A Study on Android Malware Detection Using Machine Learning Algorithms

K. S. Ujjwal Reddy, S. Sibi Chakkaravarthy, M. Gopinath(✉), and Aditya Mitra

School of Computer Science and Engineering, VIT-AP University, Amaravati, India
{ujjwal.20bci7203,chakkaravarthy.sibi,gopinath.19phd7021,
aditya.20bcr7009}@vitap.ac.in

Abstract. Today, Android has become the most popular operating system because of its salient features. As it is an open-source mobile OS, several developers are developing and publishing their android applications. On the other side, attackers are manipulating those applications in the form of malicious software (Malware) by leveraging the application or functional flow of android OS and those malwares create loss or leakage of confidential sensitive information. Though most anti-virus software affords defence against malware attacks, still the attacks are highly possible in the real time adversarial environment. In this paper, the machine learning-based detection method is designed by combining the features of application namely permission and activity which are obtained during the installation of apps. In our design, permissions and activities of each app are extracted making use of Androguard tool. Using this feature combination, malicious apps are classified as either benign or malicious. The advantage of this method is that there is no need for any dynamic analysis. In our experimentation, we used real-world app samples with 500 malware and 500 benign to train the algorithm for better performance. Based on the experimentation results, highest detection rate is attained by Random Forest (RF) with 95% of accuracy and lowest detection rate is obtained by K-Nearest Neighbors (KNN) with 79% of accuracy.

Keywords: Android malware · Malware detection · Permissions · Activities · Smartphone protection · Machine learning

1 Introduction

Mobile devices are typically handheld devices like portable telephones which are used for receiving calls over radiofrequency. Based on the respective purposes, various mobile devices are used such as smart phones, laptops, tablets, and smart watches. The economy of a smart phone is higher than personal computers (PC) [1]. Most malware developers have a keen interest in aiming at mobile devices rather than PCs because of the stored confidential information related to users [2, 3].

Android phones hold more than 85% of mobile phone sales whereas iOS phones come after them with 14% of mobile sales. Android phones and iOS phones are in the top two positions of mobile sales globally. In 2019, it is found that android is the most

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2023
Published by Springer Nature Switzerland AG 2023. All Rights Reserved
S. Nandan Mohanty et al. (Eds.): ICISML 2022, LNICST 470, pp. 232–247, 2023.
https://doi.org/10.1007/978-3-031-35078-8_20

insecure OS [4]. There is an enormous amount of growth in the development of android malware over the last four years. In the year 2016, android malware reported its highest explosion, and consecutive increased amount of security incidents lead android malware to the first place as a risky OS with maximum vulnerabilities. User data and devices are affected by various types of Potential Harmful Applications (PHAs) or malwares like Trojans, spyware apps, backdoor, stalkerware (commercial spyware), Denial of Service, hostile downloader, phishing, rooting, ransomware, spyware, spam and Mobile Unwanted Software (MUwS). In addition mobile billing fraud activities are also implemented through call, SMS and toll where users are automatically charged in deceptive manner.

Android mobile operating system is Linux kernel based, and it is built as open-source software where every developer has a great advantage of applying logical implementation. As android is the leading smart phone operating system based on the recent reports, every day more than 5000 android applications (APK) are developed and published in Google play store service [5]. On the other hand, malicious developers are also using the apps and trying to manipulate them in order to convert as malicious software. Malicious software is defined as malware that tries to collapse the system and helps in the loss or leakage of confidential information. Though Google Play Store Service has a built-insecurity system, malware developers are trying to find a new way to get pass the security system of Google Play Store Service.

Many researchers are focusing on finding a new way of android Malware detection and classification. As of now, the most traditional way of classifying android malware is based on the signature (md5) of the App. In signature-based method, they extract signatures used by the developers in the app and they try to see and find the patterns of the signature to classify whether the app is malicious or not. But, with the help of advanced obfuscation techniques, malware developers can easily pass through signature-based detection mechanisms nowadays. In general, static analysis and dynamic analysis are two primary ways of malware detection used by researchers.

Static analysis is the process of classifying software as malware and benign without real time execution in the system. The tools which are used in the static analysis are APK Inspector, Dex2Jar, etc. In this method, the app is first unpacked using Androguard. Then, extraction of all the permissions, components, and API (Application Programming Interface) calls which are connected in manifest files and Smali files are initialized, and data is used to classify benign and malware. Dynamic analysis is another kind of analysis technique that is performed by executing software to find if it is malware or benign. In this research work, the app is executed in a Sandbox (which is a real-time testing environment for detecting malicious apps) to find whether the app has malware by checking its behavioural features during execution. There are many sandboxes in use like Virus total, Droid Box, Cuckoo Sandbox, etc. It is seen that dynamic analysis is one of the best methods of detecting malicious software. Shortcomings of static and dynamic analysis are overcome by hybrid analysis in which signature specifications of malware are analysed first and then behavioural parameters are further analysed for effective complete malware analysis.

