DOI: 10.56892/bima.v7i01.405

SURVEY ON LITERATURES FOR THE DETECTION OF ANDROID MALWARE USING MACHINE LEARNING

[1]*MOHAMMED IBRAHIM, [1]ABDULLAHI ABDULLAHI, [2]ALHASSAN ADAMU, [3]AMINU USMAN JIBRIL and [1]KHALID HARUNA

[1]Department of Computer Science, Kaduna State University (KASU), Nigeria
[2]Department of Computer Science, Aliko Dangote University of Science and Technology, Wudil, Kano, Nigeria
Department of Cimputer Science, Bayero University Kano (BUK), Nigeria

Corresponding Author: mohammed.ibrahim@kasu.edu.ng

ABSTRACT

Android Operating System is an open source operating system with high efficiency and flexibility, which has led to enormous acceptance globally. Its populous prompts the advent of Android malware with the aim of invading users' information without their knowledge and posing a threat to the android community at large. For that reason, a great number of signature-based tools to detect Android malware are available on the market, but they can't detect unknown Android malware. Thus, many researchers have conducted studies using machine learning techniques to detect Android malware, and the results have proved promising for detecting both known and unknown Android malware. This paper gives a study of machine learning-based methods for Android malware detection. In this regard, it succinctly provides a little background on Android applications and the Android Dataset. Besides, a critical evaluation of existing works on machine learning for detecting Android malware, the analyzes and summarizes a number of research papers based on sample collection, feature choice, model strength, and model problem for the benefit of the research community and identifies the areas that require additional study in spite of the dynamics of both Android technology and the related advancements in malware penetration.

Keywords: Android, Dataset, Detection, Malware, Operating System.

INTRODUCTION

Since the introduction of Android in September 2008, it has dominated the mobile industry due to a variety of characteristics that have made it the most widely used mobile platform worldwide. Because of its features and rapid growth (Arnab Chakraborty, 2023). Android is one of the most popular operating systems, and by the end of 2022, it will own around 71.8% of the market's shares globally (Petroc Taylor, Feb 21, 2023). Despite the significant increase in Android users, malware authors have taken advantage of this growth to negatively affect and steal a large number of users' information. Currently, a number of mobile operating systems, including iOS from Apple, Blackberry, Symbian, Windows Mobile, and Android from Google, serve the majority of mobile devices. Google's Android has been monopolizing the mobile OS market out of these five widely used mobile operating systems. With an 80% market share in the third quarter of 2013, Android outperformed other operating systems (van der Meulen R. & Rivera J., 2013). It was also shown that in 2016 Android remain the most leading operating system in the smartphone companies and contributing to more than 81% of devices (Martin et al, 2018).

The Android operating system (OS) has grown to play a significant role in the market for mobile devices and regrettably, the popularity of Android and the facilities it renders to develop and upload applications have negative effects in some instance (Martin et al, 2018). Consequently, Android application attracted the attention of malware developers. In order to gain a better understanding of Android malware and provide a practical way to protect mobile devices from mobile malware, the authors of this paper evaluated a number of recent studies on mobile malware. Additionally, this study looks into a large number of studies that have been conducted to protect Android mobile devices from malicious applications with the intention of categorizing the current mobile malware detection approach, datasets used, and model performances.

Through analysis and evaluation of the review works, several machine learning techniques have centered their proposed techniques on detecting malware infiltration on Android phone devices. Nevertheless, the accuracy of detecting this malware remains a hot issue. Many works have focused on feature selections to detect Android malware, while others detect the malware using non future selections. For the benefit of the research community, this paper critically highlights the areas that need further research considering the dynamics of both Android technology and the corresponding advances in malware infiltration. As such, these dynamic changes in both technologies required a dynamic approach, like machine learning, that can significantly detect future malware on Android phone.

The paper is organized as follows: section 2 overview the Android malware, section 3 related work, section 4 exploration of a research gap identified from the related works and section 5 conclude the review of the existing work.

ANDROID MALWARE

Mobile malware is malicious code designed to harm a user's device, and these malware authors lure users to install applications which will allow them to gain unauthorized root access to an infected device, or they deceit the user through various traditional sources such as embedding malicious code in emails, shuffling dubious websites, or repackaging original apps with the malicious code for update purposes. Once the malicious code is rooted into users' devices, the malicious functionality will take place in the background while the user is exploring the application. For example, if the user installs or updates an app containing malicious code unknowingly, when the app containing malicious code is loaded for use, the malicious functionality will start carrying out dubious acts in the background, such as sending short message service (SMS), stealing and exacting your personal information for dubious purposes. Malware include viruses, trojan horses, worms, and botnets, is often found on desktop computers and is very uncommon on mobile devices. However, as mobile device technology progresses to a high standard and therefore is able to support complicated operating systems, it has become the next target for malware authors. Android malware is on the rise as a result of the market's strong adoption of the Android operating system.

A trojan infection was found on the Google Play Store in the middle of 2017 disguised as the game "Colourblock". Over 50,000 people

DOI: 10.56892/bima.v7i01.405

unknowingly downloaded spyware after believing it to be a game. The trojan, known as Dvmap, gave attackers the ability to monitor the device it was installed and even install new software to them (Kanal S. Sajan, 2022).

Dataset for android malware detection

To ensure that the learned model can be applied trustfully to make predictions on new data, it is crucial to train the model with high-quality samples of data (Zhou, 2016). If the sample data is not adequate and representative, it may result in incorrect results. The sample data for a classification issue in the detection of Android malware shouldn't be excessively biased in terms of the proportion of benign and malicious applications. Obtaining examples of safe Android applications is a rather simple process. App stores should be regarded as a reliable source of safe programs because Android applications that are available in different app stores are typically subject to rigorous testing before they are released (Liu et al., 2020). The collection of samples for malware detection was mostly represented as a form of dataset.

Table 1 contains Android malware dataset; most Android malware researchers use this to evaluate the performance of their malware detectors because it remains one of the fastest sources of malware samples. Most old dataset features are static with a small sample size, while dynamic features are just provided in some more recent datasets. The collection of static features is quicker and simpler than that of dynamic features. While dynamic feature collection requires the use of real devices or Android emulators, static analysis examines malware files without really running the application. This distinction may have an impact on the quality of the data gathered. Based on the information contained in table 1, most of the datasets are considerably small in sample size and, in some cases, they must be combined with other datasets or sources of malware in order to get the largest and most complete picture of the Android malware history and evolution. For such purposes, database for Android malware repository provides accessible general malware repositories that also contain Android malware samples. These repositories are a remarkable source of malware that has already been used to complement and enrich existent datasets for research. Mostly designed as database services, they are growing repositories of malware samples. More specifically, *VirusTotal* and *VirusShare* are upon request malware repositories, and *AndroZoo* is a large repository of Android applications, but with unknown proportions of malware and benign apps.

Metric system for measuring classifier's

There are a number of ways to gauge a classifier's effectiveness, and this study places particular attention on the metrics used for each research report evaluation. The following tables show the performance of the metric as it was shown in the study report under consideration.

