malware detection using machine learning techniques with static, dynamic, and hybrid analysis. They identify Android malware by using machine learning and genetic algorithms and the order of techniques applied in the suggested work is: Data Set preparation, using Androguard tool and Classification based on machine learning algorithms. Finally, experiments indicate that Support Vector Machine and Neural Network classifiers may maintain over 90–91% classification accuracy when dealing with lower dimension feature sets while reducing classifier training complexity.

Mahmoud Al-Ayyoub et al [30] studied dynamic, static and hybrid analysis to identify and detect harmful activities. They use classifiers based on machine learning to identify malware applications also they show the performance of these classifiers in the process of classification. They applied the effectiveness of the four machine learning classifiers that were designed to detect malware based on the static (permissions) and dynamic features (action repetition) and they investigated in three stages. Also, the results of their studies explained that accuracy obtained above 94% was from dynamic, static and hybrid analysis. Therefore, employing only static analysis is likely to be more effective and less expensive.

Table 1. Summarizes previous works that performed static, dynamic and hybrid analysis in Android detection systems.

studies	analysis method	Extracted features
[24] [25] [26] [27] [8] [9] [37] [11]	Static analysis	Permissions, intents, API calls, etc.
[28] [36] [32]	Dynamic analysis	Dynamic permissions, actions,IP address, etc.
[29 [30][31] [33] [10]	Hybrid analysis	Combining static,dynamic features

3.2 Feature Selection

Feature selection is the process that increases malware detection effectiveness by removing redundant and useless features in Android malware detection. Fig. 3 illustrates the feature selection process [21]. By selecting the good features, you can improve the accuracy, efficiency and interpretability of your predictive models. The most effective feature selection methods include Recursive Feature Elimination (RFE), LASSO regression, Principal Component Analysis (PCA), information gain (IG), support vector machine (SVM) and singular value decomposition (SVD).

To demonstrate the importance of feature selection, Sakib Shahriar Shafin et al [31] provide a two-layer machine learning detection model that relies on ensemble learning with stacked generalization techniques. The Random Forest Regression technique is used in this work to select features. They tested the suggested model using 11,598 samples with various malicious attack types from the CIC-Maldroid-2020 dataset. Consequently, a wide range of metrics including precision, accuracy, recall, and F1-score, were employed to evaluate the effectiveness of their suggested model. Finally, they introduce an efficient and robust classification system and Android malware detection that are capable of accurately detecting 99% of malicious attacks including banking malware, riskware and adware which have become increasingly common in recent times.

R. Jeberson Retna Raj and V. Joseph Raymond [32] implemented in their suggested work a modern PCA (Principal Component Analysis) based on conditional dependency features together with feature reduction technique. They developed a hybrid technique that combines static and dynamic processes in order to achieve better outcomes. Among the most significant factors in malware detection is the Android Sensitive Permission. To verify that zero-day exploits may be identified with a higher accuracy rate, performance measure datasets gathered from several repositories including Github, Virus Share, and the Canadian Institute of Cyber Security, will be compared with models. When comparing the model to other classifiers, they can observe that it has obtained an accuracy of 80% with PCA using a KNN method using k = 15, which is better than the 76% accuracy without PCA.

K S Ranadheer Kumar and Jagadish Gurrala [33] employed machine learning classifiers to assess the performance of filter-based feature selection strategies, in analyzing Android permission for malware detection. Their results demonstrate that filter-based feature selection utilizing Fisher's Score and Information Gain outperformed the Chi-Square Test in the terms of feature reduction, realizing classification accuracies of 91.53% and 91.22% on the high-dimensional CICInvesAndMal2019 dataset. Notably, the Decision Tree classifier achieved the highest accuracy of 95.09%, using a subset with only 49 features. The Chi-Square Test with the KNN classifier also demonstrated accuracy (92.16% with 50 features).