In this work, machine learning models are utilized for finding the accuracy of classification in apps amid malware and benign. First, a dataset is built with the values of 0

and 1 to classify every app making use of available features like permission and activity. We have taken all the permissions and activities identified while installing app and those features are kept in a text file where all the names of permissions and activities are added as a column name. In the dataset, there is a specific column named MALWARE which classifies between malware and benign. After the creation of the dataset, we used all the familiar models of machine learning to classify the app.

Machine learning is the study of computer algorithms that allows automated improvement with the help of data usage and experience. Machine learning algorithm models-built data which is used for training purpose and the remaining data will be used for testing the model. The ML models are primarily categorized in three distinct types which are supervised learning, unsupervised learning, and reinforcement learning. Many ML algorithms are in use, but the most commonly used machine learning models are Linear Regression, Logistic Regression, Decision Tree (DT), SVM (Support Vector Machine), Naive Bayes (NB), KNN (k-Nearest Neighbors), Random Forest (RF), Gradient Boosting Algorithms, etc.

The supervised learning algorithm is one kind of machine learning technique that uses dataset in labeled manner. It is the most familiar method used to train the model and gives us the perfect accuracy by using the labels present in the dataset. Supervised learning is mainly of two parts: Classification and Regression. Unsupervised learning is just the opposite of supervised learning, where it uses data to analyze and cluster the unlabelled dataset. Unsupervised learning is mainly of three parts: Clustering, Association, and Reduction.

Linear Regression is an algorithm where it trains the model in such a way it estimates the real value based on its continuous variables. Here, it is possible to establish the association amid the independent and dependent variables by fitting the finest line in the graph. There are two types of regression which are simple linear regression and multiple linear regression. In simple linear regression, characterization is performed by one independent variable and for Multiple Linear Regression more than one variable are utilized for characterization. Let's consider the following example for understanding linear regression. Assuming one 4th-grade student is asked to arrange his classmates based on weight, without asking their weights. Then the student tries to visualize the students and tries to arrange them based on their height and the build of their body. This is an example of linear regression in the real life.

Logistic regression is an algorithm that trains the model with data having values such as 0/1, true/false, and yes/no making use of provided independent variables set. Actually it tries to train the model in such a way it estimates the event occurrence probability. To understand this in one simple example, let's give a trigonometric or linear algebraic sum to solve to a tenth-grade student and a fifth-grade student. By this, it is understandable that the probability of solving the sum by a tenth-grade student is 80% and the probability of solving the sum by fifth grade is 20%. In such a way logistic regression predicts the outcome.

DT (Decision Tree) algorithm is a kind of supervised learning algorithm which has the most efficient way of classifying any dataset. This algorithm works on both categorical and continuous dependent variables. While training this model, it tries to divide the dataset as two or multiple sets in homogeneous manner. Decision Tree splits the

dataset into different homogeneous sets by using various techniques such as Information Gain, Chi-Square, and Entropy. When we try to buy a product from an online store, we get lots of recommendations based on what we are trying to buy, or else we will get ads on the recently viewed product, this is all done through the classification algorithm based on our search history. The decision tree algorithm works in such a method.

KNN (k-Nearest Neighbors) is mainly utilized for classification as well as regression. But this algorithm is mainly applied for categorization in industries. It works in simple way where all available are stored and new cases are classified using the mainstream votes of its k-neighbors. This algorithm uses a distance function to classify the cases from K-Nearest Neighbors. Distance functions such as Euclidean, Manhattan, Minkowski, and Hamming distances are used in this algorithm. The best to understand the KNN algorithm is by mapping it to our real lives, when we have a situation to learn about a person without any information then we will try to ask his close friends for the information. This is how the KNN (K-Nearest Neighbors) works.

The Random Forest algorithm has a similar function to the KNN. It is a collaborative approach to regression and classification. To classify an object based on attributes, each and every tree contributes towards voting process in order to perform classification. To understand this algorithm with an example, the Random Forest algorithm used in our emails, where this classification checks if mail is 'not spam' or 'spam' and classifies them appropriately. It is used in many fields like industries, banking, retail, healthcare, etc. because it classifies a huge data in a few seconds.

Gradient Boosting is a powerful algorithm for handling a huge amount of data. It uses multiple weak and average predictions to build a strong prediction on the data. This algorithm comes more useful in the data science field which uses the set of decision trees for the prediction like Random Forest, but it is more powerful than Random Forest. Gradient Boosting algorithms have several types of classifiers: Gradient Boosting Machine (GBM), Extreme Gradient Boosting (XGBoost), Light Gradient Boosting Machine, and CATBoost. For understanding the algorithm by an example, making use of annual data of house rent cost it is possible to predict the rent after 5 years using this algorithm. This is how this algorithm became familiar for use.