As demonstrated by a confusion matrix in Table 2 ("Machine Learning Glossary"), ("Interpretation of Performance Measures"), when predicting whether an Android application has malware, the results can be divided into four groups using a traditional binary classification issue.

DOI: 10.56892/bima.v7i01.405

Table 1: List of Android dataset (×=Not available)

S/no	Dataset	Year	Analysis Type	samples	Benign	Malware	Ref.
1	CICMalDroid 2020	2020	static and Dynamic	17,341	Benign	x	CICMalDroi, 2020
2	Android Malware Static Analysis (CCCS-CIC-AndMal-2020)	2020	static	400k	x	200k	CCCS-CIC-AndMal,2020
3	KronoDroid	2020	x	78137	200k	41,382	Hayretdin Bahsi & Sven Nõmm. (2021)
4	Investigation of the Android Malware (CICInvesAndMal2019)	2019	static and Dynamic	5491	36,755	426	CICInvesAndMal,2019
5	Android Malware Dataset - Kaggle	2018	x	15036	5,065	5,560	Android Malware Dataset-kaggle
6	MalDozer	2018	Static	71000	9476	33000	Karbab et al., (2018)
7	AMD Project	2017	static	405	38000	405	Li Y, Jang J & Hu X. (2017)
8	Android Malware Dataset (CICAndMal2017	2017	static and Dynamic	10854	x	4,354	CICAndMal2017
9	Kharon Malware Dataset	2016	Dynamic	7	6,500	7	Kiss et al., 2016
10	Android Adware and General Malware Dataset (AAGM)	2016	Dynamic	1900	x	400	CIC-AAGM2017
11	Android PRAGuard Dataset	2015	static	10479	1500	x	Android PRAGuard Dataset
12	M0Droid	2015	static and Dynamic	Signature Base	x	x	M0Droid
13	ISCX Android Botnet dataset 2015	2014	x	1,929	x	x	ISCX Android Botnet dataset, 2015
14	Android validation dataset	2014	x	792	x	x	Android validation dataset
15	Drebin	2014	Static	129013	x	5,560	Hubner et al.,

							(2014)
16	ContagioDump	2013	Static	28760	123453	11,960	ContagioDump
17	Genome Project	2012	static	1260	16,800	x	Genome Project
18	AndroZoo	x	x	+20m	x	x	AndroZoo
	Total				x	103047	

Table 2: Confusion matrix of predicted results.

		Prediction Class	
		Positive	Negative
Actual Class	Positive	TP	FN
	Negative	FP	TN

The concepts of FP, FN, TP, and TN are defined as follows.

A. True positive (TP): the application is a malicious application and was correctly predicted to be malicious.

B. False positive (FP): the application is a benign application but was wrongly predicted to be malicious.

C. True negative (TN): the application is a benign application and was correctly predicted to be non-malicious.

D. False negative (FN): the application is a malicious application but was wrongly predicted to be benign.

A number of performance indicators have been derived using these four fundamental ideas as the foundation. Here are a few metrics that are frequently used.

A. **Accuracy** - Accuracy is the most intuitive performance measure, and it is simply a ratio of correctly predicted observations to the total observations. One may think that if we have high accuracy, then our model is the best. Yes, accuracy is a great measure, but only when the symmetric datasets have almost identical false positive and false negative values.

Accuracy = TP+TN/TP+FP+FN+TN

B. **Precision** - Precision is the ratio of correctly predicted positive observations to the total predicted positive observations.

Precision = TP/TP+FP

C. **Recall (Sensitivity)** - Recall is the ratio of correctly predicted positive observations to the all observations in actual class.

Recall = TP/TP+FN

D. **F1 score** - F1 Score is the weighted average of Precision and Recall. Therefore, both false positives and false negatives are considered while calculating this score. Although F1 is typically more valuable than accuracy, especially when there is an unequal class distribution, it is not intuitively as simple to understand as accuracy. When false positives and false negatives cost the same, accuracy performs best. It is preferable to consider both precision and recall if the costs of false positives and false negatives are significantly different.

F1 Score = 2*(Recall * Precision) / (Recall + Precision)

DOI: 10.56892/bima.v7i01.405

RELATED WORK

A survey of the literature on Android malware detection using machine learning approaches is presented in this section. Methods for analyzing Android malware enable the collection of various features that are then used to define and construct machine learning systems. These features are static, dynamic and hybrid analysis.

This part explains the idea of related research, datasets in the study area, and a brief overview of the methods and common features used for machine learning-based Android malware detection.

Kurniawan et al. (2015). Proposed Android Anomaly Detection System Using Machine Learning Classification. This study examines anomalies in battery temperature, network traffic, and power usage as features for machine learning classification algorithms (Support Vector Machine (SVM), Random Forest (RF) and Logistic Model Tree (LMT)). Combining the three features (power consumption, battery temperature, and network data traffic) with the SVM classifier, which has the maximum 85.6% accuracy in detecting abnormality, produces the result to detect anomaly.

Hsin-Yu Chuang and Sheng-De Wang (2015): Propose Machine learning based hybrid behavior models for Android malware analysis. This study employed a method that emphasizes static analysis. Collecting data from the dataset (“contagiodump”), doing a frequency analysis (Wu et al., 2016), and obtaining the features of the data in a broad sense. The number of Android API calls that are made are calculated and sorted according to their usage from benign and malicious applications, respectively, by the frequency analysis. Using the statistics, two separate feature sets are produced. The first is a list of APIs that are more frequently utilized by legitimate applications than by malicious ones. The second is a list of APIs that are more frequently utilized by malicious than by good applications. We took into account the preferred APIs by good applications as well as the often-used APIs by Android malware when analyzing the two distinct behavior elements. We next use the support vector machine (SVM) algorithm, a well-known machine learning method, on the two feature sets to create the corresponding decision models. Our hybrid model classifier is created by combining the two decision models utilizing fusion logics. The SVM scores provide the basis of the fusion logics.

Westyarian et al. (2015): Propose Malware Detection on Android Smartphones using API Class and Machine Learning. In order to distinguish between malicious and benign applications in an android environment, this research effort uses API calls as characteristics to the three classifiers (SVM, J48, and RF). Additionally, using 16 API classes and 51 packages, there are 412 sample Android applications, 205 of which are benign, and 207 of which are malicious.

Wu et al. (2016): Propose Effective Detection of Android Malware Based on the Usage of Data Flow APIs and Machine Learning. Dataflow Application Program Interfaces (APIs) are used in this research's machine learning technique to identify Android malware as classification features. A thorough analysis was conducted in order to collect API-level dataflow information for the k-nearest neighbor (KNN) classification model. Additionally, 1,160 benign and 1,050 malicious samples totaling 2210 apk files were utilized to test the suggested system. The results show that the system has an accuracy rate of up to 97.66% for detecting unidentified Android malware. According to our static data-flow analysis experiment, the new API subset enables the discovery of over 85% of sensitive data transfer channels while cutting down on analysis time by roughly 40%.

