KICS The Korean Institute of Communications and Information Sciences

Available online at www.sciencedirect.com

ScienceDirect

ICT Express xxx (xxxx) xxx

www.elsevier.com/locate/icte

A Bayesian probability model for Android malware detection

Sharfah Ratibah Tuan Mat, Mohd Faizal Ab Razak*, Mohd Nizam Mohmad Kahar, Juliza Mohamad Arif, Ahmad Firdaus

Faculty of Computing, UMP, Pahang, Malaysia

Received 11 July 2021; received in revised form 5 September 2021; accepted 7 September 2021
Available online xxxx

Abstract

The unprecedented growth of mobile technology has generated an increase in malware and raised concerns over malware threats. Different approaches have been adopted to overcome the malware attacks yet this spread is still increasing. To combat this issue, this study proposes an Android malware detection system based on permission features using Bayesian classification. The permission features were extracted via the static analysis technique. The 10,000 samples for the judgement were obtained from AndroZoo and Drebin databases. The experiment was then conducted using two algorithms for feature selection: information gain and chi-square. The best accuracy rate of detection of permission features achieved was 91.1%.

Keywords: Mobile phone; Android malware; Bayesian; Information gain; Chi-square

1. Introduction

The evolution of mobile devices and mobile applications has drastically changed our ways of engaging in our everyday lives. Web browsing, online banking, online shopping, social networking [1], and online learning are examples of services of mobile devices through connection with the Internet. Therefore, mobile devices have played a crucial role and have become an essential part of human life [2]. The number of mobile users in 2020 has reached 4.78 billion [3]. For these 4.57 billion people worldwide that used the Internet [4], they would have spent 1.25 billion years of human time online in 2020, and it is still expanding consistently. However, despite bringing convenience to users, the mobile device is also facing malware invasion and attacks due to online social networks and online services [5]. Mobile malware can trick as a standard code, then modify any intended application to corrupt and interfere with the system's functionality. According to [6], ransomware has thrived 72% during the COVID-19 pandemic, and mobile vulnerabilities have increased by 50%.

As a protection method, Google Play has provided a permission-based system as a security mechanism that restrains the application from accessing confidential data [7]. This permission prompts the users before the installation by considering the resources of the application accessed. The users have to accept the agreement explicitly before continuing with the installation. Unfortunately, the mechanism provided by Google Play cannot protect the user entirely, as they tend to accept the agreement without a thorough inspection of the permission. Consequently, most mobile phone users are prone to suffer damages, as they ignore the permission risk that results in abuse of the application. The security researcher investigated the vulnerabilities of Google's Bouncer application to detect malware applications [8]. Hence, the study of Android malware is significant to regain the deficiency.

Machine learning studies are becoming popular as it is an effective approach that can achieve a high detection of accuracy [9,10]. Machine learning is a system of artificial intelligence (AI) that automatically improves and makes the decision to learn from the data and patterns [11]. The concept of machine learning is a minimised human mediation in the computational system. Machine learning predicts the data through computational learning theories and learns from experience or historical data. It has supervised and unsupervised classifiers that are used to analyse the features and trace the model [12]. The approach provided by machine learning is to assist in the validation of normal and malicious activities. Support Vendor Machine (SVM), Naïve Bayes, K-nearest neighbour (KNN), and Ensemble classifier are examples of the algorithm used in

* Corresponding author.
E-mail address: faizalrazak@ump.edu.my (M.F.A. Razak).

Peer review under responsibility of The Korean Institute of Communications and Information Sciences (KICS).

https://doi.org/10.1016/j.icte.2021.09.003

machine learning. A few studies present malware detection by using a genetic algorithm [13–15], and also [16] use the SVM algorithm for detecting malware. A study by [1] applied Bayes algorithm to analyse cyber-terrorism on the internet. Studies of [17,18], and [19] also used Bayesian algorithms to detect malware.

In addition, malware classification is also significant in the determination of machine learning detection. Classification of malware is a process of categorising a collection of malwares into target items based on categories, families, and classes. This process has a similarity with the data mining function. Naïve Bayes and SVM are examples of classification techniques that are frequently engaged in classification research. The new and favoured classifier in machine learning is the Naïve Bayes classification model, which is used in static analysis methods. It uses the Bayes' Theorem and is a supervised machine-learning algorithm. Many studies of Naïve Bayes are used in different fields of study as indicated in [20,21], and [22]. Naïve Bayes calculates the probability of each event and predicts the class label to problem instances and presents the values of the features. It is a simple technique and produces a high degree of accuracy in detection.

This study proposes the Bayesian probability method on permission-based features to accomplish the examination of signature-based malware. The permission-based features were extracted using static analysis that was classified using the Bayesian classifier to yield the results of the prediction of the malware. Specifically, this study applies malware detection to generate an outcome that allows users to recognise the threat in Android apps, thus preventing their mobile phones from being infected by the malware threat. The main contributions of the study are as follows:

a. The experiment practised 10,000 datasets, each containing 5000 malware and benign samples.
b. The experiment applied the permission features using the static analysis technique.
c. The malware detection applied information gain and chi-square algorithms with a Bayesian classifier.