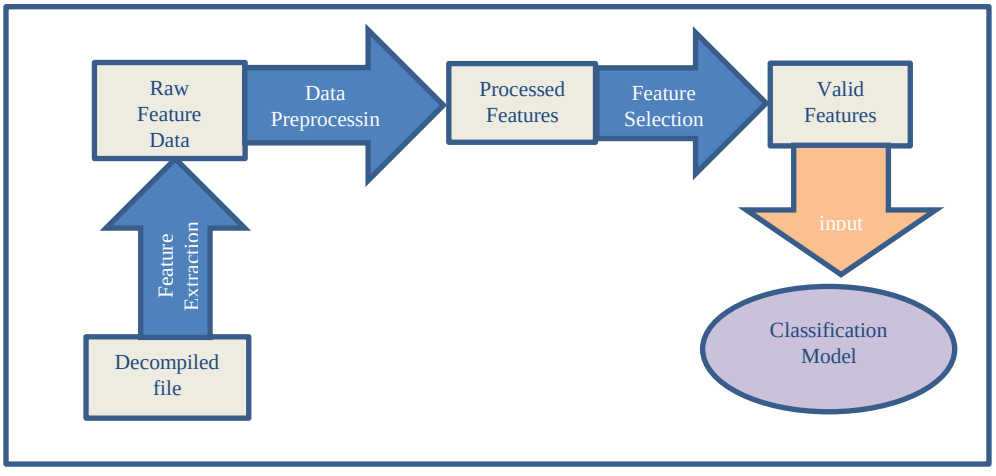

Fig. 3 - the process of feature selection

3.3 Machine Learning (ML) Categories

In the context of Android attack detection, machine learning (ML) can be categorized into several approaches, each with its methodologies and applications. Here are two main categories:

3.3.1 Supervised learning

In supervised learning, the models are trained on labeled data where the inputs and the outputs are known. The desired output is matched with the input dataset. For the program to determine the class of every given piece of data, it contains algorithms that receive a dataset and provide feedback. Regressions, classification trees, random forests, artificial neural networks (ANNs), and support vector machines are just a few of the numerous methods utilized in classification [34]. It can be categorized into two categories.:

- Regression: predicting continuous outcome.
- Classification: forecasting distinct value.

Nuren Natasha Maulat Nasri et al [8] proposed the malware detection system of Android and utilizing five different classifier types when using WEKA to carry out the simulation procedure. The dataset used contains 10000 benign and 10000 malware. So, the Random Forest classifiers they provided achieved the greatest accuracy result of 89.36%. By using RF classifiers on the Drebin malware samples compared to 89.2% which Naïve Bayes attained. Finally, they conclude that the majors can acquire better detection performances with the development of learning classifiers and feature tuning.

Seif ElDein Mohamed et al [9] present a method for detecting Android malware that relies on permissions and API features. Their goal is to assess and investigate the combination of Android characteristics with machine learning classifiers. They investigated different ways to categorize Android malware according to the feature that was being used. Furthermore, they used two different datasets in their successful tests and their results showed that the proposed method achieved an average accuracy of 99% with the Malgenome data set and an average accuracy of 86% with the Maldroid data set.

Abdelrahman Elsharif Karrar [10] employs a graph-based ML technique to manage functionalities of permissions and APIs, using app data from the Drebin project. Machine learning techniques, for example Decision Tree Algorithm (DT), Logistic Regression Algorithm (LR), Random Forest (RF) and K-Nearest Neighbor Algorithm (KNN) are used in the classification process. Finally, the work proposes that the RF classifier realizes the highest recall and accuracy. Additionally, the suggested method required less than ten seconds for analyses achieving 96.80% and 97.30% recall and accuracy, respectively. While DT and KNN deliver (96%) accuracy while LR delivers (95%).

Amarjyoti Pathak et al [11] suggested an Android malware detection method that depend on machine learning that is applied to several datasets to classify malware. The feature importance score is suggested as a feature selection approach in this study to identify the essential permissions computed using gradient boosting. The proposed methodology minimizes the dimension of the feature vector, which reduces the model's training time. The useful insights are discovered through a comparison examination of the algorithm's performance and the Random-forest classifier in all three datasets achieves the highest accuracy with unsavory or no loss in accuracy.

