CyberSecurity and Digital Forensics e-ISSN: 2615-8442
Vol. 5, No. 1, May 2022, p.1-5

STATISTICAL ANALYSIS OF ANDROID MALWARE DETECTION USING SUPERVISED MACHINE LEARNING ALGORITHM

Raden Budiarto Hadiprakoso[1], Wahyu Rendra Aditya [2], Febriora Nevia Pramitha [3]

[1,2,3] Politeknik Siber dan Sandi Negara
Email: [1]raden.budiarto@poltekssn.ac.id, [2]wahyu.rendra@poltekssn.ac.id, [3]febriora.nevia@poltekssn.ac.id

Abstract

Currently, the growth of the Android operating system on smartphone devices is popular. However, behind this popularity, the Android platform is also a potential target for cybercrimes against cybersecurity threats such as malware. Identifying this malware is critical to maintaining user security and privacy. Because the malware identification process is getting more complicated, it is necessary to use machine learning for malware classification. This study collects the static analysis features of safe and malicious applications. (malware). The dataset used in this study is a DREBIN malware dataset which is a publicly available malware dataset. The dataset consists of the CALL API features, system commands, manifest permissions, and Intents. The data is then processed using various supervised machine learning algorithms including Support Vector Machine (SVM), Naive Bayes, Decision Tree and K-Nearest Neighbors. We also concentrate on maximizing performance by evaluating various algorithms and adjusting some configurations to get the best combination of hyper-parameters. The experimental results show that the SVM model classification gets the best results by achieving an accuracy of 96.94% and an AUC (Area Under Curve) value of 95%.

Keywords*: android, malware, machine learning, malware detection, static analysis*

1. INTRODUCTION

The development of information technology today has brought many advances in various fields, one of which is in mobile devices, namely smartphones. A smartphone basically provides applications and an operating system to facilitate its users. Some of the operating systems used include iOS, Blackberry OS, and Android. Of the several existing operating systems, Android is one of the most widely used operating systems. According to Google CEO, Erich Schmidt, around 1.4 million Androids are activated every day(cycles, 2021).

As of mid-2020, the Android platform represents 70.61% of the total mobile operating system market based on NetMarketShare, which makes it the most used mobile operating system (Stat, 2021). However, the popularity of Android does not always have a positive impact on its users. Even with various protection mechanisms such as Play Store Protection, there are still various reports of malware presence in the Play Store(Alzaylaee, Yerima & Sezer, 2020)..

This widely used Android mobile operating systemcan become the target of crime. According to the McAfee Mobile Threat Report, there is a new type of malware called PhantomLance, LeifAccess or Shopper, which was only identified in May 2020 and has been active globally, mainly in the United States and Brazil with a total of 943 and 286 attacks respectively. According to Pavel Shoshin, PhantomLance is a backdoor Trojan for Android on Google Play(Qiu et al., 2020). The statement shows that even companies like Google, which are often at the forefront of keeping the Android ecosystem free of malware, have failed to take action to prevent malware.(Ma et al., 2019). From the statement above, Android malware is getting more and more sophisticated to circumvent security and infiltrate the Google Play Store. This malware will most likely spread to Android devices. Due to the increasing sophistication of malware and difficult to detect, it is important to do malware detection. Many researchers are trying to mitigate cyberattacks on malware through various approaches(Odusami et al., 2018).

Various malware detection techniques were developed to counter this expansion. Among the most common detection systems is static analysis(Taheri et al., 2020). Through an analytical approachstatic, malware is recognized as a signature extracted from an application. Usually, this signature comes from the payload. When the signature is recognized, it will be stored in the database and used to classify new malware cases based on that database.(McLaughlin et al., 2017).

On the other hand dynamic analysis is a procedure in which malware instances are examined in real-time. When running in a virtual environment such as an emulator or sandbox, malicious program activity is observed. Anti-virus programtrying to find if actions like file replication, duplication, or even impersonation were performed. After the code is decompiled, comparisons are made with known malware code to determine if the application is malicious. The main drawback of this method is the inability to find viruses that use new procedures to carry out malicious activities and are inefficient in terms of time and resources(Wang, Zhao and Wang, 2019).