2. Related work

Bayesian classification is the latest machine learning model in the static analysis method. It uses Bayes' Theorem and is one of the supervised learning machines that considers self-reliance features statistically [20,23]. The comprehensive studies of Naïve Bayes have persisted a standard method with appropriate pre-processing. The Bayesian measure is the belief that probability is subjective and refers to the future. Understanding Bayesian starts with a belief called 'prior'. Then, by using some data to update the belief, the outcome is called 'posterior'. Probability refers to past experience (prior) and predicts the future (posterior). The old posterior becomes a new prior in the big data cycle and process, and the cycle repeats. The Bayes rule included in this step is as follows:

$$\boldsymbol{P}\,(\boldsymbol{A}|\boldsymbol{B}) = \boldsymbol{P}\,(\boldsymbol{B}|\boldsymbol{A}) * \boldsymbol{P}(\boldsymbol{A})/\boldsymbol{P}(\boldsymbol{B}) \quad (1)$$

$P(A|B)$ is the 'probability', indicating a conditional probability. It means 'what is A if B happens?' and is also called Bayesian machine learning. In the equation above, $P(A)$ is the prior belief, $P(B|A)$ is the likelihood of data B, $P(B)$ is the normalising constant or evidence, and $P(A|B)$ is the posterior obtained after the process. The interpretation of odds as relative frequencies is one of the meanings of probability. Easy games containing coins, balls, dice, and roulette wheels are used as examples of probability. Similarly, predictions of the likelihood of malware being present in an Android application are based on the relative frequency of incidence in a wide number of cases.

The Bayesian Theorem is used in various fields of classification studies [20,21], and [22]. The simple technique in Naïve Bayes that produces the detection, has the lowest error rate, assigns the class label, and represents the values of the feature as a vector. [24] uses the Naïve Bayes classification method and combines it with the chi-square filtering test. Sharma et al. [18] proposed a method on the M0Droid dataset by completing the test on a large dataset from AndroZoo and Drebin. By using 400 samples from M0Droid, it was possible to achieve high accuracy by improving the F-measure. In comparison, the proposed method in this study has a similarity with the features used, but differs in the process that uses Bayesian classification through static analysis for detecting malware present in an application.

Wu et al. [25] generated an application software during runtime and analysed network traffic using Bayesian updating. The experiment was conducted with 557 samples from 14 types of malware. The features were selected using an information gain algorithm. The malware was detected on the analysis of traffic characteristics and feasible in IoT operative to achieve higher accuracy. The study by [25] differs from the current studies but has similarity with the classifier used.

Suleiman et al. [26] have presented a practical approach to alleviating the problem of detecting malicious apps using static analysis based on Bayesian classification. The study used 2000 samples retrieved from static analysis, with 49 of them identified in malware families and a comprehensive benign application. A Java-based package has been implemented to obtain the features set for the Bayesian model. The similarity in the study is that it used a reverse engineering technique but differs by using a big scale sample with re-trained new samples.

Apart from Bayesian algorithms, machine learning also employs a few algorithms for malware detection, such as SVM, RF, Adaboost, and KNN. Each algorithm takes a different role in the detection of the presence of malware. SVM is a machine learning algorithm for statistical learning theories, such as dimension theory and structural risk minimisation theory. RF ensembles learning on a decision tree by integrating numerous trees. Adaboost classifier practice is a weak classifier and acts as a base by allocating different weights, and then assembles the prediction weight as an output. KNN is a supervised algorithm that runs the detection by using the training dataset and classifies the data that frequently appears in KNN. It is also known as a lazy learning algorithm.

2.1. Permission-based features

Google Play has introduced 'permission-based' as a method to prevent the infection of malware into Android mobile [27]. The user is prompted to permit access before the installation of the application or network access. To proceed with the installation, users need to accept the requested permission or halt the installation. The list of permissions used in the Android application is stored in the AndroidManifest.xml <uses-permissions> tag file. Permission is the security mechanism [13] that restricts the access of applications to the credential part stored in a mobile phone, such as API and crucial information. The restriction is created to protect sensitive data by limiting access to a suspicious application. Thus, permission requested is necessary upon the installation of any application in a mobile device and is incorporated with API analysis features, which is the most useful for detecting malware on Android devices.

The protection of permission is split into normal, signature, and dangerous [28]. The permission provided is a shield to protect the confidential matters of Android users [29]. The system would grant automatically if normal permission is declared in an application manifest file. In contrast, dangerous permission affects crucial information stored by a user or creates chaos for other applications. Besides, the signature permission is only granted at the time of installation and is able to be used when the permission signed is the same as defined in the certificate [29]. Some of the permissions carry a risk to the information and data of the user and need permission to be granted by the system.

3. Proposed method

Bayesian probability is used for malware detection, focusing on permission-based features. The application characteristic was retrieved from static analysis using extensive malware samples that contain a variety of benign and malware. The proposed system considers the most dominant permission that contributes to malware detection. Bayesian probability was selected as a method due to its sturdy mathematical base and stability of classification competency. Moreover, Bayes' rule is one of the fundamental pillars of probability and the probability theory implemented in the computerised system. The equation of Bayes' rule is as follows:

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)} \tag{2}$$

According to Eq. (2), **A** is the event for the probability, and **B** is the new evidence that is related to A in some way. The event to be estimated is called posterior **P(A|B)**, which is the probability of getting malware when the phone gets the virus. When the probability is observed, the new evidence is **P(B|A)**, which is also called likelihood, becomes the initial hypothesis. The prior is **P(A)**; that is, the event that is not required is the additional information of prior. The theory is similar to the example that probability has malware. The marginal likelihood is **P(B),** which contains the total probability of observing the evidence.