3.3.2 unsupervised learning

Unsupervised learning detects unknown threats by identifying the data without prior labeling. The solution must find out natural grouping, many techniques are used, such as Hierarchical clustering, K-means clustering and Autoencoders. It can be categorized into two categories [34]:

- Clustering: it involves grouping the data into categories that fit the specifications.
- Dimension reduction: which reduce the details while retaining the facts.

William Bradley Glisson et al [35] investigated the efficiency of four machine learning algorithms combined with feature selection from manifest file permissions of Android to categorize apps as harmful or not. The first goal of this research was to evaluate the antivirus software and its effectiveness. This study's second goal was to assess the

performance of various machine learning algorithms when they were just used with the APK manifest file. Hence, of the considered algorithms Random Forest yielded the greatest results, with 82.5% precision and 81.5% accuracy. Gaussian Naive Bayes was the only algorithm to produce less than-ideal results, it had the lowest accuracy (0.6208), specificity (0.1313), F1 score (0.7486), and precision (0.6028).

Shakirat Aderonke Salihu et al [36] provided a performance comparison analysis of four ML techniques for Android malware detection: Naive Bayes (NB), Support Vector Machine (SVM), Decision Tree (DT) and K-means. The study's experiments used 558 APK apps, including 279 examples of benign software from the Google Play Store and 279 malware from MalGenome. While NB had 90% accuracy, K-means, SVM, and DT only managed 94% accuracy. This study effectively showed that ML techniques supervised and unsupervised machine learning approaches, including DT, SVM, and K-means, which have an accuracy of 94% in detecting Android malware, may be used to detect malware on Android devices.

3.4 Performance Measurement

Calculating specific performance metrics and visualizing the confusion matrix is essential to evaluate each classification algorithm. These will be useful for evaluating the effectiveness of different approaches and for analyzing the performance of any method.

3.4.1 Confusion Matrix

A confusion matrix is a table used to assess the performance of a classification model by summarizing the results of predictions compared to the actual classes. It provides a breakdown of correct and incorrect predictions across different classes [37]. It is shown in Fig. 4 .

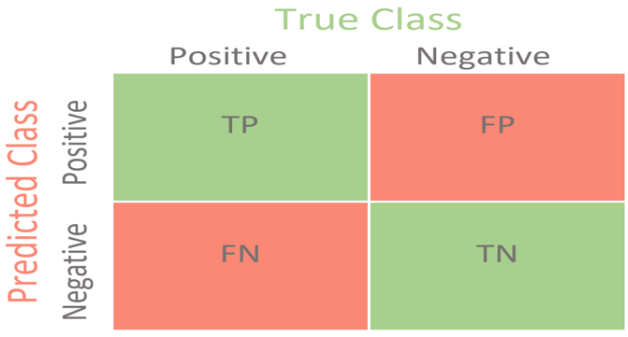

Fig.4 - Confusion Matrix [37]

While:

TP True Positive: indicates that the classifier labeled the positive tuples correctly.

TN True Negative: means the classifier labeled the negative tuples correctly.

FP False Positive: Refers to positive tuples that the classifier categorized incorrectly.

FN False Negative: Refers to negative tuples that the classifier categorized incorrectly [37].

3.4.2 Performance Metrics

The following performance metrics are calculated to determine which model works best.

Accuracy (AC) is a measure that assesses the overall performance of a classification model. It calculates the percentage of right predictions (including true positives and true negatives) among all predictions made. It is defined by using equation (1).

$$Accuracy = \frac{TP+TN}{TP+FN+FP+TN} \quad (1)$$

Precision (P) It measures the proportion of true positive predictions among all positive predictions made by the model. And this can be defined using equation (2).

$$Precision = \frac{TP}{(TP+FP)} \quad (2)$$

Recall known as true positive rate or sensitivity, is a metric used to evaluate the performance of a classification model. It measures the proportion of positive instances that were correctly identified by the model. This measure is shown in equation (3).

$$Recall \ = \frac{TP}{(TP+FN)} \quad (3)$$

F1 Score is a weighted average of recall and Precision or the True Positive (TP) rate as defined in equation (4).

$$F1 = 2 * \frac{Precision*Recall}{Presision+Recall} \quad (4)$$

The ROC Curve is a graphic representation that is used to evaluate the classification model's performance. The trade-off between the true positive rate and the false positive rate at different threshold values is illustrated via this curve [36].