The major contributions of this research work are as follows:

- In this research work, the dataset that contains all of the permissions and actions used by the app during installation is trained to determine the accuracy of all models.
- By relying solely on an app's permissions and activities, it will increase its contribution to discovering the most effective model with more accuracy in detecting malware.
- The resultant model can be used as an add-on for the antivirus software with a simple static analysis without using dynamic analysis.

The remaining sections of this research paper are categorized with the following parts: Sect. 2 gives details about recent previous existing research works regarding android malware detection, Sect. 3 explains the proposed system architecture and working flow, Sect. 4 describes the results and observations obtained during experimentation and finally conclusion and future work are presented.

2 Related Work and Research Study

More malicious app developers concentrated on targeting the users of android by incorporating advanced methods. It insists many security researchers to work on malware detection and analysis area for providing effective protection towards legitimate users and their devices. This section details the recent works related to android malware detection and summary of recent related research works are furnished in Table 1.

DroidMat, technique is proposed for discovering android malwares that first extracts every android application manifest file and obtain information like requested permissions, intent message passing, and others [6]. Further it extracts components related information such as activity, service, and receiver. Those extracted information are considered as primary requirements for permission based API calls trace. This system is trained with the help of a machine learning algorithm, called KNN (k-Nearest Neighbour) for classifying malware and benign APK files. Totally 1,738 Android applications are used for this research. It succeeds by achieving 97.87% of accuracy. Behaviour-based detection is also used which adopts dynamic analysis with the help of Sandbox or using emulation and environment deployment cost is saved.

Ravi et al. have used diverse types of ML algorithms such as Naive Bayes, Multiclass classifier, Random Forest, J48, and Multilayer Perceptron for detecting android malwares and evaluated each and every algorithm's performance [7]. Utilized framework makes them easier to classify malware and benign. For confirming the system, they have used 3,258 samples of android applications and extracted their features to train the model to find the accuracy and time taken by the model. They were successful in getting good accuracy with the help of Naive Bayes algorithm.

Taniya and Rishabh primarily used dynamic analysis for dataset creation and the model is trained making use of the dataset after that it is evaluated to get the accuracy of the classification [8]. 50 Benign and 50 malicious software in the dataset are used and syscall-capture is created for creating the dataset automatically. ML algorithms are employed for finding the accuracy and it is successfully achieved by getting accuracy of 85% and 88% for DT and RF algorithms respectively.

SpyDroid android malware detection framework is proposed which has three main components: controller, monitoring module, and malware detector [9]. Malware detector is contained with decision-maker which provides support for sub-detector of application layer. Four sub-detectors are implemented on the dataset of 4,965 APKs. From the four sub-detectors, recall is obtained in the percentage of 72, 62, 70, and 65 correspondingly. While trying to mix all four sub-detectors, recall is increased to 94%. In the future, it is planned to detect android malware by checking the inter-process communication in between application and its permission request patterns at the time of execution.

Similarly, another android malware detection method RanDroid is proposed by the utilization of permission, APIs and several available key information like dynamic code, native code, reaction code, database and others [10]. It used all these as features for training and the classification model is built making use of a variety of ML methods to automate malware and benign detection. Results show that RanDroid attains 86% detection on features of hazardous API Calls and dangerous permission amalgamations, and it got 95.5% detection on features like hazardous API Calls and dangerous permission with

the available other primary features like dynamic code, cryptographic code, reflection code, database using SVM (Support Vector Machine) machine learning algorithm.

Another ML based android malware detection framework is proposed on the basis of image patterns [11]. It used 600 APK samples totally in which 300 samples belong to malware and 300 samples belong to benign. It only dealt with the creation of grey-scale images in which 183 are malware and 300 are benign. The remaining APK's were not able to generate grey-scale images as they were corrupted or either they didn't have classes .dex file. Further, KNN, RF, and DT are also used. Considering the outcome, RF algorithm provides 84.14% accuracy which was not too great accuracy.

In other similar work, the machine learning algorithm is trained using a dataset of 1,796 APK samples [12]. Dataset is created by extracting the features like activity, permissions, feature, intent, and service receiver from the sample APKs. Then the model is trained to evaluate and detect the difference between the malware and benign. Based on the result, 95.48% accuracy is obtained from the multiclass classifier and 96% accuracy is reached from the binary classifier.

Permission-based android malware detection method is proposed [9]. It mainly used semantic analysis to classify the APK files. Further, it created a user interface website where a user can upload an APK, and then they are detected with the help of permission used by the app. Finally, the percentage of harmfulness is given to the user for differentiating malware and benign. But, in this work, they have a limitation where it can't predict the malware in some cases or give a wrong prediction and they are trying to improve the website to get a more accurate percentage of harmful in the future.