For this research, we have focused on static analysis for malware classification. Static analysis is a quick and easy method of identificationmalware without running the application or observing run-time behavior(Ma et al., 2019). This study proposes a machine learning algorithm to be used in malware detection by comparing several algorithms. Four algorithms will be compared, namely Support Vector Machine (SVM), Naive Bayes, Decision Tree, K-Nearest Neighbor (KNN).

2. LITERATURE REVIEW

For a few years, Machine Learning (ML) algorithms were tasked with developing intelligent systems by training machines to make decisions. With datasets asinput and classifier as a method, ML is able to identify new data that has similarities. There are many ML algorithms that can be used to build ML frameworks or models and each algorithm has its own advantages depending on the dataset and features used.

(Wang, Zhao and Wang, 2019)in his research, comparing the ML classifier included in the Supervised group. Supervised Machine Learning is a search algorithm that utilizes external instances to generate general hypotheses, which are then used to predict eventsThe future ML algorithms used in this study include Decision Tables, Random Forests (RF), Naïve Bayes (NB), Support Vector Machines (SVM), JRip, and Decision Trees (J48) and use machine learning tools Waikato Environment for Knowledge Analysis (WEKA). The result is that SVM is an algorithm that has the best accuracy and precision.

in 2018(Fan et al., 2018), conducted other research on malware detection. (McLaughlin et al., 2017) Fan built a malware detection framework which included six machine learning classifiers, SVM, Decision Tree (C4.5), MLP, NB, K-KN and Bagging predictor. To calculate the ML performance of the algorithm, Chen divides it into two categories: the first category contains permission features and sensitive API calls and the second category contains permission, sensitive features, API calls, sequences and dynamic behavior. In the first category the best performance was obtained by K-NN and MLP with average accuracy reaching above 91.00%. Whereas in the second category, SVM and K-NN obtained the highest accuracy with an average value of 93.80% and 93.80%. In terms of execution time all algorithms run under one minute, except (MLP) which with its multi-layer takes longer. In general, in this study, K-NN is the classifier with the best performance that requires small computational resources.

In 2020, research was conducted by(Zhang et al., 2020)malware detection based on machine learning using

dynamic analysis on android applications. In the end, created an application, to extractthe information contained in an android application. In this application several classification algorithms are trained to see the best performance in terms of accuracy and speed. In this application, it was found that the research obtained an accuracy value of 97% for detecting invisible malware from the data that had been prepared.

In research conducted by(Feng et al., 2018), performs a dynamic analysis study which uses syscall-capture to capture and analyze the behavior of system-call traces made by each application during its execution. In the end, an accuracy rate of 85% was obtained using the decision tree algorithm and an accuracy rate of 88% using the RF algorithm.

In research conducted by(Lashkari et al., 2018), malware detection is carried out on smartphones that use the Android operating system. This research includes a discussion on how to develop a malware detection system that can detect various types of malware, how to detect malware before the installation process through several stages. The first step is to extract the strings from the android application and explore the android manifest.xml file. the second step is separating the string keywords from the android manifest.xml. The third stage is classifying malware and legitimate applications using keywords and strings as input features. The final stage is identifying dangerous applications and safe applications.

Overall, what differentiates this research from previous studies is that we used a static analysis approach. This approach was chosen because the detection process is fast and lighter on memory considering the implementation environment on mobile.

3. RESEARCH METHODS

This research method is divided into several stages as depicted in Figure 1 which include research data collection, data processing, classification testing, and algorithm comparisons. The following is an overview of each of these stages.

The first stage is data collection. We collect datasets from the malware DREBIN project(Arp et al., 2014). The data contains 215 feature attributes extracted from 3799 applications (1260 malware applications and 2539 benign applications). The feature attributes in the dataset consist of static analysis attributes in the form of signatures from the application. The dataset contains static attributes like manifest permissions, command signatures, intents. Besides that, there is also an API call attribute which is the result of dynamic analysis. This dataset will be used as training data and test data for machine learning algorithms.

The second stage is data processing. The existing dataset is then divided into training data and test data. The proportion of training data and test data is 80% for training data or 3039 data and 20% for validation data or as many as 760 data. One column represents one feature and its frequency, while one row represents one android application, and the last column is class type (1 for malware applications and 0 for tame applications). The data is divided by means of cross validation, in which the data is divided equally for training and testing without overlapping each other. In this study we used 5-fold cross validation.

