Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 201 (2022) 653–658

Procedia Computer Science

www.elsevier.com/locate/procedia

The 3rd International Workshop on Data-Driven Security (DDSW 2022), March 22 - 25, 2022, Porto, Portugal

An Android Malware Detection Approach Based on Static Feature Analysis Using Machine Learning Algorithms

Ahmed S. Shatnawi[a,*], Qussai Yassen[b], Abdulrahman Yateem[c]

[a]*Department of Computer Information Systems, Jordan University of Since and Technology, Irbid, Jordan*
[b]*Department of Computer Information Systems,Jordan University of Since and Technology, Irbid, Jordan*
[c]*Department of Network and Security Engineering, Jordan University of Since and Technology, Irbid, Jordan*

Abstract

In the past decade, mobile devices became necessary for modern civilization and contributed directly to its development stages in defining mobile information access. Nonetheless, along with these rapid developments in modern mobile devices, security issues rise dramatically, and malware is the most concerning of all. Therefore, many studies and research are still trending in this spectrum, using Machine Learning approaches to prevent and reduce malware's impact. This paper seeks to add to what is already a foundation of various malware detection efforts by presenting a static base classification approach for malware detection based on android permissions and API calls. This approach is based on three well-known Machine Learning algorithms, Support Vector Machines (SVM), K-nearest neighbors (KNN), and Naive Bayes (NB) against a comprehensive new Android malware dataset (CIC InvesAndMal2019), in pursuit of achieving high malware detection rates and contribution to the efforts and studies in protecting the development of mobile information. access.

Keywords: Mobile Information Access; Android Malware; Malware Detection; Machine Learning; SVM; KNN; Naive Bayes; CICInvesAndMal2019 Dataset.

1. Introduction

Since the 1960s, wireless communication has been one of the most vibrant communication domains with its ongoing studies until today [3], leading to mobile computing's evolution as a technology that enabled data and voice and video transmission through wireless-enabled devices [5]. In the past 15 years, mobile communication devices have become one of the essential gadgets of the 21st century. They have exponentially increased and developed from multiple perspectives. These include the number of users worldwide, computational power, storage size, evolving so-

* Ahmed S. Shatnawi. Tel.: +962-791-803-57 ;
E-mail address: ahmedshatnawi@just.edu.jo

1877-0509
10.1016/j.procs.2022.03.086

phisticated functions at lower-cost models, robust operating systems, compatibility and awareness with surroundings, and usability of applications that interconnect with almost every aspect of modern life in every sector of it. Mobile devices have become a necessity of modern civilization. Adding to that, it defined mobile information access.

Nonetheless, along with these rapid developments in modern mobile devices, security issues rise dramatically, and malware is the most concerning of all. By looking at the recent mobile market, Android mobile devices are the most trending brand on top of all smart devices, with more than 2 billion unique users, 74% worldwide market share, and over 300 million yearly smart device sales, branded as the biggest platform for mobile devices [9]. On the other side, iOS platform users are profoundly restricted by the famous iTunes channel for installing desired applications. The Android platform is a flexible OS in installing apps, enabling numerous open sources and direct downloads to install applications in many ways like google play store, torrent files, third-party markets, and others [12]. Consequently, this decentralization and multi-distribution in application installation fertilize a rich environment in assembling and bundling malware, paving a shortcut for attackers to deceive Android users by running their malicious code without noticing. On one side, attackers and malware conductors are already targeting Android operating platforms with intensive malicious applications. On the other side, Android platform developers are pursuing extensive efforts against these malicious apps. Many malware security measures are taken during the installation through the Android system's permission [1].

Although many studies and research are still trending in this spectrum using Machine Learning approaches, which have been widely embraced as primary detection algorithms against malicious attacks, malware families are still growing concerningly over time. More extensive efforts in preventing malware spread are necessary to reflect higher security measurements on the continuity of mobile development towards a more interconnected future. This paper seeks to add to what is already a foundation of various malware detection efforts, magnifying the differences between malware detection approaches and presenting a high-efficiency detection method. Using static analyzing permissions, API calls to detect malware, based on three well-known Machine Learning algorithms, Support Vector Machines (SVM), K-nearest neighbors (KNN), and Naive Bayes. The dataset still has not been extensively experimented enough. Therefore, this paper will contribute to Android malware detection. Moreover, this paper defines and further illustrates the four classes of android malware within the addressed dataset and paves the way to achieve high detection rates by proposing a static analysis-based malware detection method with this recent dataset's aid.