DroidLicious is a solution method proposed for discovering the malicious behaviors of android apps [14]. ML classifier is used for classifying static features which are obtained with the help of state-of-the-art static taint analysis methods. Training in the proposed classifier is performed with a static feature set which has 3269 malware and 3269 benign apps. Because of its high classification accuracy, RF is used.

Making use of n-gram features related to Smali files, machine learning models are developed [15]. ML models like KNN, Naïve Bayes, Logistic regression, SVM, DT, and RF are trained together by incorporating various methods of feature extraction and selection to compare performance. Variance threshold-based feature selection is performed in which features with the smallest variance are removed. Further information-based feature selection is also implemented. TBA and T-SNE methods are used for feature size reduction.

A static feature selection-based android malware detection method is proposed [16]. CICInvesAndMal2019 dataset is used for android malware detection in which a feature set of permission and intent is used. The Principal Component Analysis technique is applied for selecting features. After the training dataset, testing is performed with familiar ML models such as Naïve Bayes, SVM, RF, DT, and KNN. Among them, RF reached the highest accuracy with 96.05%.

A random forest and tree-based classifiers are implemented for detecting malicious Android apps [17]. Dataset of existing malignant applications is taken by computing Help vector machine. Choice tree computation creates an association with dataset preparation.

Table 1. Summary of recent related works

Research Paper	Features	Detection method	Algorithms of classification
Vivekanandam (2021) [20]	Optimized functionality selection, mean and standard deviation	hybrid method	genetic algorithm
Fiky et al. (2020) [19]	DREBIN and MALGENOME, PCA, and IG methods	ML-based + Intelligent	RF classifier,
Han et al. (2020) [18]	API calls	ML based	SVM
Soni et al. (2020) [17]	Help vector machine, Choice tree computation	ML based	Random forest and tree-based classifier
Sangal and Verma (2020) [16]	Static feature selection, CICInvesAndMal2019 dataset	ML based	Naïve Bayes, SVM, RF, DT and KNN
TAHTACI and CANBAY (2020) [15]	n-gram features, TBA and T-SNE methods	ML based	KNN, Naïve Bayes, Logistic regression, SVM, DT and RF
AlSobeihy yet al (2020) [14]	static features	DroidLicious	RF
Rishab (2020) [13]	app permission	Permission based	semantic analysis
Bhagyashri (2019) [12]	activity, permissions, feature, intent, and service receiver	ML-based	multiclass-classifier, binary classifier
Fauzi (2018) [11]	image patterns, GIST descriptor	Image based	KNN, RF and Decision Tree
Koli (2018) [10]	permission, APIs, dynamic code, Reaction code, native code, database	RanDroid	SVM
Shahrear& Mohammad (2018) [9]	runtime information, sub-detectors	SpyDroid	Random Forest
Taniya and Rishabh (2017) [8]	dynamic analysis	syscall-capture system	J48 Decision Tree & RF
Ravi (2017) [7]	accuracy and time	ML based	Naive Bayes, RF, Multiclass classifier, J48, and Multilayer Perceptron
Wu et al. (2012) [6]	permissions, intent message passing, Behavior-based	DroidMat	KNN

With this prepared dataset, it is possible to predict 93.2% of new malicious portable executables. Out of 135 permissions, 22 permissions were only considered which improved runtime performance by 85.6%.

API (Application Program Interface) calls are mainly concentrated, and they are treated as features for android malware detection [18]. Dataset of API calls seems integrally sparse and high dimensional. Static analysis is applied to the dataset. The experimental setup is done with 133,227 API calls of 58,602 android applications. SVM is

used for classification. Based on the results, 99.75% accuracy is reached with a 99.97% compelling recall rate.

A machine learning-based intelligent model is proposed for detecting android malicious apps [19]. It is static analysis based. Android apps related extracted features like permissions, intents, system commands, and API calls are examined. DREBIN and MALGENOME are the two distinct datasets used by the proposed model. By applying multi-level feature selection methods ML classifier is implemented making use of PCA and IG methods together. Based on the experimental results it is seen that the RF classifier reaches higher performance in android malware detection.

Further genetic algorithm based hybrid method is proposed for android malware detection [20]. It focuses on optimizing the problems related to functionality selection. Algorithm is developed and examined by performing changes in the size of population and mutation rate for balancing purpose in order to attain optimum variable rate. Better results are attained in terms of mean and standard deviation.

After performing the detailed literature survey on recent android malware detection techniques it is inferred that new generation malwares are developed massively day by day in advanced manner with new obfuscation techniques. They are hardcoded and they are intended in evading against the security system adopted with device and security providers. Improved security mechanism is essential for combating against such malwares by continuously monitoring their behaviors.