The third stage is classification testing. In this study, the data will be used by the SVM, NB, Decision Tree, and K-NN algorithms. Each algorithm will use training data in the same proportion, namely 80% for training data and 20% for test data.

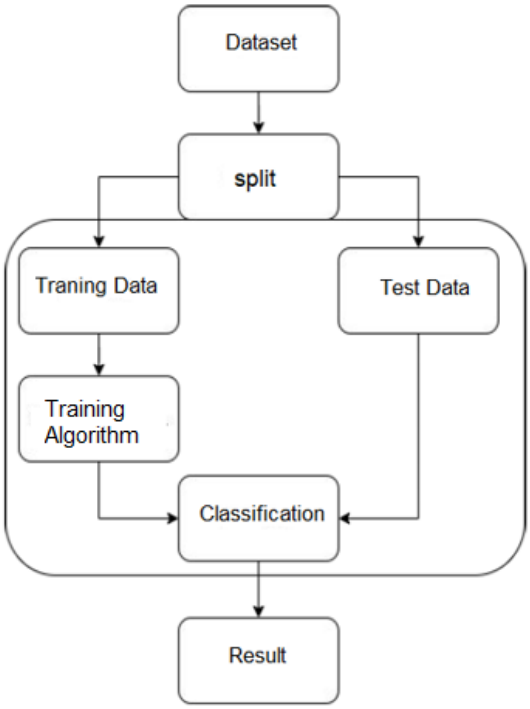

Figure 1. Research flow

The following is an overview of the process at this stage:

1. The training data was obtained from random separation of the dataset which took 80% of the entire dataset.
2. Random separation test data obtained from a dataset taken 20% of the entire dataset.
3. Algorithm training is the process of training a classification algorithm with training data.
4. Classification at this stage is tested between training data and test data, by calculating the classification accuracy. The result is the level of accuracy in classifying applications.

The next step is the comparison of the algorithms. The result of the third stage is accuracy in predicting applications, including malware or not. The level of accuracy of the SVM, NB, Decision Tree, and K-NN algorithms will be compared. From this comparison, it is obtained that the algorithm is considered more effective in classifying applications, whether it is malware or not.

4. RESULTS AND DISCUSSION

Once the data is collected, we use Google Colab to process it. We used a test environment with an i3-2328M processor with 10GB of RAM: At the data preprocessing stage using the SMOTE technique, we obtained the same number of sample classes, namely 2539 in each class. Stratified 5-fold

cross-validation divides the data equally with 80% of the training data, and the remainder goes to validation. The techniques used for training and validation are SVM, K-NN, Decision Tree and NB. Table 1 describes the average training accuracy and latency results.

Table 1. Test results

Algorithm	accuracy	Latency
SVM	0.969471	4,497
Naive Bayes	0.738667	1,247
DecisionTree	0.923684	2,754
K-NN	0.928667	1,549

From the accuracy table above, it is known that the SVM algorithm has an average accuracy of 96.94% which is also the highest value compared to the other three algorithms. In addition, the KNN algorithm also has an average accuracy that is close to the average accuracy of the SVM algorithm.

SVM shows the best malware detection results. SVM shows the best results here presumably because the dataset is not too big, and the data is clean from noise. SVM tends to work better on data that is not too large and free of noise. Another factor is the feature size of the dataset which is quite large with 215 features because SVM is more effective for datasets with large dimensions.

AlgorithmNB does not produce high accuracy scores because it tends to work more effectively with less training data. This algorithm also has the shortest latency time because the training data set is only stored in memory and reused during prediction. This results in short training times but longer testing times. The same way of working also applies to the K-NN algorithm. Therefore, this algorithm has a relatively short training time. The K-NN algorithm shows quite good results, this is due to the distribution of non-linear data because this algorithm is known for classifying non-linear data.

The results of the next test are shown in Figure 2 which is the ROC (Receiver Operating Characteristics) curve of the SVM model. The ROC curve is used to assess the performance of a classification problem. Area Under Curve (AUC) is the size of the area under the ROC curve, the wider this area indicates the better the proposed classification model. ROC is a graphical representation of the relationship between sensitivity and specificity. This value indicates how well the model's ability to separate classes. Figure 2 depicts a fairly wide AUC of around 95%. This value indicates that the resulting model is suitable for input categorization. In addition, Figure 2 also shows that the AUC area in the training and validation dataset is relatively stable. These results indicate that the proposed model is fit.