The rest of the paper is organized as follows. Section 2 discusses some of the literature reviews and related work on Android malware detection methods. The chosen dataset is described and discussed in Section 3. Moreover, the experimental environment and setup are illustrated under Section 4, and the applied methodology during this study is explained in Section 5. The results are displayed and compared with previous work under Section 6, and finally, the paper concluded in Section 7.

2. Related Work

As mentioned previously, many studies and research are yet trending using machine learning approaches. One of the latest studies conducting malware detection on the latest CICInvesAndMal2019 dataset is[11]. The study performs a static base classification applying principal component analysis as a feature selection approach through a feature set for malware detection containing android permissions and intent—the study trains and testes the chosen dataset over several machine learning models. The Random Forest was the best classifier applied to deliver an accuracy rate of 96.05% when the dataset features were degraded to avoid any overheads. The experiment also reported an 88.23% accuracy rate using Naive Bayes, 91.26% accuracy rate using SVM, and 93.88% accuracy rate using KNN and only covered the permission features. Among other static base classification studies is [6], presenting a novel approach for a permission-based android malware detection system that depends upon static investigation. The proposed system is based on Logistic Regression and Decision Tree algorithms and consists of three parts, client, server, and a signature database. In this study, the server plays an essential role in the whole analysis process, warning the device user if the desired application is malicious or not by detecting the label or category of the application and accordingly classifying them, where results show 95% accuracy rate and 4% false-positive ratio. Another static analysis was performed in [16]. The authors examined RFA, J48 DT, and CRT algorithms to analyze 500 Android APK files by inspecting the application's permissions and using 160 permissions as a feature vector. Nevertheless, the experiment

lacks insight regarding the parameter settings applied for the Random Forest algorithm and does not seem clear about the amount of benign and malicious applications (malapps) used.

Another machine learning model presented in [4] uses several distinct properties to counteract and detect malware. The study was done using Rain Forest, Bayesian Network, and KNN for detection purposes and collected a variety of 14 static features, including frequencies of opcodes with their sequences. Among the findings on the benign applications was an opcode (RSUB_INT). The experiment results show about 96% accuracy by applying the Random Forest classifier against a dataset containing 10,000 applications. Among them, 5,000 were malicious samples, and the other 5,000 were benign ones. The studies also show that the ideal way to predict any malapp hostility is by android permissions reaching about 96% accuracy with the Random Forest classifier.

Another recent hybrid approach presented in [13], this malware detection model based on a Tree Augmented Naive Bayes (TAN), uses conditional dependencies against both static and dynamic features such as API permissions and system calls. It then detects malicious behavior by combining the outputs of both these features gained from the classifier. Although the model shows 97% accuracy, it does not show information regarding the android version or user input generated during dynamic analysis. In [15], a different hybrid detection system is proposed. The model uses SVM and a linear classifier based on a new open-source framework called CuckooDroid. The framework enables Cuckoo Sandbox's features by a misuse detector to detect well-known malware and classify android malware by combining static and dynamic analysis. The proposed model benefits from the low false-positive rate of misuse detection and the ability of anomaly detection to detect zero-day malware and is evaluated with 5560 malware and 12000 benign. This model achieved 98.79% accuracy detection rates through classifying 98.32% malware samples.

3. Dataset

This study chooses the (CICInvesAndMal2019) [14] dataset to complete the experiment. As a recent publicly available dataset and includes (permissions, intents, and API calls as static features). All the log files generated in the dataset are dynamic features and are done over three levels. The first is During installation, the second is before restarting, and the third step is after restarting the phone. Furthermore, compared to its previous version, the dataset includes an enhanced malware category and contains improved malware family classification performance. Moreover, it provides additional features captured, such as battery information, log events, packages, process logs, and more. It consists of a total of (426) malware and (5,065) benign; among them are static features such as permission and intents of (1522) applications which consists of (396) malware and (1126) benign applications.

4. EXPERIMENTS

Evaluations were conducted on a Window 10 2.6GHz Intel Core i7-9750H CPU with 16.0GB physical memory, using PyCharm and coding the required functions to run the three determined models, Support Vector Machines (SVM), K-nearest neighbors (KNN), and Naive Bayes (NB) using Python, including downloading two essential tools to help interoperate with the code. Like most recent machine learning research in this field, the first tool was Scikit-learn, a Python built software dedicated to (ML) libraries. It is mainly designed to interact with the Python mathematical and logical libraries and maintains several main classifications and regressions. It also includes clustering algorithms, such as (SVM), random forests, gradient boosting, k-means, etc. [8]. The second tool was Pandas, where this software library is also used for Python, and practically for data manipulation and analysis. Due to its aiding capabilities in manipulating numerical tables. It is essential in structuring data and time-series operations and datasets that include observations over multiple periods [7].