Long Wen and Haiyang Yu, (2017). Present an Android Malware Detection System Based on Machine Learning, features were extracted from the APK files based on static and dynamic analysis. These features were reduced using Principle Component Analysis (PCA) and relief. The model is divided into two main sections: the client and server side. On the client side, it mainly provides the UI (user interface) for the users and triggers an alert when a prediction occurs. However, due to limited resources, a simple check was made on the client side by extracting the value of MD5 whenever a new application is installed and comparing its value with the malicious MD5 value stored on the server side. If the matches are found, the model triggers a malicious alert to the client with the option of deleting the files. Else, the APK file is submitted to the server for the feature's extraction using static and dynamic analysis using marching learning classifier SVM and evaluates the unknown Android application by classifying it into malware or benign.

Milosevic et al. (2017): They Propose Machine Learning Classifiers for Android malware. This research work presents two machine learning-aided approaches for static analysis of Android malware. The first approach is based on permissions, and the other is based on source code analysis. Manifest analysis and code analysis were used as features by machine learning classifiers to detect malicious Android apps. These two techniques of machine learning assisted (SVM and clustering) were based on app permissions and source code analysis to detect and analyze malicious Android apps. That's because SVM uses extracted permissions while clustering uses code analysis. The M0Droid dataset used contains 200 malicious and 200 benign Android apps.

Kakavand et al. (2018). Propose Application of Machine Learning Algorithms for Android Malware Detection. In order to improve malware detection findings, this research project involves static app analysis, which requires examining the presence and frequency of keywords in the Android application manifest file and creating static feature sets from a dataset of 400 apps. The accuracy and true positive rate of the ML algorithms' classification performance are evaluated and examined in order to determine which approach is better suited for Android malware detection. SVM and KNN algorithms, the two most promising machine learning (ML) classifier techniques identified from earlier research, are used to assess both the user permissions and intent filters requested in an Android app's manifest file.

The main objective of this study is to determine whether using the two machine learning algorithms and looking for keywords in the permissions requested in an app's manifest file and system call logs may improve our ability to identify malicious apps. The experimental results for a dataset of real malware and benign apps show average accuracy rates of 79.08 percent and 80.50 percent, respectively, with an average true positive rate of over 67.00 percent and 80.00 percent.

Oktay Yildiz & Ibrahim Alper Doğru, (2019). Present Permission-based Android Malware Detection System Using Feature Selection with Genetic Algorithm (GA). This study suggests a feature selection approach for identifying Android malware using a genetic algorithm (GA). However, three different classifier methods (Decision Tree (DT), Naive Bayes (NB), and SVM) with varied feature subsets were created and compared using GA

DOI: 10.56892/bima.v7i01.405

to detect and analyze Android malware. This classifier SVM achieves the best accuracy result of 98.45% with the 16 stated permissions and a dataset of 1740 samples containing 1119 malwares and 621 benign samples. With 152 permissions, the accuracy drops to 96.92% for both features supplied by GA.

Ma et al. (2019). A Combination Method for Android Malware Detection Based on Control Flow Graphs and Machine Learning Algorithms. A machine learning-based combination method for identifying Android malware is presented in this study. Decompile the Android application and build the control flow graph (CFG) from the source code to acquire API information. Extracting API calls from the CFG will allow you to create three different kinds of API data sets: Boolean data sets, frequency data sets, and chronological data sets.

Based on API calls, API frequency, and API sequence, three detection models are built for Android malware detection using these three types of data. Finally, studies using a machine learning ensemble meta-algorithm on 10010 benign and 10683 malicious applications were conducted. The findings show that our detection model achieves 98.98% detection precision, as well as excellent accuracy and stability.

Han et al. (2020). Enhanced Android Malware Detection: An SVM-based Machine Learning Approach. SVM, a machine learning classifier, and API calls are used in this study as features that were taken from the Android program files, or APK files. The 58,602 Android applications were utilized to extract 133,227 features via static analysis. However, the experiment employed the Drebin dataset for malware identification, which included 30,113 dangerous apps and 28,489 benign apps. The testing result indicates 99.75% total accuracy after sounds were eliminated from 133,227 to 41,545 API features.

Roy et al. (2020). Proposed Android Malware Detection based on Vulnerable Feature Aggregation. The Android API (application programming interface) calls are used in this research to extract features, combine them to determine the overall frequency of each characteristic, and represent them in a single tuple per Android Package Kit (APK) file.

Non-negative matrix factorization (NMF), a powerful machine-learning technique for reducing the overall number of features, is used to make our model lightweight and scalable. The efficacy of this feature set is assessed with the use of numerous machine learning classifiers (Logistic Regression, KNN, SVM, and RF). The best accuracy achieved with RF is 93.77%, while the highest accuracy with SVM is 93.35%, and the highest accuracy with non-reduction features is 88.72%.

McDonald et al. (2021). Machine Learning-Based Android Malware Detection Using Manifest Permissions. In this study, it is examined how well four different machine learning algorithms perform at classifying programs as harmful or benign using features taken from the Android manifest file permissions. Results from a case study on 5,243 test samples show accuracy, recall, and precision rates of above 80%. Random Forest outscored the other algorithms with 82.5 percent precision and 81.5 percent accuracy (SVM, Gaussian Nave Bayes (GNB), and K-Means(KM)).

S. Abijah Roseline & S. Geetha, (2021). Present Android Malware Detection and Classification using LOFO Feature Selection and Tree-based Models. The malware detection method for Android presented in this work classifies malware applications based on the most important features using tree-based learning models. The DREBIN data set, which contains 15,036 samples—5560 of which are malicious apps and 9476 of which are benign applications—is used for the experimental evaluation. Each sample has 215 attributes gleaned from static code analysis to show the effectiveness of the suggested strategy. With a small number of features, the XGBoost classifier surpasses other tree-based models with prediction accuracy of 95.59%.

Sahin et al. (2021). Proposed a novel permission-based Android malware detection system using feature selection based on linear regression. This paper proposes a malware detection strategy based on machine learning to discriminate between malicious and legitimate Android applications. The feature selection stage of the proposed malware detection system tries to minimize redundant characteristics by using a feature selection technique based on linear regression. The classification model can now be used to real-time malware detection systems, with the training period shortened and the dimension of the feature vector reduced. When the study's results are analyzed, it is shown that using at least 27 characteristics results in the greatest F-measure score of 0.961.

Sahin et al. (2021). Proposed a novel Android malware detection system: adaption of filter - based feature selection methods. This approach makes use of machine learning-based static malware detection for Android. The system is built to use features that are derived from application file permissions. To speed up the processing time and increase the effectiveness of machine learning algorithms, dimension reduction is carried out using eight different feature selection techniques. The remaining document frequencies (threshold, relevance frequency, feature selection, etc.) are adapted from text classification studies. Android malware detection systems use four of these document frequencies Information Gain (IG), Odds ratio, Chi-square, and Inverse document frequency. The retrieved features and classification outcomes of the modified approaches are contrasted. When the results are analyzed, it is clear that the modified approaches, which may be applied in this field, increase the effectiveness of the classification algorithms.