Table (2) summarizes previous works that performed Supervised and unsupervised machine learning algorithms for Android attack detection.

Table (2) summarizes previous works

Study	Dataset	samples	Features selected	Result
[24]	DREBIN MALGENOME	18,835	215 features	89% with RF

[25]	CIC InvesAndMal 2019	(426) malware (5,065) benign	50 features	94% with SVM
[31]	CIC-Maldroid-2020	11,598 samples	231 features	99.49%
[28]	real-world and synthetic datasets	14,000 samples	1912 features	90%
[26]	MalDroid and DefenseDroid	1,00,000 samples	56000 features	96%
[8]	Drebin dataset	10,000 benign 10,000 malwares	15 features	89.36% with RF
[9]	Malgenome and Maldroid	3800 samples	High number of features	99 % with Malgenome 86% with the Maldroid
[30]	Andro CT and Andro Zoo	195623 samples	176 actions and 668 permissions	94%
[32]	virus sharing, Github, and the CIC	1403 samples	16 features	80% with PCA
[29]	Android malware dataset	APK files	less than half of the original feature set	more than 90-91 %
[36]	MalGenome dataset and Google Play store	558 applications	number of features reduced	94% with SVM
[27]	the Drebin data set	123,453 benign 5,560 malware	1,000 features	99%
[33]	CICInvesAndMal2019	1187 benign 407 malware	49 features	95.09%
[35]	Google Play store, AndroZoo	10597 samples	Numerous features	81.5% with RF
[10]	Drebin dataset	15,036 samples	183 features	97.30%
[11]	Defense Droid	10,000 benign 19999 malicious	29 features	93.96%

4. Discussion

Depending on a summarized comparison has been extracted as shown in Table (2). Hence, the following points can be highlighted:

- Many of the studies used static analysis, dynamic analysis was used by smaller number of studies, and the remaining studies used hybrid analysis, as shown in Fig .5 . The high use of static analysis can be related to its lower cost when compared to hybrid and dynamic analyses. Many studies provide evidence that a hybrid approach combining static and dynamic analyses yields superior results, achieving accuracies above 90%.

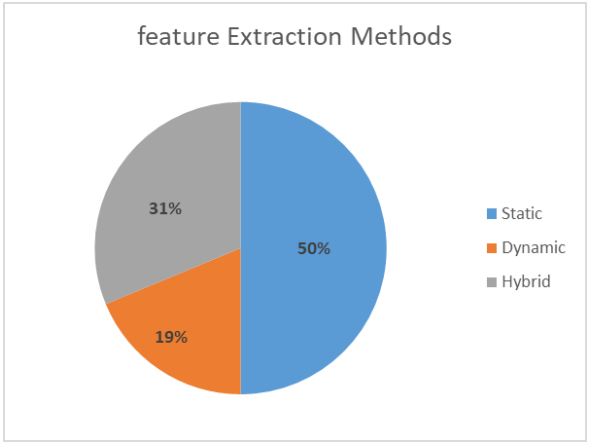

Fig.5 - feature extracted methods used in the reviewed studies

- By using the feature extraction methods, the most widely extracted features are permissions, system calls and API calls. indicating that careful selection and optimization of these features can lead to important improvements in detection performance.

- Many studies used PCA as the feature selection method in the malware detection system, which has proven effective in reducing the dimensionality of datasets without compromising accuracy. On the other hand, a smaller number of studies used another feature selection method like information gain, RF, and Gini impurity.
- Fig. 6 shows the datasets utilized in ML-based Android malware detection research. The most popular dataset for Android malware detection was Drebin. The Drebin dataset is used the most since it offers a complete labeled dataset. The other popular datasets are Maldroid, MalGenome, and CIC InvesAndMal2019.