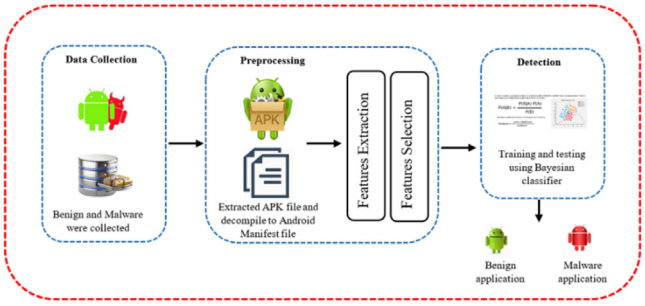

Fig. 1. The Bayesian classifier detection system.

3.1. General architecture

The architecture of the main system is presented in Fig. 1. They are divided into three phases: dataset collection, preprocessing phase, and a detection phase. These phases are integrated into each other.

3.1.1. Dataset collection phase

The detection process starts with collecting benign and malware datasets from Drebin and AndroZoo, which comprises 10,000 samples. The benign application consists of 5000 applications retrieved from AndroZoo, owned by Google Play store. Meanwhile, the malware application consists of 5000 applications that were downloaded from the Drebin database. This collection comprises all permissions for benign and malware applications. This process includes decompiling the .apk file, extracting, and filtering permissions. The permissions are collected and saved as a .arff file in a readable format. The .arff file holds all the attributes of the functionality used to optimise functionality so that noise and irrelevance are excluded. Then, the downloaded datasets are extracted and compiled in a .csv file format.

The samples were predefined manually, then labelled, such as benign or malware. The inspection of the Android application status in VirusTotal is completed with the labelling process of malware and benign. It is widely used by researchers in their fields [30,31]. VirusTotal is an online website provided for checking viruses through URLs and uploaded files. The samples of malware were run through VirusTotal as the VirusTotal completed the validation. Additionally, benign datasets are screened in VirusTotal to determine if they are malicious or not. The dataset sample was then used for feature selection.

Table 1 presents the top 20 malware families used in this experiment. This table contains four (4) columns, namely features (SHA256), virus total, family, and total permission. SHA-256 is a proprietary cryptographic hash function that produces a 256-bit value. SHA256 is retrieved from the VirusTotal website and prepares a unique key in each sample. The virus total next to the SHA256 column presents the difference between the malware and benign dataset. This experiment only used an application dataset with a ranking of 0/50, meaning that the APK is deceptive according to the number of optimistic claims.

Table 1
Top 20 malware families.

Features (SHA256)	Virus total	Family	Total permission
1d22924bbe5dce7696e18d880482b63ce19ca0746f8671aaec865cce143f6e6f	37	Arspam	14
a3a8ec093ba12db3b6e82a385985b84829c84de9abbe5db17a4b2d2fb1715cb7	27	Plankton	22
78c1c57100bc14f9689c3f670d48405d9eb7487df1a34a846296f8dd4ab34e33	36	Plankton	21
602205ae45915a3954897b83d7261a39b5d0b0803b1c36e79633ffe027c83983	39	Plankton	21
8728a1a0b31748f97695c26dc94a4124cf84001444980df01fcc2b731beb30bc	31	Plankton	19
8cb40e8dce05482907ff83b39911831daf20e4a69ee63a6cff523c880eed1acf	38	Rootsmart	8
06b53d3ea2aeee828123194b4cea8135f5b868296d8d7ab3cb839e34b2f04d6a	39	Adrd	13
294cfb2bc890b65d7bc9135225369ab9bbd0ca81baa109f829e2c22478b4db2f	37	Adrd	13
4cff30bad920382c6e2f6833b5cdd8b4b0fcde3df0b47167a552c82096d97fc3	40	Adrd	13
4d05c2c2f0c0881f8e0c3421da4d6656cc8507ac15658a2fb1309f77e72bd94d	39	Adrd	13
5a5dae9254618bb49428654050069387c8cb2931159953c9566fba9341f31666	40	Adrd	13
5e8c8a69fc749194f86bfcdf285be45b47e675cbe4792616fcd0d00b4584eec0	42	Adrd	13
6e28480d51bf723529c48c6320eb7fcb2a5a8ace8ea2415201da7bae87adb48d	41	Adrd	13
70fd9fbcc343eca79152aacba595bece9d3ca6e5d4bcae02da0d90d8138d952a	38	Adrd	13
9bcf0a67e128813b8cfcc2bc37df0b7ced18459a16ef11e79c7eb6385e4763b7	38	Updtkiller	12
aabe5b64af5e841e02392865dc10dcd2df499ec6448392270209 99b3ee9a87ec	40	Adrd	13
b092c3fb1def71dba11efc28ccb2e29c8aea779e82037538875834152ca0967f	39	Adrd	13
c82ab78288ac8b5fd9cce0b2701f034320dcc6cfbe2118527713b9121a9d35b0	35	Plankton	21
cb116cefc4cb1004b06d7726258006bb2db468c3dd4ce486b1035a29f6b43305	40	Adrd	13
e2190399392695d8768d8315278e739d0453b3047c2b76dc4dd19911f3fd9598	38	Adrd	10

3.1.2. Pre-processing phase

APK is decompiled to parse AndroidManifest.xml containing permissions and compiles classes.dex with all Dalvik bytecodes, such as API calls through static analysis. The feature extraction needs different tools, such as Androguard and ApkTool, to gain AndroidManifest.xml and classes.dex files that comprise permission, system call, intent, and native code. The AndroidManifest file stores all the information of the Android application like permission. The tag of *<uses-permission/>* classifies the types of permission requested in the application. The APK file is decompressed and converted from the binary manifest file to readable XML format. The Androguard tool is used to obtain permission tags.