Arif et al (2021). Proposed a static analysis approach for Android permission-based malware detection systems. The Drebin dataset for malware applications (5000 malware) and the Androzoo dataset for benign applications (5000 benign) were integrated into a database for training and testing sets in this study. The datasets were filtered as unsupervised and randomized after preprocessing. The best permission features were then provided by static analysis using particle swarm optimization (PSO), information gain, and evolutionary computation. To identify malware and other threats, five machine learning classifiers (RF, MLP, kNN, J48, and Adaboost) were used to evaluate the feature selection strategies. Each classifier's performance is assessed using five metrics, including True Positive Rate (TPR), False Positive Rate (FPR), precision, recall, f-measure, and accuracy.

Kumar et al. (2022). Analysis of Malware in Android Features Using Machine Learning. An efficient machine learning-based approach for detecting Android malware is presented in this research paper and is based on an evolutionary genetic algorithm. The SVM and the Neural Network (NN), two machine learning classifiers, are trained using the optimal set of features produced from the genetic method, and their performance in identifying malware is compared before and after the features are picked. Genetic

DOI: 10.56892/bima.v7i01.405

algorithms were used to cut the initial set of features to half of what they were. Our research corroborates this. Machine learning-based classifiers maintain over 94 percent classification accuracy after feature selection, decreasing the computational burden of learning classifiers by handling significantly smaller feature dimensions.

Shatnawi et al. (2022). An Android Malware Detection Leveraging Machine Learning. In order to classify the harmful and beneficial applications in the android environment, this research work looked at the analysis of static, dynamic, and hybrid applications. They suggest an Android malware detection model based on static, dynamic, and hybrid analysis along with machine learning classifiers and use the feature rank approach, as this method leverages certain critical elements in feature arrangement because it has the capacity to select the proper features required to build malware detection models. Then, in order to find the best accurate algorithm, they apply a variety of machine learning algorithms (including gradient boosting, XGBoost, Decision Tree (DT), and RF) and compare their results. The accuracy obtained from static, dynamic, and hybrid analyses was over 94%, according to the results, therefore in these situations using static analyses alone should be effective and less expensive for classification.

Urooj et al. (2022). Malware Detection: A Framework for Reverse Engineered Android Applications through Machine Learning Algorithms. This study uses machine learning techniques and reverse-engineered Android application features to find weaknesses in smartphone apps. Two things determine if this endeavor is successful. First, a model that, in comparison to conventional approaches, combines more novel static feature sets with the largest available datasets of malware samples. Second, in order to boost our model's performance, ensemble learning was integrated with machine learning techniques like AdaBoost, SVM, and others. Detecting malware from Android applications is 96.24% accurate, with a 0.3 FPR, according to the trial results and findings (FPR).

Amer et al. (2022). Using Machine Learning to Identify Android Malware Relying on API calling sequences and Permissions. An Android malware detection method based on APIs and permissions is presented in this research paper. The objective is to evaluate and look into how well-known Android features like APIs and permissions interact with machine learning classifiers. They looked into a number of techniques for classifying Android malware according to the feature being used. They investigated the performance of every machine learning classifier for Android malware detection. Additionally, pretreatment and processing are both incorporated in this study project. Preprocessing is the process of extracting features from Android apps by getting the permissions and API calls that are utilized the most frequently. 3,800 different Android applications were collected from the Malgenome data collection as the input. The Maldroid dataset contains safe apps, adware, banking malware, and mobile riskware, just as the Malgenome dataset has permissions and API calls for both good and bad apps. During the processing phase, the data set was divided into three subgroups: training, validation, and testing. Multiple models were tested and trained on the training set using a number of techniques, including KNN, NB, SVM, and DT.

Ahmed et al. (2022). Proposed an Android Malware Detection Approach Based on Static Feature Analysis Using Machine Learning Algorithms. The permission and API call features from the (CIC InvesAndMal2019) dataset were used in this research work to propose a static base classification method for Android malware detection, and feature importance selection using the Recursive Feature Elimination (RFE) on the Logistic Regression model was then applied. The model is built on the SVM, KNN, and NB machine learning techniques, which demonstrate that the (SVM) classifier had the greatest rates when other classifiers were compared. In an effort to achieve high malware detection rates, it provided an average accuracy rate of 83% when utilizing API call features and 94% when using permission features.

Mohamed Salem Alhebsi (2022). Present Android Malware Detection using Machine Learning Techniques. On two independent datasets, this research study used permission-based and signature-based techniques to differentiate between legitimate and malicious programs. To differentiate between malicious and good applications, various classification models (kNN, Logistic Regression, and RF) were developed. While the second data source provides details on the apps' API call signatures, the first data source provides details on the permissions given to the applications. Utilizing three feature selection methods including frequency counts, correlations, and chi-square, the classifiers were given twenty of the best features. After the models have been trained, analysis and comparison are done on their performance indicators, including recall and precision. In order to begin, this comparison is conducted for several classification models inside an approach. The best outcomes for each strategy are then contrasted to determine which of the two methods is more effective in detecting malware. Random Forest and kNN Classifier are the best models for the permissions-based and signatures-based approaches, respectively.

Akbar et al. (2022). Permissions-Based Detection of Android Malware Using Machine Learning. From Android application packages, the proposed Permission-based Malicious Apps detection system (PerDRaML) extracts permission. However, PerDRaML concentrates on a subset of permissions that are efficient in differentiating and enhancing malware detection rates rather than assessing all requested permissions, and the relevant permissions were enumerated using Random Forest-based feature importance. The proposed method classified data using the SVM, Rotation Forest, NB, and RF classifiers, with an average accuracy of 89.7% for the SVM model, 89.96% for the RF model, 86.25% for the Rotation Forest model, and 89.52% for the NB model.

Table 3 contains a summary of techniques used in malware infiltration. This summary could provide an insight to the Android malware researchers on what areas need additional research based on features, analysis types, and the age of the datasets.

Figure 1 displays the number of sample programs (benign and malware) that each algorithm uses to detect Android malware using machine learning, and it is clear that some algorithms use a small number of sample applications while others utilize a large number of sample applications to separate malware from benign software. While those with a small dataset can only have a limited number of features, and these algorithms will have the disadvantage of being evaded by newer malware as a result of learning from insufficient data sources, those with a large dataset that is rich in features will enable the machine to learn the variety of the feature in order to filter known and unknown malware.