Fig. 6 - datasets and how they were used in the studied research

- The most widely used ML models to detect Android malware are SVM, RF, KNN, DT and NB since the cost of resources to run SVM, RF, and NB models is low. Less often used models include ensemble learning, logistic regression, and Kmean. Fig. 7 shows all the ML models that have been researched and how they were used in the reviewed papers.

- According to the reviewed studies, it was noticed that high accuracy was achieved by SVM, RF, ensemble learning, DT and KNN machine learning methods. Therefore, this may be why these methods are widely used in the Android malware detection system.

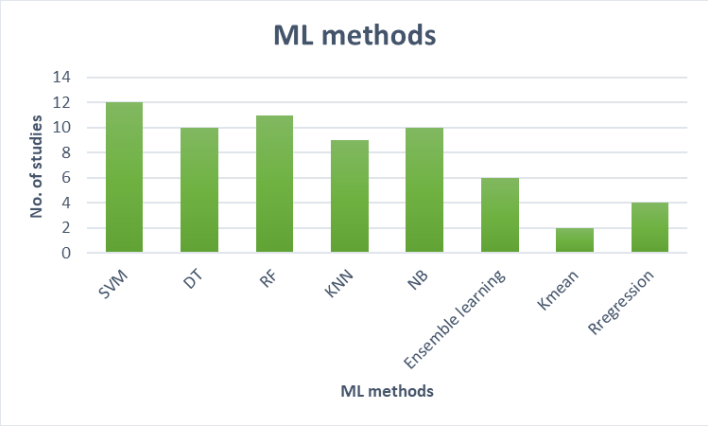

Fig. 7 - ML methods with their usage in the reviewed studies

- Finally, this paper gives a thorough review of different existing literature in the field of Android malware and attack detection using machine learning techniques and proposes doing hybrid for two or more machine learning algorithms to enhance the accuracy and effectiveness of attack detection.

5. Conclusion

Smartphones are inherently vulnerable to security threats. Attackers are more interested in targeting Android devices due to their constant evolution in both hardware and applications. This is primarily due to Android's open-source nature and its greater market share compared to other mobile operating systems. The Android architecture and its particular attacks were covered in this study. It also reviews recent research on machine learning-based techniques for Android malware detection, focusing on studies published between 2020 and 2024. The discussion includes an analysis of available machine learning models and their effectiveness in detecting Android malware, core ideas, datasets, and performance metrics of the current techniques, feature extraction methods, and feature selection methods. In conclusion, the integration of advanced machine learning techniques with effective feature selection and a hybrid analysis approach presents a promising path for enhancing Android malware detection systems. As mobile threats continue to improve, these studies collectively advocate for outstanding research and development to ensure robust security measures are in place, safeguarding devices from malicious applications.

Acknowledgements

The authors thank the Department of Computer Science, College of Science Mustansiriyah University, for supporting this work.