The sample of a dataset that contains permission features is labelled as benign or malware in the last column to be stored as .arff file in the database. A total of 10,000 samples indicated by hash numbers to avoid duplications was saved in binary format (.csv) and subsequently, transformed into the .arff (attribute relation file format) format file. These features served as attribute values and were termed as 'feature sets'. The binary value representing permission requested or not by that application was stated as '1' for permission requested by a specific application, while '0' for permission not requested by the specific application.

The feature selection technique is widely used for data pre-processing. This is to find the most valuable features by extending and filtering certain features that are irrelevant or have a negative impact on the results of classification. The advantages of feature selection are decreasing the period for training and testing, and improving efficiency and accuracy. Two studies [32] used permission-based features, [33] used API call sequence features, while [32] and [34] used intent. Implementing feature optimisation in the permission features is time-consuming in training and testing, but reduces over lifting and simplifies malware detection. On the other hand, optimisation is able to increase the accuracy of detection in the experiment. The process of optimisation starts with cleaning the dataset to remove artefacts and redundant features. The missing value data were filled with zero before the randomising process. This is because the filtering process predicted is at random. This process allows possible tendencies to be removed during the experiment.

The next step is optimising the best algorithm to use in the experiments. Several algorithms were implemented using WEKA (Waikato Environment for Knowledge Analysis) to gain randomisation of data and the classification purpose. The purpose is to optimise and regulate the dataset to achieve the highest possible feature sensitivity. As a result of the best algorithms, we have employed information gain and chi-square to optimise feature selection, as these algorithms produce the best performance according to our dataset. Chi-square and information gain are able to increase the efficiency of classifier performance [24,35,36]. The chi-square and information gain play the role in identifying and selecting the most important features. The purpose of using these algorithms in this study is to find the best performance of accuracy.

Two algorithms were chosen to find the dissimilarity and to get the best accuracy according to feature selection. As many as 10,000 Android applications for benign and malware samples were used in the experiment. The feature selection in this study was implemented by the Waikato Environment For Knowledge Analysis (WEKA). It is a well-developed software that provides a set of algorithms for machine learning [9]. The information gain and chi-square were implemented directly to our dataset.

3.1.3. Detection phase

This study applied Bayesian probability in a machine learning approach to detect malicious codes in Android applications. Bayesian probability has excellent statistical properties that classify some malicious code as benign and some benign

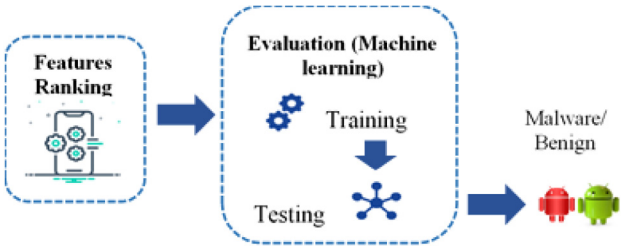

Fig. 2. Malware detection using machine learning.

code as malicious, and can potentially detect unknown malicious code. Fig. 2 illustrates the architecture of the Android malware detection system using probability prediction.

The likelihood of a feature vector application in Y class, $\overline{x} = (x_1, x_2, \ldots, x_n)$ is defined by the theorem of Bayes:

$$P\left(Y = y|\overline{X} = \overline{x}\right) = \frac{P\left(Y = y\right)\prod_{i=1}^{n} P(X_i = x_i|Y = y)}{\sum_{j\in\{0,1\}} P(Y = y_j)\prod_{i=1}^{n} P(X_i = x_i|Y = y_j)} \quad (3)$$

where the predictable frequencies measured for the learning app are $P\left(X_i = x_i|Y = y\right)$ and $P(Y = y_j)$. n represents the number of a feature used in the classification, while y_0 and y_1 are the benign and suspicious class, respectively. A vector app $\overline{x} = (x_1, x_2, \ldots, x_n)$ is classed as benign if $P\left(Y = benign|\overline{X} = \overline{x}\right) > P(Y = suspicious|\overline{X} = \overline{x})$.

In order to evaluate the detection performance of the Bayesian classifier, the dataset was divided into two parts: the training set and the testing set during the learning phase. The training set is a crucial part of training the dataset to obtain accurate detection. The samples used for the training set consisting of malware and benign applications were used. For the testing and evaluation, ten-fold cross-validation was used based on the evaluation criteria provided. Thus, 70% of the samples were used for training, and the remaining 30% were used for testing. All the permissions used in this phase by benign and malware were determined, and the features with the minimum possibility to identify the class target were filtered. The result of the process will be measured based on the measurement of accuracy. The measurement of the Positive Rate is described as, $Positive\ Rate = \frac{TPR}{TPR+FNR}$, Negative Rate described as, $Negative\ Rate = \frac{FPR}{TNR+FPR}$ and the overall accuracy is given as $\frac{TPR+TNR}{TPR+TNR+FPR+FNR}$.

4. Result & discussion

Information gain and chi-square are the algorithms used for feature selection, combined with Naïve Bayes classifier for detection. The experiment is divided into four different sets of features ranked from 15, 20, 25, and 30. As illustrated in Table 2, the features of both algorithms have a similarity. The performance of the classifier has been evaluated by four (4) metric assessments, such as TPR, FPR, precision, and F-measure. The accuracy of the detection is presented in Table 2.