DOI: 10.56892/bima.v7i01.405

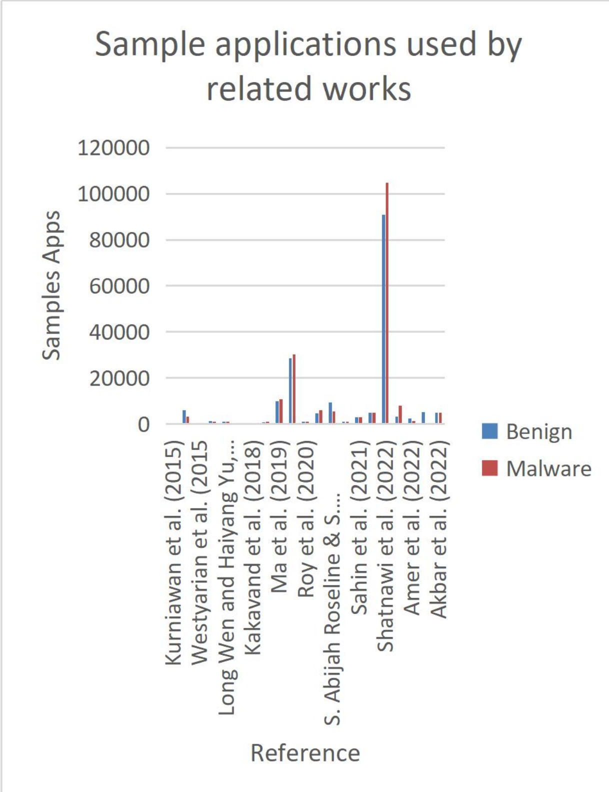

Figure 1: The weight of the data source in the related works

Summary of the results of related works

The results of the related works in this study were tabulated in tables 4 and 5 to show the efficacy of each work in Android malware detection. The tables contain the reference id of the research work and the algorithm used in the study with their metric systems. However, some of this research uses 10-fold cross validation in evaluating the performance of an algorithm, so the metric with the highest level of scores is used in this survey work.

As illustrated in Table 4, it clearly shows the performance of each related work algorithm in Android malware infiltration. These metrics (accuracy, precision, recall, and F-score) prove the effectiveness of each algorithm in Android malware detection, and by comparing the results of this related work, we can quickly conclude that this algorithm outperforms the others and those that need improvement are seen due to their poor performance in accuracy, precision, recall, and F-score.

Research gap identified from the related works

This section summarizes a number of key findings based on the works surveyed in previous Sections. As shown in table 6, the reference ids of each work, the strength of the work, problems associated with the work (if any), and future work are all shown in detail.

Bima Journal of Science and Technology, Vol. 7 (1) Mar, 2023 ISSN: 2536-604

DOI: 10.56892/bima.v7i01.405

Table 3: A synopsis of the data sources used in the captioned research thesis. (x= not available)

S/no	Ref	Features	Analysis Types	Feature selection Technique	Dataset type	Dataset year
1	Kurniawan et al. (2015)	Internet traffic, battery usage battery temperature	Dynamic	x	200 malwares from Android Malware Gnome Project. 200 benigns from google play store	2015
2	Hsin-Yu Chuang & Sheng-De Wang, (2015).	API calls	Static	x	6005 benign apps and 3368 malware apps	x
3	Westyarian et al. (2015	API calls	Static	x	205 benign apps and 207 malware apps	x
4	Wu et al. (2016)	API calls	Static	x	1,160 benign and 1,050 malicious samples	x
5	Long Wen and Haiyang Yu, (2017).	permission, intent, CPU consumption, battery consumption, number of running processes, number of short message and API calls	Static and Dynamic	PCA-RELIEF	1000 benign Apps from google play store, 1000 malware Apps from Drebin Project and Android Malware Genome Project	x

DOI: 10.56892/bima.v7i01.405

6	Milosevic et al. (2017)	permission and source code	Static and Dynamic		M0Droid with 200 benign apps and 200 malware apps	2015
7	Kakavand et al. (2018)	permission and system call logs	Static and Dynamic		M0Droid with 200 benign apps and 200 malware apps	2015
8	Oktay Yildiz & Ibrahim Alper Doğru, (2019)	permission	static	GA	Genome Project (AMGP) with 1119 malwares and 621 benign from google play store	2012
9	Ma et al. (2019)	API calls	static	control flow graph	AndroZoo with 10010 benign and 10683 malwares from VirusShare, Google Play and third party security companies	2010 to 2016
10	Han et al. (2020)	API calls	static	noise filter	28,489 benign from Google Play, Amazon AppStore and APKPure. 30,113. malwares from AMD and Drebin witth	2016 to 2017

DOI: 10.56892/bima.v7i01.405

11	Roy et al. (2020)	API calls	Static	Non-negative Matrix Factorization	133,227 attributes 1100 malware from DREBIN 1100 benign from "CICInvesAndMal2019	2010 to 2012 2014
12	McDonald et al. (2021)	permission	static		4597 benign apps from the Google Play store 6000 malicious apps from the AndroZoo	2017 2018
13	S. Abijah Roseline & S. Geetha, (2021).	Permission and API calls	static	LOFO	Drebin: 5560 malware apps, 9476 cleanware apps and each sample has 215 features	x
14	Sahin et al. (2021)	permission	static	linear regression	1000 malware apps randomly selected from the Android Malware Dataset and 1000 benign apps are downloaded from APKPure	x
15	Sahin et al. (2021)	Permision	static	filter- based	3000 malicious apps from APKPure (APKPure	2020

DOI: 10.56892/bima.v7i01.405

					2020) and 3000 benign apps from VirusShare dataset (Dataset 2020).	
16	Arif et al. (2021)	permission	static	Particle swarm optimisation (PSO), information gain, and evolutionary computation.	5,000 benign apps from Androzoo and 5,000 malware apps from Drebin	2014
17	Kumar et al. (2022)	Permission and API call signature	static	GA	3799 Android apps from Google play store	
18	Shatnawi et al. (2022)	Permission, API call, intent, DNS, IP address, port address and action repeat	static, dynamic and hybrid	RF Algorithm	104747 malware applications and 90876 benign applications from Palo Alto Networks	2017
19	Urooj et al. (2022)	API calls, permissions, intents, packages, receivers and services	static		1795 benign and 10,516 malwares from MalDroid 1500 benign and 3062 malwares from	2020 2021 2021

Bima Journal of Science and Technology, Vol. 7 (1) Mar, 2023 ISSN: 2536-604
DOI: 10.56892/bima.v7i01.405

					DefenseDroid 2421 benign and 5000 malwares from GD Malgenome dataset which contains 1260 malware apps and 2539 benign apps	
20	Amer et al. (2022)	API and permission	static	x	Maldroid dataset has 11599 categorised as benign, adware, banking malware, and mobile riskware	x
21	Shatnawia et al. (2022)	API calls and Permission	static	Recursive Feature Elimination	426 malware apps and 5,065 benign apps from CIC InvesAndMal2019	2019
22	Mohamed Salem Alhebsi. (2022)	permissions and signatures	static	Frequency Counts, correlations and Chi-Square Test	permissions data is obtained from Mahindru, Arvind (2018),Mendeley Data, V5	x

Bima Journal of Science and Technology, Vol. 7 (1) Mar, 2023 ISSN: 2536-604
DOI: 10.56892/bima.v7i01.405

23	Akbar et al. (2022)	Permission	static	Random Forest-based feature importance	signatures data is obtained from Malgenome 5,000 malware apps from VirusShare 5,000 benign apps from google play store	2021

Table 4. Performance results of related work for Android malware detection (x=not available)