5. METHODOLOGY

In this section, the applied methodology in this paper is shown in Fig. 1. and explained. As for the first phase, data pre-processing was held to check the data for any incomplete attribute values or lacking specific attribute interests. Also, checking if the dataset contains any errors, outliers, or any noise and inconsistency. In our case, no such missing values were excluded or omitted, and no null values were detected in the choosing dataset. The second phase was

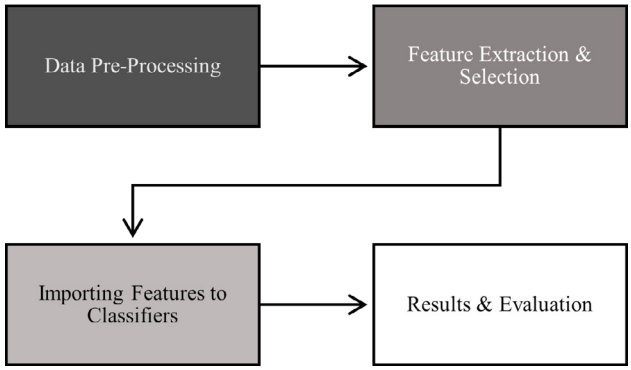


Fig. 1: A visualization of the ML classifiers accuracy rates using the permission features

choosing the Android API calls and permission features from the dataset, separating both feature sets distinguishably. Therefore, classifying the total (8111) features obtained from the training dataset and separating them into two datasets distinguishing (2089) permission features and (6022) API calls features. This was done by reading the training dataset and differentiating between each column containing (com) and inserting it to the API call feature dataset; those columns not containing are added to a permission feature dataset. Furthermore, features datasets were obtained from their paths, and taking the goal column that consists of the class of the traffic or permissions, for either malware or normal, and dropping it to ensure it is not used during the training process—then naming the storing list where the column titles will be stored (column name). Next and within the second phase, applying feature importance selection using the Recursive Feature Elimination (RFE) on the Logistic Regression model. It is a well-known feature selection algorithm due to its simple configuration and utilization; it is an effective method for selecting training dataset features most applicable in terms of target variable prediction. The logistic regression classification model is based upon classifying the dataset features to either malware or benign and ranking the essential features to represent the probability of a (0 or 1) variable to distinguish the vital feature from the least important.

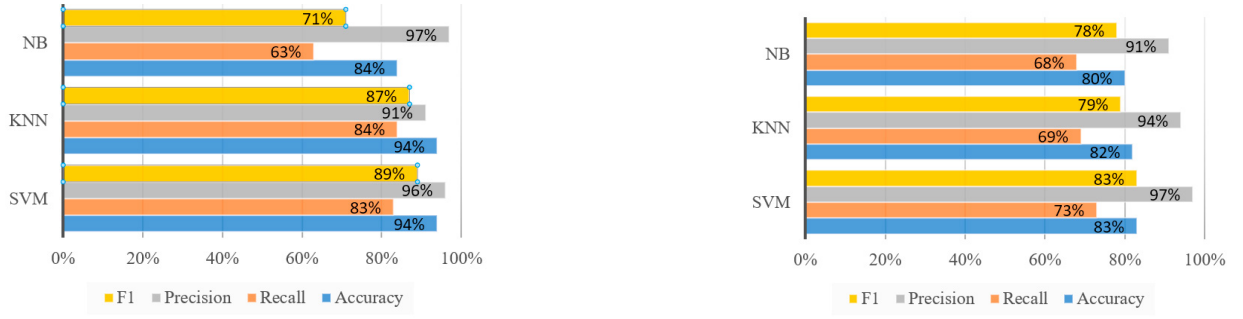
Logistic regression becomes a classification technique only if applied with a decision threshold. Therefore, (RFE) takes the Logistic Regression model's data, trains it based on a set of predictors, then computes an importance score for each predictor and compares the scoring ranking results from the Logistic Regression model. It then removes the least important predictors and rebuilds the model to compute the importance scores again [2]. Therefore, the RFE will choose the most important features based on the Logistic Regression model's scoring ranking to do feature ranking based on setting the number of essential features to be ranked and then fitting (training) the data. Then, store the column names to the (column name) list and print the number of features chosen to be ranked and the ranked features' names. As for the testing scheme, the holdout method was done. Moreover, the data were divided into test and train datasets for validation. The test takes 20% of the dataset, and the train takes 80%. The data is split between X_train and y_trainṪhe (tar) is split between X_test and y_test. X_train contains the features to train (features), y_train contains the features column that consists of malware and normal (features), X_test is the column of the features in the X_train be tested, and the y_test is used to test the validation accuracy. The third phase is importing the selected important features to the ML classifiers. The SVC will then split the data into two portions and classify the data based on this separation. Next, fitting (training) the SVM classifier on the X_train (row) and y_train (column). After training the data, the predict function is called to test data that was split to 20% X_test, to output values that should more or less equal the y_test, which are the actual data values that consist of the column that represents every row in the X_test if it was malware or normal based on the class opposite to each row. The expected predicted results should more or less equal the y_test. To ensure the expected predicted results should be more or less equal to the y_test, the accuracy, recall score, precision score, and f1 score are inspected based on the y_test and the predicted value. Similarly, after importing the KNN classifier and setting the number of neighbors to 7, if the new data is closer to seven times the nearest than the previous data, it is considered within the nearest range of either malware or normal features. Then, fitting the KNN classifier and predicting the test data split to 20% to output values that should more or less equal the y_test, as done with the SVM classifier. To ensure expected predicted results should be more or less equal to the y_test,