As noted in Table 2, the performance of detection using a Bayesian classifier with different features selected by using

Table 2
Detection performance result using Naïve Bayes.

Features optimisation	Number of features	TPR	FPR	Precision	F-measure
Information Gain	30	90.2%	0.096	0.906	0.902
	25	89.4%	0.104	0.899	0.894
	20	88.65%	0.116	0.891	0.886
	15	87.55%	0.123	0.879	0.875
Chi-Square	30	91.1%	0.091	0.914	0.911
	25	89.6%	0.103	0.900	0.896
	20	90.6%	0.094	0.906	0.906
	15	91.0%	0.091	0.911	0.91

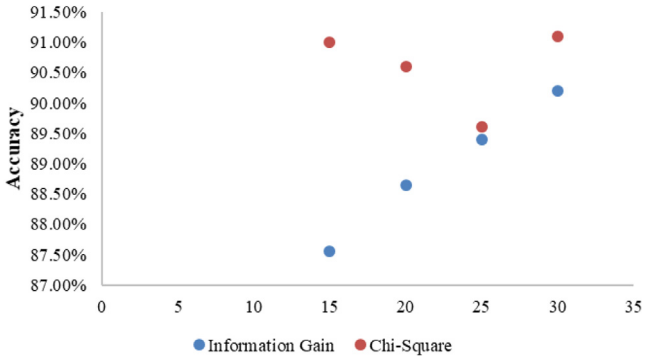

Fig. 3. Illustration of accuracy performance.

the information gain and chi-square algorithms. For both algorithms, the accuracy of detection increased according to the number of features selected. It seems when the features are optimised, the accuracy increases. However, the accuracy of both algorithms was slightly lower in 20 features. The four features excluded in the 20 features in the information gain algorithm were READ_SETTING, WRITE_APN_SETTING, READ_CONTACT, and ACCESS_WIFI_STATE. On the other side, five features excluded in chi-square were INSTALL_SHORTCUT (Motorola.launcher), INSTALL_SHORTCUT (lge.launcher), INSTALL_SHORTCUT (Motorola.dlauncer), READ_SETTING (fede.launcher), and READ_SETTING (adw.launcher). Thus, the decrease in accuracy for 20 features is related to the exclusion of features in the selected 20 features. Moreover, removing the five features from each algorithm impacts the detection accuracy as the removed features significantly influence the accuracy of detection. However, the feature selection using chi-square and information gain algorithms in this study achieve high accuracy with 91.1% and 90.2%, respectively. Hence, it can be concluded that feature optimisation is also critical in defining reliable features [37].

Fig. 3 illustrates the comparison of the algorithm used in this study. The curve shows that the accuracy of chi-square is higher compared to information gain. As noted, the detection drops to 25 features for both algorithms. To conclude, the highest detection performed by 30 features is 91.1% for chi-square and 90.2% for information gain.

In addition, we compared our detection results of the same classifier to other related studies, as demonstrated in Table 3.

Table 3 reveals the comparison of the proposed method with previous studies, considering the same classifier. Studies

Table 3
Detection performance result using Naïve Bayes.

Reference	Dataset (source)	Accuracy	Algorithm
[17]	100 samples (Drebin and Google Play)	94%	Naïve Bayes
[13]	1119 malware (Android Malware Genome) and 621 benign (Google Play)	93.85%	Genetic Algorithm and Naïve Bayes
[32]	5560 malware (Drebin) and 1846 benign (Google Play)	83%	Naïve Bayes
[38]	Worm dataset	95%	Naïve Bayes
Proposed method	5000 malware (Drebin) and 5000 benign (AndroZoo)	91.1%	Chi-Square & Naïve Bayes

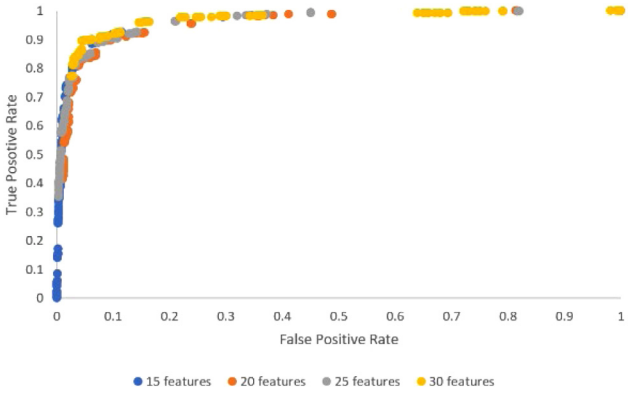

Fig. 4. Performance of ROC curve.

by [17] and [38] achieved higher accuracy with a few significant features. Other studies [13] achieved 93.85% accuracy using 1740 samples of benign and malware datasets. The malware dataset (1119) was taken from the Android Malware Genome Project (AMGP) that was issued by Yajin Zhou and Xuxian Jiang in 2012. They used 152 permission features for training and tested using Naïve Bayes, SVM, and DT classifiers. Studies by [17] and [32] have similarities with this study that used the Drebin database for malware detection.

In contrast, this proposed study has used the AndroZoo database for benign detection, while the other studies used Google Play as their database. The results of this study's experiments may be marginally smaller compared to the previous study. However, the malware dataset used in this study is massive in comparison to other studies. A wide range of datasets contributes to enhanced malware classification efficiency [39, 40]. In addition, the significant and minimal features can produce acceptable detection performance of more than 90% while reducing model resources and constraints. To conclude, the collection of database and source retrieved affected the accuracy of the detection of malware.