S/no	Ref	Algorithm	Accuracy %	Precision %	Recall %	F-Score %
1	Kurniawan et al. (2015)	RF SVM LMT	86 84 85	x	x	x
2	Hsin-Yu Chuang & Sheng-De Wang, (2015).	SVM	96.69	95.53	95.25	x
3	Westyarian et al. (2015	SVM RF	x	92.40 91.40	x	x
4	Wu et al. (2016)	KNN	97.66	x	x	x
5	Long Wen and Haiyang Yu, (2017).	SVM	95.2	x	x	x
6	Milosevic et al. (2017)	SVM	95.1	x	x	x
7	Kakavand et al. (2018)	SVM KNN	79.08 67.00	x	x	x
8	Oktay Yildiz & Ibrahim Alper Doğru, (2019)	NB DT SVM	95.69 97.24 98.45	x	x	94.90 96.70 98.10
9	Ma et al. (2019)	API usage ApI frequency API sequence	x	96.81 97.70 98.45	96.17 97.11 98.79	96.49 97.40 98.62

10	Han et al. (2020)	Linear SVM	99.75	99.54	99.97	x
		Logistic	91.86	94.77	84.72	89.47
		Regression	93.35	90.88	93.06	91.95
	Roy et al. (2020)	SVM	93.77	99.80	84.72	91.73
11		RF	93.15	89.37	94.44	91.84
		KNN				
		RF		82.49	85.85	841.88
		SVM		80.54	85.45	82.92
12	McDonald et al. (2021)	GNB	81.53	60.28	60.28	74.86
		KM	79.87	80.54	85.45	74.86
		DT		x	x	x
	S. Abijah Roseline & S.	RF	93.86	94.91	92.83	93.86
13	Geetha, (2021).	XGBoost	95.57 95.59	94.85	92.95	93.89
14	Sahin et al. (2021)	Multi-Layer Perceptron (MLP)	x	x	x	96.1
15	Sahin et al. (2021)	RF	x	x	x	95.2
16	Arif et al. (2021)	RF	91.59	91.6	91.6	91.6
		KNN	91.56	91.6	91.6	91.6
17	Kumar et al. (2022)	SVM	93.00			
		NN	98.02			
		Gradient boosting	99.51	99.70	99.49	99.58
		XGBoost	99.55	99.73	99.53	99.61
18		DT	99.25	99.43	99.38	99.25
	Shatnawi et al. (2022)	RF	96.03	96.93	96.73	96.80
		AdaBoost	96.24			
		DT	90.12			
		SVM	92.00			
		KNN	89.45			
19	Urooj et al. (2022)	NB	88.65			
		RF	89.00	x	x	x
		KNN	99.30	98.20	100	100
		NB	97.40	99.20	100	100
20	Amer et al. (2022)	SVM	100	100	100	100
		DT	100	100	100	100
		SVM	94.36	95.9	82.6	88.8
21	Shatnawia et al. (2022)	KNN	93.42	91.1	83.7	87.3
		NB	84.33	97.4	60.00	70.8
		Permission:RF		97.34	78.5	86.91
22	Mohamed Salem Alhebsi. (2022)	Signature - based: KNN		97.23	94.38	95.78
23		SVM				
		RF	89.7	89.77		89.69
		Rotation	89.96	89.97		89.95
		Forest	86.25	86.25		86.23
	Akbar et al. (2022)	NB	89.52	89.53		89.50

DOI: 10.56892/bima.v7i01.405

Table 5. Errors in related work for Android malware detection (x=not available)

S/no	Ref	Algorithm	TP %	FP %	TN %	FN %
1	Kurniawan et al. (2015)	RF SVM LMT	86 79 85	14 20 15	x	x
2	Hsin-Yu Chuang & Sheng-De Wang, (2015).	SVM	x	2.5	x	x
3	Westyarian et al. (2015	SVM RF	x	x	x	x
4	Wu et al. (2016)	KNN	x	x	x	x
5	Long Wen and Haiyang Yu, (2017).	SVM	94.7	13.3	x	x
6	Milosevic et al. (2017)	SVM	x	x	x	x
7	Kakavand et al. (2018)	SVM KNN	80.50 80.00	x	x	x
8	Oktay Yildiz & Ibrahim Alper Doğru, (2019)	NB DT SVM	97.00 98.00 98.90	x	93.40 95.80 97.50	x
9	Ma et al. (2019)	API usage ApI frequency API sequence	x	3.36 2.43 1.65	x	x
10	Han et al. (2020)	Linear SVM	x	x	x	x
11	Roy et al. (2020)	Logistic Regression SVM RF KNN	x	x	x	x
12	McDonald et al. (2021)	RF SVM GNB KM DT	x	x	x	x
13	S. Abijah Roseline & S. Geetha, (2021).	RF XGBoost	x	x	x	x
14	Sahin et al. (2021)	Multi-Layer Perceptron	x	x	x	x

		(MLP)				
15	Sahin et al. (2021)	RF	x	x	x	x
16	Arif et al. (2021)	RF KNN	91.6	x	x	x
17	Kumar et al. (2022)	SVM NN Gradient boosting	x	x	x	x
18	Shatnawi et al. (2022)	XGBoost DT RF AdaBoost	x	x	x	x
19	Urooj et al. (2022)	DT SVM KNN NB RF	x	0.30 0.73	x	x
20	Amer et al. (2022)	KNN NB SVM DT	x	x	x	x
21	Shatnawia et al. (2022)	SVM KNN NB	x	x	x	x
22	Mohamed Salem Alhebsi. (2022)	Permission:RF Signature - based: KNN	x	x	x	x
23	Akbar et al. (2022)	SVM RF Rotation Forest NB	89.70 89.96 86.24 89.52	x	x	x

Table 6. Key findings from related work on android malware detection (x=not seen)

S/no	Ref	Strength	Weaknesses	Future Work
1	Kurniawan et al. (2015)	A combination of three features: network data, battery consumption, and battery temperature, with Support Vector Machine as an algorithm, seems promising in detecting malware.	The apps have to run for some period of time before the model can detect malware. Thus, they pave the way for the theft or loss of information	Build a model that first uses static analysis to counter attack malware and, thereafter, explores these features (network data, battery consumption, battery temperature) to detect malware as a result of failed static analysis.
2	Hsin-Yu Chuang & Sheng-De Wang,	It has a high accuracy in the prediction of	x	x

	(2015).	malware with a low false positive rate		
3	Westyarian et al. (2015	It has a low rate of false positives.	Small samples were used for the experiment; therefore, malware could evade the algorithm	A large number of samples could be used to improve the existing algorithm with more features
4	Wu et al. (2016)	It has a low rate of false positives.	x	Use of semantic-based features such as data dependency graphs and control flow graphs to classify Android malware. Semantic-based approaches can profile malicious behaviors in detail, which is useful in studying the countermeasures for malware variants and unknown malware
5	Long Wen and Haiyang Yu, (2017).	Robotics in detecting malware	It has incurred a large number of computational overheads and consumes a lot of time	x
6	Milosevic et al. (2017)	x	Dataset is small and obsolete	Using a significantly larger balanced dataset, utilizing online learning, and Another research focus is combining static and dynamic software analysis, in which multiple machine learning classifiers are applied to analyze both source code and dynamic features of apps in run-time.
7	Kakavand et al. (2018)	It recorded a high true positive rate using KNN, i.e., KNN yielded a promising result	It incurs a large number of computational overheads and is time-consuming to analyze and detect	To expand their methodology by considering two categories of dynamic and hybrid malware analysis and compare the