3 Proposed Detection System

Android apps or APK files are mostly developed in JAVA or Kotlin. They are implemented in JAVA Virtual Machine (JVM) environment. Every android app has a jar file that has .apk as the extension. There are many components of distinct types in the Android application. Every part of the application has an entry which allows user to enter into the application. Further, there are four main components in android application which are activities, services, broadcast receivers, and content providers. Each and every component should be declared inside the application's manifest file for utilization purpose.

Majorly, the inter component communication is done with the help of intents and intent filters. The Inter-Component Communication mechanism, or ICC mechanism, allows Android apps to communicate with one another. Android provides the Inter-Component Communication mechanism to reduce developer burden and to promote functionality reuse. ICC allows inter-component collaboration, but it can now be used by new malwares to create perplexing app behaviors and avoid detection by existing malware detection methods. For example, malware with the package name com.jx.theme can install APK files at runtime. It does not request the permission required for the installation of the package android. Permission.INSTALL PACKAGE, and it can also install APK files from the SD card. For such challenges, we proposed the Inter-Component Communication Detector. Messaging objects which are used to request the actions from the other application components are referred as intents. Expressions stated in the manifest file of application which specifies the receiving part of intent type is defined as intent filters. The permissions and activities are used to get access to the system with all the necessary fields to start the application.

Before performing the classification of a model, it is important to have raw data which helps the model to get trained. To get the raw data the following are required which are unpacking and decompiling the application, extracting the features, and encoding. All these jobs of extracting the raw data were made easy with the help of Androguard. Androguard is fully designed with the help of python3 which helps users to play with Android files. The proposed system architecture is portrayed in Fig. 1.

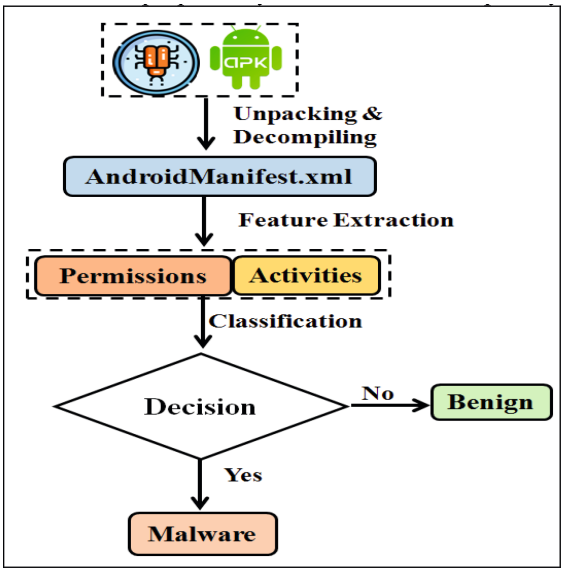

Fig. 1. Malware detection system architecture

While considering weightages given to the features extracted from the dataset, assuming that x features exist in the expected features set fi = {1, 2,…, x}. As the importance values of features differ, each feature is assigned a weight (Wi) on this basis of the nature of feature fi. According, the features vector transforms into the weighted feature vector, which is from the original of the application. We multiply r = each feature by its corresponding weight and the classifier for fine-tuning to find the optional weight.

We made our own dataset and extracted all of the permissions and activities from the APK files. We used approximately 500 malicious and 500 benign files. We extracted all of the features using Amazon AWS servers and created our own dataset file. This file was used to train and classify malware and benign files.

3.1 Unpacking and Decompiling

All the APK files are zip files which are compressed with application source code, manifest file, resources and assets. Dexfiles like Dalvik Executable Files hold encoded format of application source code and interpretation is carried by Dalvik Virtual Machine. Manifest file has several declarations and specifications; it also has image, HTML file,

and others. Since the dex files are basically considered as compiled as well as binary encoded, they are not supposed for interpretation. Due to this, we cannot extract the features directly. Therefore, we need to decompile the dex file to the other format so that it is possible to read and interpret, like Smali code. Smali code is obtained by decompiling the dex file which is in the format of assembly code of the app. After the completion of decompiling APK files further step is carried.

3.2 Feature Extraction

Feature Extraction is considered as the very significant part where feature is used for training a ML model. Based on the utilized features, upper bound of provided model's performance is defined. After studying several research papers, it is decided to use permissions and the activities used by the APK file which are encoded in source code as well as manifest file.

- ***Permissions:*** Android utilizes the permission system where it tries for safeguarding user's privacy. In an app, sensitive data is requested by asking for permissions for CAMERA, LOCATION, CONTACT, etc. All the features used by the app are obtained with help of Androguard and all the features are stored in a text file for the dataset creation.
- ***Activities:*** An activity of an Android application is provided in which the application draws its UI (User Interface). In addition, one or more activities of an app are obtained to know how many interfaces it uses to run the app. In such a way, Androguard is used to get all the activities used by the app and all the activities are stored in a text file so that we can create a dataset for training the model for the classification between benign and malware.