4.1. ROC curve

The ROC curve visualised and interpreted the evaluation and performance of TPR in graphical form. The ROC curve measured the value and effectiveness of the prediction classifier. Fig. 4 presents the different TPR and FPR through cross-validation with 15, 20, 25, and 30 features using the Naïve Bayes classifier. The left axis denotes the percentage of samples that are appropriately detected as malware. It is clearly shown in this figure that the entire features are close to left and top border. Hence, the ROC curve shows a high accuracy rate with minimal false alarm. Additionally, the region under the ROC curve, known as an area under the curve (AUC), is to optimise the detection and balance classifier output on malware collection.

Table 4
Result of AUC.

Features optimisation	Number of features	AUC
Information Gain	30	0.9655
	25	0.9597
	20	0.9609
	15	0.9548
Chi-Square	30	0.9675
	25	0.9653
	20	0.9575
	15	0.9639

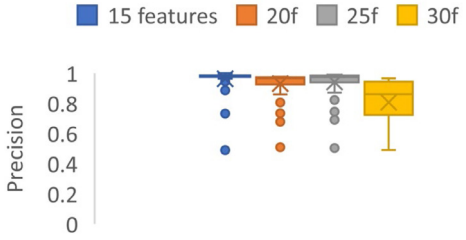

Fig. 5. Precision.

Table 4 lists the results of AUC taken from the experiment for 15, 20, 25 and 30 features. The details of the results of AUC are described as where the malware detection performance is measured with a threshold of 1.0, which is equal to perfect detection. The result of AUC in Table 4 indicates that all the detection based on different features have excellent prediction rates, and AUC values are appropriate to detect malware on the Android platform.

4.2. Empirical assessment

To obtain the following results, the experiment is validated using a different parameter for the measurement process of detection. This study chooses the result obtained from the chi-square algorithm, as this algorithm achieves better performance. Figs. 5, 6 and 7 present the validation test results for precision, recall, and F-measures, respectively.

The results demonstrated in the figures indicate that most feature detection revealed high precision, recall, and F-measure

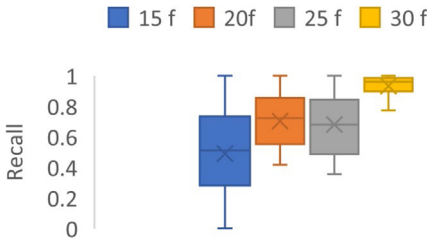

Fig. 6. Recall.

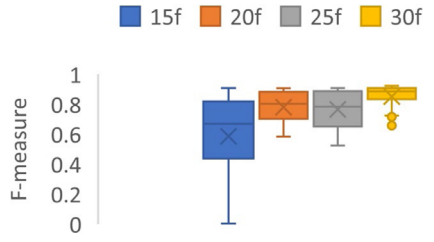

Fig. 7. F-measure.

values achieved. Recall and accuracy are vital in the evaluation of detection performance [41]. The increase in precision rate proves the model classification is linked to a positive rate. The detection of malware also presents an accurate result. Nevertheless, the high recall illustrates that malware features are similar to benign. Thus, the visualised result in the box plot describes that the precision with high recall effectively detects malware performance.

5. Conclusion

This study has presented an analysis of permission-based features using static analysis. In order to improve the malware detection, these features were optimised using information gain and chi-square algorithms. In this work, the selected permission features were separated into several groups to find the best performance of accuracy. As a result, the chi-square algorithm with 15 features, produced the best performance among the features with a 91% accuracy rate. With the high accuracy achieved, the proposed system appeared to provide good performance for malicious applications on Android mobile. As a future work, the existing work may extend to other features and evaluate the risk of each feature, as well as the study of Sharma et al. Risk assessment is considered a crucial part of malware detection to categorise, prioritise, and zone the permission request [42]. The practice of risk assessment would be able to raise the awareness of mobile users about potential malware damages.

The information is expected to encourage more research in the future as a way to overcome the rapid growth of malware. Additionally, the outcome of this study may improve the marketability of mobile security in Android operating systems, particularly for permission-based. In conclusion, this study provides a general overview of the topic and aims to show the importance of expansion in Android malware research.

CRediT authorship contribution statement

Sharfah Ratibah Tuan Mat: Conception and design of study, Analysis and/or interpretation of data, Writing – original draft, Writing – review & editing. **Mohd Faizal Ab Razak:** Acquisition of data, Writing – review & editing. **Mohd Nizam Mohmad Kahar:** Writing – review & editing. **Juliza Mohamad Arif:** Analysis and/or interpretation of data. **Ahmad Firdaus:** Acquisition of data.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The work is funded by the Ministry of Higher Education, Malaysia FRGS RACER/1/2019/ICT03/UMP/2 under Project ID: RDU192613 and RDU192607. The authors thank anonymous reviewers for their constructive comments and University Malaysia Pahang for their support.

All authors approved the version of the manuscript to be published.

References

[1] I. Castillo-Zúñiga, F.J. Luna-Rosas, L.C. Rodríguez-Martínez, J. Muñoz-Arteaga, J.I. López-Veyna, M.A. Rodríguez-Díaz, Internet data analysis methodology for cyberterrorism vocabulary detection, combining techniques of big data analytics, NLP and semantic web, Int. J. Semant. Web Inf. Syst. 16 (1) (2020) 69–86, http://dx.doi.org/10.4018/IJSWIS.2020010104.