References

[1] Smartphone Market Share. Accessed: Apr. 30, 2020. [Online]. Available: https://www.idc.com/promo/smartphone-market-share/os.
[2] R. Srinivasan, Karpagam.S, M. Kavitha, R. Kavitha, An Analysis of Machine Learning-Based Android Malware Detection Approaches. International Conference on Electronic Circuits and Signalling Technologies, 2022.
[Online]. Available: https://www.appbrain.com/stats/number-of-androidapps.
[3] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-an, and H. Ye, ``Signifcant permission identifcation for Machine-Learning-Based Android malware detection," IEEE Trans. Ind. Informat., vol. 14, no. 7, pp. 32163225, Jul. 2018.
[4] J. Todd McDonald, Nathan Herron, William Bradley Glisson, Ryan K. Benton, Machine Learning-Based Android Malware Detection Using Manifest Permissions. 54th Hawaii International Conference on System Sciences | 2021.
[5] Sufatrio, D. J. J. Tan, T.W. Chua, and V. L. L. Thing, ``Securing Android: A survey, taxonomy, and challenges," ACM Comput. Surv., vol. 47, no. 4, p. 58, May 2015.
[6] Qing Wu , Xueling Zhu , and Bo Liu," A Survey of Android Malware Static Detection Technology Based on Machine Learning", Mobile Information Systems Volume 2021, Article ID 8896013, 18 pages.
[7] A. Feizollah, N. B. Anuar, R. Salleh, and A. W. A. Wahab, "A review on feature selection in mobile malware detection," Digital Investigation, vol. 13, pp. 22–37, 2015.
[8] Nuren Natasha Maulat Nasri, Mohd Faizal Ab Razak, RD Rohmat Saedudin, Salwana Mohamad Asmara and Ahmad Firdaus, Android Malware Detection System using Machine Learning. International Journal of Advanced Trends in Computer Science and Engineering, vol.9, no.1, 2020, 327 – 333.
[9] Eslam Amer, Seif ElDein Mohamed, Mostafa Ashaf, Amr Ehab, Omar Shereef, Haytham Metwaie, Ammar Mohammed, Using Machine Learning to Identify Android Malware Relying on API calling sequences and Permissions. Journal of Computing and Communication Vol.1, No.1, PP. 38-47, 2022.
[10] Abdelrahman Elsharif Karrar, Adopting Graph-Based Machine Learning Algorithms to Classify Android Malware. IJCSNS International Journal of Computer Science and Network Security, VOL.22 No.9, September 2022.
[11] Amarjyoti Pathak, Utpal Barman, Th. Shanta Kumar, Machine learning approach to detect android malware using feature-selection based on feature importance score. Journal of Engineering Research, 2024.
[12] D. Youchao, "Android Malware Prediction by Permission Analysis and Data Mining," Univ. Michigan-Dearborn, 2017.
[13] Brownlee, J. (2016). Supervised and Unsupervised Machine Learning Algorithms. Retrieved May 13, 2018, from http://machinelearningmastery.com/supervised-andunsupervised- machine-learning-algorithms/ https://doi.org/10.1007/978-3-030-22475-2_1.
[14] Garg, B. (2013). Design and Development of Naive Bayes Classifier (Master Thesis). North Dakota State University of Agriculture and Applied Science.
[15] Namratha, M., & Prajwala, T. R. (2012). A Comprehensive Overview of Clustering Algorithms in Pattern Recognition. Journal of Computer Engineering, 4(6), 23–30.
[16] Scott, G. (2015). ML 101: Reinforcement Learning. Retrieved November 23, 2017, from http://scottge.net/2015/07/02/ml101-reinforcement-learning/.
[17] Guyon, I., & Elisseeff, A. (2006). Feature Extraction, Foundations and Applications: An introduction to feature extraction. Studies in Fuzziness and Soft Computing, 207, 1–25. https://doi.org/10.1007/978-3-540-35488-8_1
[18] 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).
[19] Platform Architecture. Available online: https://developer.android.com/guide/platform (accessed on 19 May 2021).
[20] Android Runtime (ART) and Dalvik. Available online: https://source.android.com/devices/tech/dalvik (accessed on 19 May 2021).