Bima Journal of Science and Technology, Vol. 7 (1) Mar, 2023 ISSN: 2536-604
DOI: 10.56892/bima.v7i01.405

			anomalies.	results with our findings in this research (larger Dataset)
8	Oktay Yildiz & Ibrahim Alper Doğru, (2019)	This method reduces the number of redundant permissions, which will reduce the analysis time and improve detection efficiency.	AMGP is an old dataset, so the lower API level was used to include more permission	The study can be implemented for higher API level permissions with a newer dataset
9	Ma et al. (2019)	This method reduces the number of redundant APIs in order to reduce the analysis time and improve detection efficiency, as requested by application samples in the dataset	x	Use the newer dataset and build a multi-class classification model to determine which malicious family the application belongs to if detected malicious
10	Han et al. (2020)	It has high accuracy in malware detection	x	focus on the dynamic analysis of Android applications API calls
11	Roy et al. (2020)	Non-negative matrix factorization improved the efficiency of malware detection	Because the dataset contains a small number of sample apps, malware may be able to circumvent the model.	use a newer dataset with many features
12	McDonald et al. (2021)	x	x	Explore more static features associated with APKs that are elaborated in the current literature that could be easily combined with a manifest file approach to form a greater feature set and build a model that is more finely tuned towards the detection of malware
13	S. Abijah Roseline & S. Geetha, (2021).	This remove redundant features in order to reduce the analysis time.	x	x
14	Sahin et al. (2021)	It reduces the feature vector dimension, and	It uses fewer static permissions	Explore different feature selection methods in this

		the training time is decreased		field to increase the classification performance
15	Sahin et al. (2021)	There is improved classification performance, and execution time has decreased	Its emphasis is only on permission requested	Apart from the greedy metric combination approach, different search strategies can be developed and the classification performance will increase.
16	Arif et al. (2021)	It lowers the running time of the classifiers as per not all features were selected for analysis to distinguish malware from benign	It focused only on permission-based features	Future studies should extend the model by identifying more malware behaviours and extracting other features, such as API calls and code analysis
17	Kumar et al. (2022)	It lowers the classifier's complexity	It obtained higher accuracy than filtering features with a genetic algorithm because the dataset was not enormous	Future work can take advantage of larger datasets for better results and an examination of the impact on other machine learning techniques
18	Shatnawi et al. (2022)	It's demonstrated that a combination of the permissions and action repetition features has achieved good results and can take advantage of larger datasets for better results, as well as an examination of the impact on other machine learning techniques.	It is expensive and time-consuming to classify and analyze malware .	To expand the investigation and move beyond binary classification to create a brand-new approach based on machine learning to categorize malware families.
19	Urooj et al. (2022)	It has immunized the false positive rate in malware detection	time-consuming in analyzing malware by using an enormous dataset.	Consider model resilience in terms of enhanced and dynamic features
20	Amer et al. (2022)	Running time for the classifiers are minimized	The approach requires a continual update in terms of training because the features that was use in training the	A more in-depth investigation of feature selection techniques in future work with a larger dataset

			model is not enormous to detect malware variants or adversarial attacks can identify the training patterns and evade or trick the model	
21	Shatnawia et al. (2022)	less complexity in running time		A more in-depth investigation of feature selection in future work with a larger dataset
22	Mohamed Salem Alhebsi. (2022)	Signature-based authentication gives a better performance and accuracy in detecting malware with fewer false positive	The model can be evaded by code obfuscation, method renaming, and string encryption techniques because fewer features are selected for permission, and the signatory base can only identify existing malware and fails against unseen variants.	Explore alternative approaches in addition to permissions and signatures described in the research work
23	Akbar et al. (2022)	It lowers the running time of the classifiers as per not all features were selected for analysis to distinguish malware from benign	The permission feature is not sufficient to detect malware because malware vendors will only study the targeted permission and come up with robotic malware to evade the model	The performance can be improved by heightening dataset (an enormous dataset) and incorporate API calls into the dataset to improve selected features in order to increase the performance of the classifiers or model

DOI: 10.56892/bima.v7i01.405

CONCLUSION

This study summarizes future research work in Android malware detection using machine learning algorithms. The most recent Android datasets were summarized with their reference ids, which could help researchers easily pick up the dataset to use for research work. However, the unavailability of a larger Android malware dataset remains a great problem in evaluating the efficacy of research work in Android malware detection using machine learning. When rich datasets are properly shared among researchers, this could potentially lead to a solid counter to Android malware vendors. This paper gives a better understanding and explores the fact that future work can easily be embarked upon in detecting both known and unknown Android malware with machine learning.

This paper identified various directions in which Android malware detection can be done using machine learning by summarizing the existing research endeavors, the model problem, and future progress. Android malware is expanding, so it's important to solve this issue by developing more efficient Android malware detection systems that can not only increase the precision of identifying existing malware but also reveal zero-day malware attacks.

REFERENCES

Aafer, Y. W. Du and Yin. H. (2013). "DroidAPIMiner: Mining API-Level Features for Robust Malware Detection in Android," in SECURECOMM, 2013.

Abijah, S. R. and Geetha, S. (2021). Android Malware Detection and Classification using LOFO Feature Selection and Tree-based Models. Journal of Physics: Conference Series 1911(2021) 012031.

Accuracy, Precision, Recall & F1 Score: Interpretation of Performance Measures [Online]. Available: https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/

Ahmed, S. S. Aya J. Tuqa, B. Y. Eyad, T. Mahmoud, A. and Dheya, M.(2022). An Android Malware Detection Leveraging Machine Learning. Hindawi Wireless Communications and Mobile Computing, Volume 2022, Article ID 1830201.

Ahmed, S. S. Qussai, Y. and Abdulrahman, Y. (2022). An Android Malware Detection Approach Based on Static Feature Analysis Using Machine Learning Algorithm. The 3rd International Workshop on Data-Driven Security (DDSW 2022), March 22 - 25, 2022, Porto, Portugal.

Alejandro, G-M, Hayretdin B. and Sven N. (2021). KronoDroid: Time-based Hybrid-featured Dataset for Effective Android Malware Detection and Characterization. Published by Elsevier Ltd. [Online]. Available: https://github.com/aleguma/kronodrod

Android Adware and General Malware Dataset (CIC-AAGM2017). [Online] Available: https://www.unb.ca/cic/datasets/android-adware.html

Android Malware Static Analysis (CCCS-CIC-AndMal-2020) [Online]. Available:https://www.unb.ca/cic/datasets/andmal2020.html

Android Malware Dataset – Kaggle, [Online]. Available:

DOI: 10.56892/bima.v7i01.405

https://www.kaggle.com/datasets/shashwatwork/

Android PRAGuard Dataset [Online]. Available:http://pralab.diee.unica.it/en/AndroidPRAGuardDataset

Android validation dataset. [Online]. Available:https://www.unb.ca/cic/datasets/android-validation.html

AndroZoo [Online]. Available: https://androzoo.uni.lu/

Arindaam R. Divjeet S. J., Gitanjali J. and Kapil S. (2020). "Android Malware Detection based on Vulnerable Feature Aggregation". International Conference on Smart Sustainable Intelligent Computing and Applications under ICITETM2020

Arnab C. Facts about Android Operating System. [online] available: https://www.tutorialspoint.com/facts-about-android-operating-system

Beenish, U. Munam, A. S. Carsten, M. Muhammad, K.A. and Sidra, R. (2022). Malware Detection: A Framework for Reverse Engineered Android Applications through Machine Learning Algorithms".