[2] R.K. Archana Singh, A Two-Phase Load Balancing Algorithm for Cloud Environment, 2021.

[3] A. Turner, How Many Smartphones are in the World, Bankmycell, 2020, [Online]. Available: https://www.bankmycell.com/blog/how-many-phones-are-in-the-world#:~:text=According to GSMA real-time, mobile device in the world. & text=Statista predicts that by 2023, will increase to 7.33 billion. [Accessed: 01-Feb-2021].

[4] Hootsuite, Digital Around the World, Datareportal, 2020, [Online]. Available: https://datareportal.com/global-digital-overview. [Accessed: 02-Feb-2021].

[5] T.-S.M. Steven Yen, Melody Moh, Detecting Compromised Social Network Accounts using Deep Learning for Behavior and Text Analyses, 2021.

[6] S. Security, COVID-19 Pandemic Sparks 72% Ransomware Growth, Mobile Vulnerabilities Grow 50%, Cision, 2020, [Online]. Available: https://www.prnewswire.com/in/news-releases/covid-19-pandemic-sparks-72-ransomware-growth-mobile-vulnerabilities-grow-50--817268901.html. [Accessed: 08-Sep-2020].

[7] C.H. Liu, Z.J. Zhang, S. De Wang, An android malware detection approach using Bayesian inference, in: Proc. - 2016 16th IEEE Int. Conf. Comput. Inf. Technol. CIT 2016, 2016 6th Int. Symp. Cloud Serv. Comput. IEEE SC2 2016 2016 Int. Symp. Secur. Priv. Soc. Netwo, 2017, pp. 476–483, http://dx.doi.org/10.1109/CIT.2016.76.

[8] R. Lemos, Google Bouncer Vulnerabilities Probed by Security Researchers, Eweek, 2020, [Online]. Available: https://www.eweek.com/security/google-bouncer-vulnerabilities-probed-by-security-researchers. [Accessed: 29-Apr-2020].

[9] M.F.A. Razak, N.B. Anuar, F. Othman, A. Firdaus, F. Afifi, R. Salleh, Bio-inspired for features optimization and malware detection, Arab. J. Sci. Eng. 43 (12) (2018) 6963–6979, http://dx.doi.org/10.1007/s13369-017-2951-y.

[10] S.R.T. Mat, M.F.A. Razak, M.N.M. Kahar, J.M. Arif, A. Firdaus, Towards a systematic description of the field using bibliometric analysis: malware evolution, Scientometrics 126 (2021) http://dx.doi.org/10.1007/s11192-020-03834-6, Submitted for publication.

[11] Q.Z.S. Brij, B. Gupta, Machine Learning for Computer and Cyber Security. Principle, Algorithms, and Practices, 2019.

[12] N.N.M. Nasri, M.F.A. Razak, R.D.R. Saedudin, S. Mohamad-Asmara, A. Firdaus, Android malware detection system using machine learning, Int. J. Adv. Trends Comput. Sci. Eng. 9 (1 Special Issue 5) (2020) 327–333, http://dx.doi.org/10.30534/ijatcse/2020/4691.52020.

[13] O. Yildiz, I.A. Doğru, Permission-based android malware detection system using feature selection with genetic algorithm, Int. J. Softw. Eng. Knowl. Eng. 29 (2) (2019) 245–262, http://dx.doi.org/10.1142/S0218194019500116.

[14] N. An, A. Duff, G. Naik, M. Faloutsos, S. Weber, S. Mancoridis, Behavioral anomaly detection of malware on home routers, in: Proc. 2017 12th Int. Conf. Malicious Unwanted Software, MALWARE 2017, Vol. 2018-Janua, 2018, pp. 47–54, http://dx.doi.org/10.1109/MALWARE.2017.8323956.

[15] J. Lanet, C.T. Eds, I. Conference, and D. Hutchison, For Information Technology, Springer International Publishing, 2018.

[16] X.L. Hu, L.C. Zhang, Z.X. Wang, An adaptive smartphone anomaly detection model based on data mining, Eurasip J. Wirel. Commun. Netw. 2018 (1) (2018) http://dx.doi.org/10.1186/s13638-018-1158-6.

[17] S. Singhal, S. Maheshwari, M. Meena, Recent Findings in Intelligent Computing Techniques, Vol. 707, 2019, pp. 229–238, http://dx.doi.org/10.1007/978-981-10-8639-7.

[18] K. Sharma, B.B. Gupta, Towards privacy risk analysis in android applications using machine learning approaches, Int. J. E-Serv. Mob. Appl. 11 (2) (2019) 1–21, http://dx.doi.org/10.4018/IJESMA.2019040101.

[19] S.B. Almin, M. Chatterjee, A novel approach to detect Android malware, Procedia Comput. Sci. 45 (C) (2015) 407–417, http://dx.doi.org/10.1016/j.procs.2015.03.170.

[20] O. Koucham, T. Rachidi, N. Assem, Host intrusion detection using system call argument-based clustering combined with Bayesian classification, in: IntelliSys 2015 - Proc. 2015 SAI Intell. Syst. Conf., 2015, pp. 1010–1016, http://dx.doi.org/10.1109/IntelliSys.2015.7361267.

[21] S.S. Damawale, P.V.V. Bag, Classification of unstructured data using machine learning algorithm, 5 (5) (2016) 72–76.