3.3 Classification

Next, all the permissions and activities are processed which were stored in a text file for cleaning. In cleaning, we tried to remove all the duplicate permissions and activities that were present in the text file. Hence, names present in the text file are used for the column name in the dataset. After the process was over, value 1 is given for the name of the APK and the permissions and activities which it used are at the respective matching column name. Then, all the empty blocks are given as value 0 so that it will be easy to train the model. At last, all the models of machine learning are trained to find the best classification accuracy for detecting the benign and malware APK.

3.4 Evaluation Metrics

In general, best accuracy can be attained in detecting whether APKs are benign or malicious in two ways: static and dynamic phases. However, static phase is the best phase to detect and requires the least amount of time to classify. We decided that collecting data during the static phase would be the best way to begin training the entire machine learning models and then comparing them to find the best accuracy model for classifying

benign and malware. Evaluation metrics which are utilized to obtain the classification result of our experiment are explained briefly below.

- ***True Positive (TP):*** It is defined as the value of an resultant which describes the correctly predicted positive class
- ***True Negative (TN):*** It is defined as the value of an resultant which describes the correctly predicted negative class.
- ***False Positive (FP):*** It is defined as the value of an resultant which describes the incorrectly predicted positive class.
- ***False Positive (FN):*** It is defined as the value of as the resultant which describes the incorrectly predicted negative class.
- ***Confusion Matrix:*** It is the two-dimensional square matrix with rows and columns that shows the performance measurement of the model. Confusion matrix is depicted in Table 2

Table 2. Confusion Matrix

		Predicted Values	
Actual Values		Benign	Malware
	Benign	TP	TN
	Malware	FP	FN

- ***Accuracy*:** To find accuracy total value of true positives and true negatives is computed and then it is divided by all the instances.

$$\text{Accuracy} = (\text{TP} + \text{TN})/(\text{TP} + \text{TN} + \text{FP} + \text{FN})$$

4 Results and Observation

From the experimentation, it is shown that we have tried the ML models like DT, RF, KNN, Gradient Boosting Classifier, SVM, and Logistic Regression for this study. Out of all the algorithms, RF achieves an accuracy of 95% in classifying the malware. RF based classification output is depicted in Table 3 and the confusion matrix of RF is shown in Table 4. In addition, it is seen that the KNN was not so successful in detecting the malware as we got 79% of accuracy. KNN based classification output is provided in Table 5 and the confusion matrix of KNN algorithm is shown in Table 6. The other algorithms Decision Tree achieved 92% of accuracy where it was able to detect 93% of the malware and 91% of the benign APK files. DT based classification output is shown in Table 7 and confusion matrix of the DT is provided in Table 8. With the help of Gradient Boosting Classifier, we achieved 94% accuracy where it was able to detect 95% of the malware and 93% of the benign APK files. GB based classification output is depicted in Table 9 and confusion matrix of the Gradient Boosting Classifier is shown in Table 10. On using SVM, we got an accuracy of 91% where it was able to detect 92% of malware

and 90% of the benign APK. SVM based classification output is provided in Table 11 and confusion matrix of SVM is shown in Table 12. At the last, using Logistic Regression was able to get an accuracy of 93% where we got an accuracy of 91% where it was able to detect 94% of malware and 92% of the benign APK. LR based classification output is provided in Table 13 and confusion matrix of Logistic regression is shown in Table 14.

Table 3. Classification output using Random Forest Algorithm

Random Forest	Precision	Recall	F1-Score
Benign	0.91	0.91	0.91
Malware	0.93	0.93	0.93

Table 4. Confusion matrix using Random Forest algorithm

Random Forest	Benign	Malware
Benign	42	3
Malware	2	53

Table 5. Classification output using KNN algorithm

KNN	Precision	Recall	F1-Score
Benign	0.71	0.91	0.80
Malware	0.90	0.69	0.78

Table 6. Confusion matrix using KNN algorithm

KNN	Benign	Malware
Benign	41	4
Malware	17	38

Table 7. Classification output using Decision Tree algorithm

Decision Tree	Precision	Recall	F1-Score
Benign	0.91	0.91	0.91
Malware	0.93	0.93	0.93

Table 8. Confusion matrix using Decision Tree algorithm

Decision Tree	Benign	Malware
Benign	41	4
Malware	4	51

Table 9. Classification output using Gradient Boosting classifier algorithm

Gradient Boosting Classifier	Precision	Recall	F1-Score
Benign	0.95	0.91	0.93
Malware	0.93	0.96	0.95

Table 10. Confusion matrix using Gradient Boosting classifier algorithm

Gradient Boosting Classifier	Benign	Malware
Benign	41	4
Malware	2	53

Table 11. Classification output using SVM Algorithm

SVM	Precision	Recall	F1-Score
Benign	0.91	0.89	0.90
Malware	0.91	0.93	0.92

Table 12. Confusion matrix using SVM algorithm

SVM	Benign	Malware
Benign	40	5
Malware	4	51

As part of the experimentation, we tried three trials on each model, and accuracy value is computed in all the three trials which are illustrated in Figs. 2, 3 and 4 respectively. From the figures, we can see that the Random Forest is the most efficient model with 95% accuracy. The lowest accuracy of 79% is attained from KNN (k-Nearest Neighbors) and it is not so successful in classifying the malicious files.