the accuracy, recall score, precision score, and f1 score are inspected based on the y_test and the predicted value. The same procedures are applied to the GaussianNB. Then finally, displaying the results in the final phase.

6. RESULTS

This section presents the results comparing the chosen classifiers' efficiencies in malware detection accuracy based on both the permission and API call features. After the environmental setup was prepared for feature extraction and code execution, the experiment first started with the process of separating the API calls and permission features from the training dataset into separated datasets and settling on (50) important features from the ranking process on both sets after multiple extensive runs and experiments. Moreover, observing the classifiers' behaviors through its evaluation measurements and choosing the most promising accuracy rates among these multi-observation runs.

The malware detection based on the permission features results are represented in Fig. 2a. The (SVM) and (KNN) classifiers have the highest detection accuracy rates in this experiment. A closer look at the results exhibits that the (SVM), among the other classifiers, scored an 88.75% F1 rate against the 87.27% rate delivered by (KNN) in the same evaluation. On the other hand, Naïve Bayes had an 84.33% accuracy rate and achieved a 70.81% F1 score. Therefore, the results show that the (SVM) outperforms all classifiers in terms of almost all evaluation measurements.



(a) Permission features

(b) API call features

Fig. 2: A visualization of the ML classifiers accuracy rates using permission features and API call features.

As for the malware detection results based on the API call features represented in Fig. 2b, it shows that both the (SVM) and (KNN) classifiers have higher detection rates than the Naïve Bayes classifier accuracy rate. The Naïve Bayes classifier has slightly lower rates on this dataset then the (KNN). Moreover, the (SVM) outperforms all classifiers in terms of all evaluation measurements classifying this data feature set.

Comparing our proposed work with one of the latest studies referred to in [11]. The study applied five classifiers in detecting malware permission features but did not cover malware API call features in the detection process. As for our study, we covered both, where the (SVM) classifier delivered the highest rates in this experiment, reaching an average of 94% accuracy rate using permission features and almost 83% accuracy rate using the API call features. Additionally, in terms of comparing results obtained based on permission feature detection as shown in Table 1. The comparison shows that our applied methodology outcomes a higher accuracy rate with the (SVM) classifier. It delivers a 94.36% accuracy rate and an 88.75% F1 score compared with the referred study that achieved 91.26% accuracy rate and a 90.8% F1 score. Our classification results are less by an average of 4% in terms of the Naïve Bayes accuracy rate, where the compared study delivered an average of 88.23%. Besides, both studies deliver adjacent touching results using the (KNN) classifier, where our results reveal a 93.42% accuracy rate against 93.88% in the compared study. Furthermore, the Principal Component Analysis (PCA) adopted in the compared study is recognized as an unsupervised technique, and the used dataset is labeled. (PCA) is known for having some limitations in distributing classes against small datasets [10]. As a process of computing principal components and utilizing them to produce a change of basis on a chosen data, using only the first few principal components and neglecting what remains.