[22] A. Choi, N. Tavabi, A. Darwiche, Structured features in naive bayes classification, in: 30th AAAI Conf. Artif. Intell. AAAI 2016, 2016, pp. 3233–3240.

[23] C. Yuan, S. Wei, Y. Wang, Y. Yue, S.G. ZiLiang, Android applications categorization using Bayesian classification, in: Proc. - 2016 Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discov. CyberC 2016, 2017, pp. 173–176, http://dx.doi.org/10.1109/CyberC.2016.42.

[24] L. Yu, Z. Pan, J. Liu, Y. Shen, Android malware detection technology based on improved Bayesian classification, in: Proc. - 3rd Int. Conf. Instrum. Meas. Comput. Commun. Control. IMCCC 2013, 2013, pp. 1338–1341, http://dx.doi.org/10.1109/IMCCC.2013.297.

[25] F. Wu, L. Xiao, J. Zhu, Bayesian model updating method based android malware detection for IoT services, in: 2019 15th Int. Wirel. Commun. Mob. Comput. Conf. IWCMC 2019, 2019, pp. 61–66, http://dx.doi.org/10.1109/IWCMC.2019.8766754.

[26] S.Y. Yerima, S. Sezer, G. McWilliams, I. Muttik, A new android malware detection approach using Bayesian classification, in: 27TH International Conference on Advanced Information Networking and Applications (AINA), 2013, pp. 121–128, http://dx.doi.org/10.1109/AINA.2013.88.

[27] A. Qamar, A. Karim, V. Chang, Mobile malware attacks: Review, taxonomy & future directions, Future Gener. Comput. Syst. 97 (2019) 887–909, http://dx.doi.org/10.1016/j.future.2019.03.007.

[28] G. Shrivastava, P. Kumar, Intent and permission modeling for privacy leakage detection in android, Energy Syst. (2019) http://dx.doi.org/10.1007/s12667-019-00359-7.

[29] developer.android, 2020, https://developer.android.com/guide/topics/permissions/overview. [Online]. Available: https://developer.android.com/guide/topics/permissions/overview.

[30] K. Liu, S. Xu, G. Xu, M. Zhang, D. Sun, H. Liu, A review of Android malware detection approaches based on machine learning, IEEE Access 8 (2020) 124579–124607, http://dx.doi.org/10.1109/ACCESS.2020.3006143.

[31] Y. Pan, X. Ge, C. Fang, Y. Fan, A systematic literature review of android malware detection using static analysis, IEEE Access 8 (2020) 116363–116379, http://dx.doi.org/10.1109/ACCESS.2020.3002842.

[32] A. Feizollah, N.B. Anuar, R. Salleh, G. Suarez-Tangil, S. Furnell, AndroDialysis: Analysis of android intent effectiveness in malware detection, Comput. Secur. 65 (2017) 121–134, http://dx.doi.org/10.1016/j.cose.2016.11.007.

[33] M.A. Jerlin, K. Marimuthu, A new malware detection system using machine learning techniques for API call sequences, J. Appl. Secur. Res. 13 (1) (2018) 45–62, http://dx.doi.org/10.1080/19361610.2018.1387734.

[34] R. Taheri, M. Ghahramani, R. Javidan, M. Shojafar, Z. Pooranian, M. Conti, Similarity-based Android malware detection using hamming distance of static binary features, Future Gener. Comput. Syst. 105 (2020) 230–247, http://dx.doi.org/10.1016/j.future.2019.11.034.

[35] L. Cen, C.S. Gates, L. Si, N. Li, A probabilistic discriminative model for android malware detection with decompiled source code, IEEE Trans. Dependable Secure Comput. 12 (4) (2015) 400–412, http://dx.doi.org/10.1109/TDSC.2014.2355839.

[36] J. Lopes, C. Serrão, L. Nunes, A. Almeida, J. Oliveira, Overview of machine learning methods for android malware identification, in: 7th Int. Symp. Digit. Forensics Secur. ISDFS 2019, 2019, pp. 1–6, http://dx.doi.org/10.1109/ISDFS.2019.8757523.

[37] S.F. Shetu, M. Saifuzzaman, N.N. Moon, S. Sultana, R. Yousuf, Student's Performance Prediction using Data Mining Technique Depending on Overall Academic Status and Environmental Attributes, Vol. 1166, 2021.

[38] O. Qasim, K. Al-Saedi, Malware detection using data mining Naïve Bayesian classification technique with worm dataset, Int. J. Adv. Res. Comput. Commun. Eng. 6 (11) (2017) 211–213, http://dx.doi.org/10.17148/IJARCCE.2017.61131.

[39] M.F.A. Razak, N.B. Anuar, R. Salleh, A. Firdaus, M. Faiz, H.S. Alamri, 'Less Give More': Evaluate and zoning Android applications, Meas. J. Int. Meas. Confed. 133 (2019) 396–411, http://dx.doi.org/10.1016/j.measurement.2018.10.034.

[40] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-An, H. Ye, Significant permission identification for machine-learning-based Android malware detection, IEEE Trans. Ind. Inform. 14 (7) (2018) 3216–3225, http://dx.doi.org/10.1109/TII.2017.2789219.

[41] B.B. Gupta, Computer and Cyber Security Principles, Algorithm, Applications, and Perspectives, 2020.

[42] K. Sharma, B.B. Gupta, Mitigation and risk factor analysis of android applications, Comput. Electr. Eng. 71 (March) (2018) 416–430, http://dx.doi.org/10.1016/j.compeleceng.2018.08.003.