Table 13. Classification output using Logistic Regression Algorithm

Logistic Regression	Precision	Recall	F1-Score
Benign	0.93	0.91	0.92
Malware	0.93	0.95	0.94

Table 14. Confusion matrix using Logistic Regression algorithm

Logistic Regression	Benign	Malware
Benign	41	4
Malware	3	52

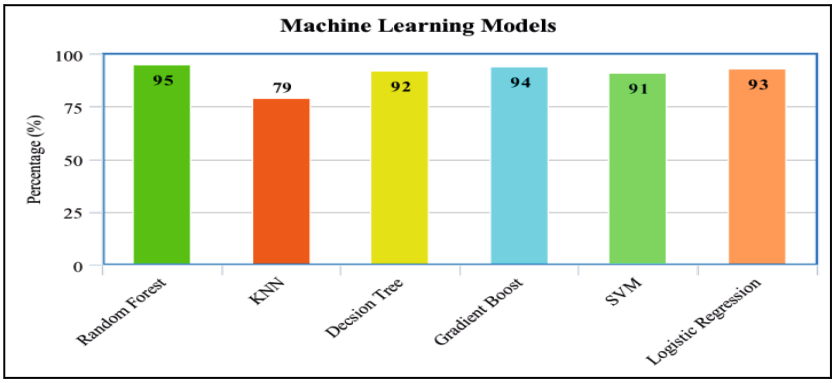

Fig. 2. Trail No 1. Accuracy (in %) vs Machine learning models

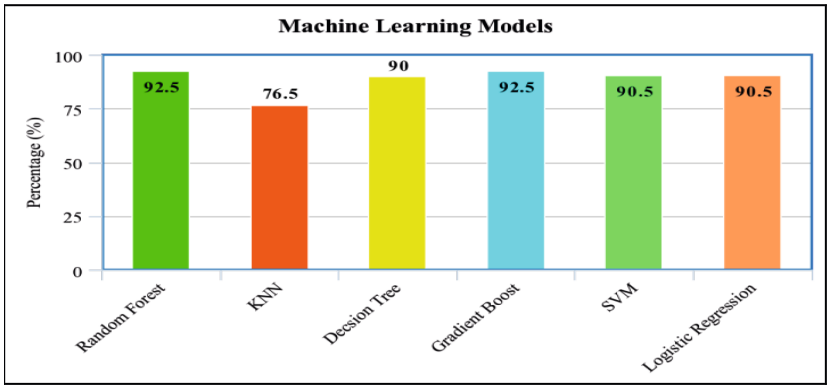

Fig. 3. Trail No 2. Accuracy (in %) vs Machine learning models

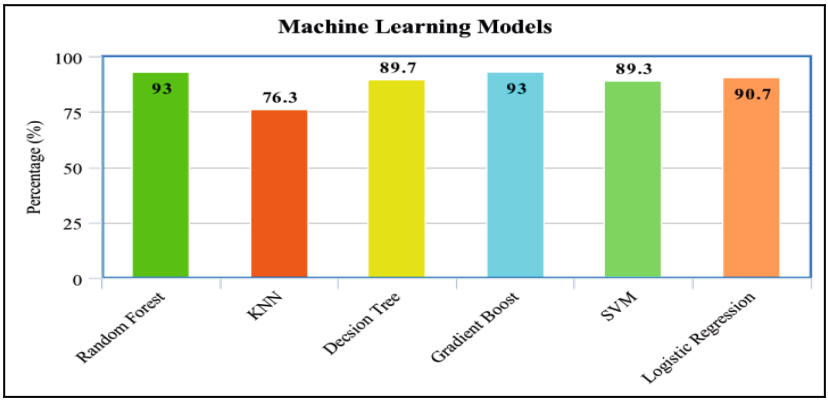

Fig. 4. Trail No 3. Accuracy (in %) vs Machine learning models

5 Conclusion

Detection of malicious apps is a highly challenging task because of the existence of a vast number of android apps and their distinct properties in business. Features like permission and activity are alone utilized for android malware detection without performing any dynamic analysis in this research work. Further, we have used various types of ML algorithms such as Random Forest, KNN (k-Nearest Neighbors), Decision Tree, Gradient Boosting Classifier, SVM (Support Vector Machine), and Logistic Regression to classify the malware and evaluated the performance of each and every machine learning algorithms. From the experimental results, we found that RF algorithm provides its best performance while comparing with other algorithms with the accuracy of 95%. With an accuracy of 79%, the KNN (k-Nearest Neighbors) is having the lowest detection rate. Also, we achieved the detection accuracy only by using the permission and activities feature without performing dynamic analysis.