CICMalDroid (2020). [Online]. Available: https://www.unb.ca/cic/datasets/maldroid-2020.html

ContagioDump. [Online]. Available: http://contagiodump.blogspot.com/2013/03/16800-clean-and-11960-malicious-files.html

ContagioDump. [Online]. Available: http://contagiominidump.blogspot.tw/

Daniel, A. Michael S. Malte, H. Hugo G. and Konrad R. (2014). DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket. ISBN 1-891562-35-5

Durmus, O. S. Oguz, E. K. Sedat, A. and Erdal, K. (2021). A novel permission-based Android malware detection system using feature selection based on linear regression.

Durmus O. Sahin, O. K. Sedat, A. and Erdal, K. (2021). "A novel Android malware detection system: adaption of filter-based feature selection methods". Journal of Ambient Intelligence and Humanized Computing.

ElMouatez, B. K. Mourad D. Abdelouahid D. and Djedjiga M. (2018). MalDozer: Automatic framework for android malware detection using deep learning. Published by Elsevier Ltd on behalf of DFRWS. [Online]. Available:

Eslam, A. Seif, E.M. Mostafa, A. Amr, E. Omar, S. Haytham, M. and Ammar, M. (2022). 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

Fahad, A. Mehdi, H. Rafia, M. Qaiser, R. Ainuddin W. A. W. and Ki-Hyun J. (2022). Detection of Android Malware Using Machine Learning. Symmetry 2022,14,718.

Genome Project [Online]. Available: http://www.malgenomeproject.org/

Hsin-Yu C. and Sheng-De W. (2015). Machine learning based hybrid behavior models for Android malware analysis. 2015 IEEE International Conference on Software Quality, Reliability and Security.

Hyoil, H. SeungJin, L. and Kyoungwon S. (2020). Enhanced Android Malware Detection: An SVM-based Machine Learning Approach.IEEE International Conference on Big Data and Smart Computing (BigComp) ©2020 IEEE.

Investigation of the Android Malware (CICInvesAndMal2019), [Online]. Available:*https://www.unb.ca/cic/datasets/invesandmal2019.html*

DOI: 10.56892/bima.v7i01.405

ISCX Android Botnet dataset (2015) [Online]. Available:https://www.unb.ca/cic/datasets/android-botnet.html

Juliza, M. Arif. Mohd, F. A. Suryanti, A, Sharfah, R.T.M, Nor S. N. I and Ahmad, F. (2022). A static analysis approach for Android permission-based malware detection systems. PLoS ONE 16(9): e0257968.

Kaijun, L. Shengwei, X. Guoai X. Miao, Z. Dawei S. & Haifeng, L. (2020). A Review of Android Malware Detection Approaches based on Machine Learning.

Kanal, S. Sajan (15 March, 2022). Effects of Android malware [online] available: https://www.tutorialspoint.com/what-is-android-malware

Kurniawan, H. Yusep R. and Budiman D. (2015). Android Anomaly Detection System Using Machine Learning Classification. The 5th International Conference on Electrical Engineering and Informatics 2015. August 10-11, 2015, Bali, Indonesia

Li Y, Jang J and Hu X. (2017). Android malware clustering through malicious payload mining[C]//International Symposium on Research in Attacks, Intrusions, and Defenses. Springer, Cham, 2017: 192-214.

Long W. and Haiyang Y. (2017). "An Android malware detection system based on machine learning". AIP Conference Proceedings 1864, 020136 (2017).

Machine Learning Glossary.[Online]. Available:https://ml-cheatsheet.readthedocs.io/en/latest/#machine-learning-glossary.

Mohamed. S. A. (2022). Android Malware Detection using Machine Learning Techniques.Thesis. Rochester Institute of Technology, RIT Dubai. April 27, 2022.

Milosevic, N. Ali D. and Kim-Kwang R. C. (2017). Machine learning aided Android malware classification.

Mohsen, K. Mohammad D. and Ali D. (2018). Application of Machine Learning Algorithms for Android Malware Detection. ACM ISBN 978-1-4503-6595-6/18/11

M0Droid[Online].Available:https://www.azsecure-data.org/other-data.html

Nicolas, K. Jean-François, L. Mourad, L. & Valérie, V. Triem T. (2016). Kharon dataset: Android malware under a microscope. The Learning from Authoritative Security Experiment Results (LASER) workshop, May 2016, San Jose, United States. pp.1-12. hal-01311917.

Oktay Y. and Ibrahim A. D. (2019). Permission-based Android Malware Detection System Using Feature Selection with Genetic Algorithm. International Journal of Software Engineering and Knowledge Engineering. Vol. 29, No. 2 (2019) 245–262

Petroc. T, Market share of mobile operating systems worldwide 2009-2022, [Online]available:https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/ access: Feb 21, 2023

Songyang, W. Pan, W. Xun L. and Yong, Z. (2016.) Propose Effective Detection of Android Malware Based on the Usage of Data Flow APIs and Machine Learning.

Sumath, N. Kumar, P. Vishnu V. P. Akhil C. and Spandan K.S.V (2022). Analysis of Malware in Android Features Using Machine Learning. Journal of engineering sciences. Vol 13, Issue 01, Jan / 2022 ISSN NO: 0377-9254

Todd, J. M. Nathan, H. William, B. G. and Ryan, K. B. (2021). Machine Learning-Based Android Malware Detection Using Manifest Permissions. Proceedings of the 54th Hawaii International Conference on System Sciences | 2021

Van der Meulen, R., and Rivera, J. (2013). Gartner Says Smartphone Sales Accounted for 55 Percent of Overall Mobile Phone Sales in Third Quarter of 2013. URL: http://www. gartner. com/newsroom/id/2623415 (7.04. 2019).

Westyarian, Y. R. and Budiman Dabarsyah. (2015). “Malware Detection on Android Smartphones using API Class and Machine Learning”. The 5th International Conference on Electrical Engineering and Informatics 2015. August 10-11, 2015, Bali, Indonesia.

Zhou Z. H. (2016), Machine Learning. Beijing, China: Tsinghua University Press (in Chinese), 2016.

Zhuo M. Haoran G. Yang L. Meng Z. and Jianfeng M. (2019). A Combination Method for Android Malware Detection Based on Control Flow Graphs and Machine Learning Algorithms. Vol 7(2019). 2169-3536©2019 IEEE. For more information.http://www.ieee.org/publications_standards/publications/rights/index.html.