[21] Faruki, P., Bharmal, A., Laxmi, V., Ganmoor, V., Gaur, M.S., Conti, M., Rajarajan, M., 2015. Android security: a survey of issues, malware penetration, and defenses. IEEE Commun. Surv. Tutor. 17 (2), 998–1022. doi: 10.1109/COMST.2014.2386139.
[22] Sikder, R., Khan, M.S., Hossain, M.S., Khan, W.Z., 2020. A survey on android security: development and deployment hindrance and best practices. TELKOMNIKA (Telecommun. Comput. Electron. Control) 18 (1), 485. doi: 10.12928/telkomnika. v18i1.13288.
[23] Li Meijin, Fang Zhiyang, Wang Junfeng, Cheng Luyu, Zeng Qi, Yang Tao, Wu Yinwei and Geng Jiaxuan, A Systematic Overview of Android Malware Detection, APPLIED ARTIFICIAL INTELLIGENCE 2022, VOL. 36, NO. 1, e2007327 (524 pages).
[24] Ayman Elshenawy, Ahmed Hashem El Fiky, and Mohamed Ashraf Madkour, Detection of Android Malware using Machine Learning. IEEE, (2021).
[25] Ahmed S. Shatnawia, Qussai Yassen, Abdulrahman Yateem, An Android Malware Detection Approach Based on Static Feature Analysis Using Machine Learning Algorithms. Procedia Computer Science, 201 (2022) 653 – 658.
[26] Beenish Urooj, Munam Ali Shah, Carsten Maple, Muhammad Kamran Abbasi and Sidra Riasat, Malware Detection: A Framework for Reverse Engineered Android Applications Through Machine Learning Algorithms. Digital Object Identifier 10.1109/IEEE ACCESS.2022.3149053.
[27] Vasileios Syrris, Dimitris Geneiatakis, on machine learning effectiveness for malware detection in Android OS using static analysis data. Journal of Information Security and Applications 59 (2021) 102794.
[28] Faris Mutar Mahdi Aledam, Bilal Majeed Abdulridha Al-Latteef, Enhanced Malware Detection for Mobile Operating Systems Using Machine Learning and Dynamic Analysis. International Journal of Safety and Security Engineering, 14 No. 2 (2024) 513-521.
[29] R. Srinivasan, Karpagam.S, M. Kavitha, R. Kavitha, An Analysis of Machine Learning-Based Android Malware Detection Approaches. International Conference on Electronic Circuits and Signalling Technologies, 2022.
[30] Ahmed S. Shatnawi, Aya Jaradat, Tuqa Bani Yaseen, Eyad Taqieddin, Mahmoud Al-Ayyoub, and Dheya Mustafa, An Android Malware Detection Leveraging Machine Learning. Hindawi, Wireless Communications and Mobile Computing Volume 2022, Article ID 1830201, 12 pages.
[31] Sakib Shahriar Shafin, Md. Maroof Ahmed, Mahmud Alam Pranto, Detection of Android Malware using Tree-based Ensemble Stacking Model. IEEE, (2023).
[32] V. Joseph Raymond and R. Jeberson Retna Raj, Investigation of Android Malware with Machine Learning Classifiers using Enhanced PCA Algorithm. Computer Systems Science & Engineering, CSSE, 2023, vol.44, no.3.
[33] K S Ranadheer Kumar and Jagadish Gurrala, Enhancing Android Malware Detection through Filter-Based Feature Selection and Machine Learning Classification. Journal of Electrical Systems 20-3 (2024): 2801-2809.
[34] Louridas, P., & Ebert, C. (2016). Machine Learning. IEEE Software, 33(5), 110-115. doi: 10.1109/MS.2016.114
[35] J. Todd McDonald, Nathan Herron, William Bradley Glisson, Ryan K. Benton, Machine Learning-Based Android Malware Detection Using Manifest Permissions. 54th Hawaii International Conference on System Sciences | 2021.
[36] Shakirat Aderonke Salihu, Sodiq Olamilekan Quadri, Oluwakemi Christiana Abikoye, Performance Evaluation of Selected Machine Learning Techniques for Malware Detection in Android Devices. Iloein journal of computer science and information technology (ILJCSIT) ISSN:2550 – 7214 (print) Vol. 3, No. 1 (2020)
[37] Mohamed Salem Alhebsi, Android Malware Detection using Machine Learning Techniques, Rochester Institute of Technology RIT Digital Institutional Repository (2022)
[38] C. Urcuqui-L´opez, Navarro Cadavid, Framework for malware analysis in Android, Sist. Y. Telem. ´atica 14 (37) (2016) 45–56, https://doi.org/10.18046/syt. v14i37.2241. https://www.icesi.edu.co/revistas/index.php/sistemas_telematica/ article/view/2241.