In the future, we will try to work on the same analysis for deep learning models for practical deployment.

References

1. Digital:Trends. https://www.digitaltrends.com/android/smartphone-sales-exceed-those-of-pcs-for-first-time-applesmashes-record/ (2011)
2. Ciobanu, M.G., Fasano, F., Martinelli, F., Mercaldo, F., Santone, A.: A data life cycle modeling proposal by means of formal methods. In: Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security, pp. 670–672. ACM (2019)
3. Fasano, F., Martinelli, F., Mercaldo, F., Santone, A.: Energy consumption metrics for mobile device dynamic malware detection. Procedia Comput. Sci. **159**, 1045–1052 (2019)
4. https://www.avtest.org/fileadmin/pdf/security_report/AV-TEST_Security_Report_2019-2020.pdf
5. https://www.latimes.com/business/la-xpm-2012-nov-14-la-fi-tn-android-smartphone-market-share-20121114-story.html

6. Wu, D.-J., Mao, C.-H., Wei, T.-E., Lee, H.-M., Wu, K.-P.: DroidMat: android malware detection through manifest and API call tracing. In: 2012 Seventh Asia Joint Conference on Information Security. IEEE (2012)
7. Varma, P.R.K., Raj, K.P., Raju, K.V.S.: Android mobile security by detecting and classifying of malware based on the permissions using machine learning algorithm. In: International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC 2017) (2017)
8. Bhatia, T., Kaushal, R.: Malware detection in android based on dynamic analysis. In: 2017 International Conference on Cyber Security And Protection Of Digital Services (Cyber Security). IEEE (2017)
9. Darus, F.M., Salleh, N.A.A., Ariffin, A.F.M.: Android malware detection using machine learning on image patterns. In: 2018 Cyber Resilience Conference (CRC). IEEE (2018)
10. Koli, J.D.: Droid, R.: Android malware detection using random machine learning classifiers. In: 2018 Technologies for Smart-City Energy Security and Power (ICSESP). IEEE (2018)
11. Iqbal, S., Zulkernine, M.: SpyDroid: a framework for employing multiple real-time malware detectors on android. In: 2018 13th International Conference on Malicious and Unwanted Software: "Know Your Enemy". IEEE (2018)
12. Chavan, B., Tamna, B., Jaiswal, S., Nadkarni, S., Jawre, N.: Android malware detection based on machine learning. In: IRJET (2019)
13. Agrawal, R., Shah, V., Chavan, S., Gourshete, G., Shaikh, N.: Android malware detection using machine learning. In: 2020 International Conference on Emerging Trends in Information Technology and Engineering. IEEE (2020)
14. AlSobeihy, M., Altamimi, S., Salem, E., Alhazzani, H., Alhjaile, E.: Using machine learning to classify android application behavior. In: 2020 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE), pp. 1–4 (2020).https://doi.org/10.1109/CSDE50874.2020.9411630
15. Tahtaci, B., Canbay, B.: Android malware detection using machine learning. In: 2020 Innovations in Intelligent Systems and Applications Conference (ASYU), pp. 1–6 (2020).https://doi.org/10.1109/ASYU50717.2020.9259834
16. Sangal, A., Verma, H.K., A static feature selection-based android malware detection using machine learning techniques. In: 2020 International Conference on Smart Electronics and Communication (ICOSEC), pp. 48–51 (2020).https://doi.org/10.1109/ICOSEC49089.2020.9215355
17. Soni, H., Arora, P., Rajeswari,D.: Malicious application detection in android using machine learning. In: 2020 International Conference on Communication and Signal Processing (ICCSP), pp. 0846–0848 (2020).https://doi.org/10.1109/ICCSP48568.2020.9182170
18. Han, H., Lim, S., Suh, K., Park, S., Cho, S., Park, M.: Enhanced android malware detection: an SVM-Based machine learning approach. In: 2020 IEEE International Conference on Big Data and Smart Computing (BigComp), pp. 75-81 (2020).https://doi.org/10.1109/BigComp48618.2020.00-96
19. Fiky, A.H.E., Elshenawy, A., Madkour, M.A.: Detection of android malware using machine learning. In: 2021 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC), pp. 9–16 (2021). https://doi.org/10.1109/MIUCC52538.2021.9447661
20. Vivekanandam, B.: Design an Adaptive Hybrid Approach for Genetic Algorithm to Detect Effective Malware Detection in Android Division. J. Ubiquitous Comput. Commun. Technol. **3**(2), 135–149 (2021)