Table 1: RESULT COMPARISON WITH PREVIOUS WORK IN TERMS OF ANDROID MALWARE DETECTION USING PERMISSION FEATURES.

Our Work Vs. Previous Work [8]								
Classifier	**Accuracy**		**Recall Score**		**Precision Score**		**F1 Score**	
SVM	94.36%	91.26%	82.6%	91.3%	95.9%	91.2%	88.8%	90.8%
KNN	93.42%	93.88%	83.7%	93.9%	91.1%	93.8%	87.3%	93.8%
NB	84.33%	88.23%	63.0%	88.2%	97.4%	87.7%	70.8%	87.7%

Therefore, in consideration of static analysis and applying (PCA) accordingly, the static features chosen from the dataset will most likely return principal components that signify combinations of features, not actual features, integrating and reducing the number of features. To that extent and in our belief (PCA) is hard to give an exact outcome of the essential features. Consequently, in our study, we have applied the (RFE) feature selection method to effectively select the most effective training dataset features in terms of target variable prediction.

7. CONCLUSION

This study adds to the foundation of various malware detection efforts by proposing a static base classification method for Android malware detection on permission and API call features from the (CIC InvesAndMal2019) dataset. This approach, based on three well-known Machine Learning algorithms (SVM), (KNN), and (NB), shows that the (SVM) classifier had the highest rates among the other compared classifiers. It delivered an average of 94% accuracy rate using permission features and an 83% accuracy rate using the API call features in pursuit to achieve high malware detection rates and contribute to the efforts and studies in protecting the development of mobile information access.

References

[1] Alfalqi, K., Alghamdi, R., Waqdan, M., 2015. Android platform malware analysis. International Journal of Advanced Computer Science and Applications (IJACSA) .
[2] Escanilla, N.S., Hellerstein, L., Kleiman, R., Kuang, Z., Shull, J., Page, D., 2018. Recursive feature elimination by sensitivity testing, in: 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), IEEE. pp. 40–47.
[3] Goldsmith, A., 2005. Wireless communications. Cambridge university press.
[4] Hahn, S., Protsenko, M., Müller, T., 2016. Comparative evaluation of machine learning-based malware detection on android. Sicherheit 2016-Sicherheit, Schutz und Zuverlässigkeit .
[5] Johnson, S., Twilley, N., Zhang, T., Zhou, Z., Wu, S., 2014. Mobile computing: A look at concepts, problems and solutions.
[6] Magdum, M., Wagh, S.K., 2016. Permission based android malware detection system using machine learning approach. International Journal of Computer science and Information Security 14, 465.
[7] McKinney, W., et al., 2011. pandas: a foundational python library for data analysis and statistics. Python for high performance and scientific computing 14, 1–9.
[8] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al., 2011. Scikit-learn: Machine learning in python. the Journal of machine Learning research 12, 2825–2830.
[9] Protalinski, E., 2017. Android passes 2 billion monthly active devices — venturebeat. https://venturebeat.com/2017/05/17/android-passes-2-billion-monthly-active-devices/. (Accessed on 11/11/2021).
[10] Ravichandran, S., Ramasamy, C., 2016. Performance comparison of dimensionality reduction methods using mcdr. Training 3.
[11] Sangal, A., Verma, H.K., 2020. A static feature selection-based android malware detection using machine learning techniques, in: 2020 International conference on smart electronics and communication (ICOSEC), IEEE. pp. 48–51.
[12] Saudi, M.M., Abdullah, Z.H., 2013. An efficient framework to build up malware dataset, in: Proceedings of World Academy of Science, Engineering and Technology, World Academy of Science, Engineering and Technology (WASET). p. 534.
[13] Surendran, R., Thomas, T., Emmanuel, S., 2020. A tan based hybrid model for android malware detection. Journal of Information Security and Applications 54, 102483.
[14] Taheri, L., Kadir, A.F.A., Lashkari, A.H., 2019. Extensible android malware detection and family classification using network-flows and api-calls, in: 2019 International Carnahan Conference on Security Technology (ICCST), IEEE. pp. 1–8.
[15] Wang, X., Yang, Y., Zeng, Y., Tang, C., Shi, J., Xu, K., 2015. A novel hybrid mobile malware detection system integrating anomaly detection with misuse detection, in: Proceedings of the 6th International Workshop on Mobile Cloud Computing and Services, pp. 15–22.
[16] Zarni Aung, W.Z., 2013. Permission-based android malware detection. International Journal of Scientific & Technology Research 2, 228–234.