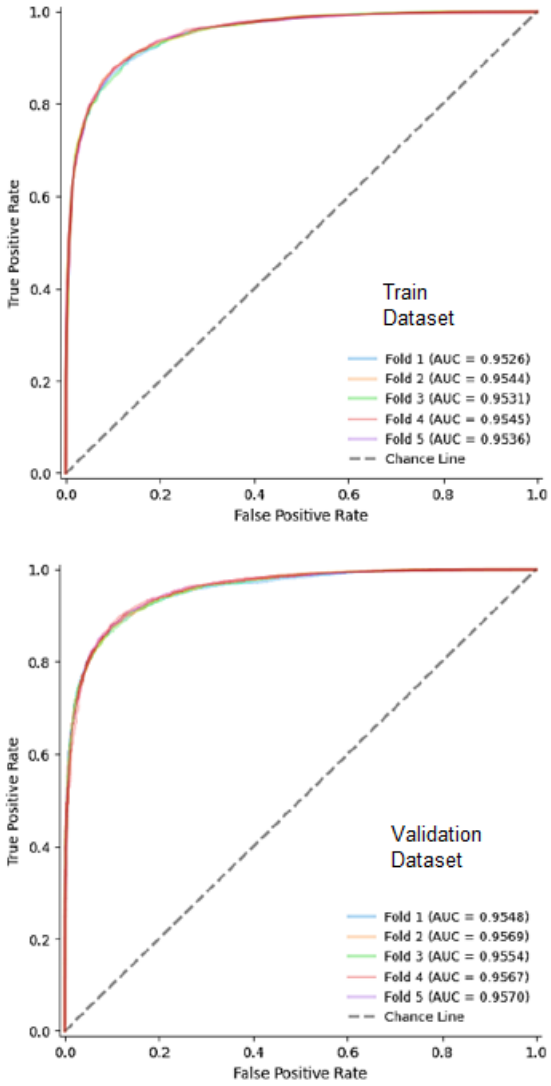

Figure 2. ROC curve on the SVM model

5. CONCLUSIONS AND RECOMMENDATIONS

The popularity of the Android operating system has made it a target for many crimes, such as malware. Various malware detection engines are emerging with various static or dynamic detection methods and machine learning algorithms. In this study, malware detection uses static analysis methods and machine learning algorithms. The results shown prove that our model provides a high accuracy of 96.94% using the SVM algorithm. The results of the ROC curve test also show that the model has an AUC area of around 95%. For further research development, deep learning algorithms can be investigated to improve malware detection capabilities on the Android platform.

BIBLIOGRAPHY

Alzaylaee, M.K., Yerima, S.Y. and Sezer, S., 2020. DL-Droid: Deep learning based android malware detection using real devices. Computers & Security, 89, p.101663.

Arp, D., Spreitzenbarth, M., Hübner, M., Gascon, H. and Rieck, K., 2014. DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket. Symposium on Network and Distributed System Security (NDSS). . https://doi.org/10.14722/ndss.2014.23247.

Cycles, D., 2021. Topic: Android. [online] Statista. Available at: <https://www.statista.com/topics/876/android/> [Accessed 11 Oct. 2021].

Fan, M., Liu, J., Luo, X., Chen, K., Tian, Z., Zheng, Q. and Liu, T., 2018. Android malware familial classification and representative sample selection via frequent subgraph analysis. IEEE Transactions on Information Forensics and Security, 13(8), pp.1890–1905.

Feng, P., Ma, J., Sun, C., Xu, X. and Ma, Y., 2018. A novel dynamic Android malware detection system with ensemble learning. IEEE Access, 6, pp.30996–31011.

Lashkari, A.H., Kadir, A.F.A., Taheri, L. and Ghorbani, A.A., 2018. Toward developing a systematic approach to generate benchmark android malware datasets and classification. In: 2018 International Carnahan Conference on Security Technology (ICCST). IEEE.pp.1–7.

Ma, Z., Ge, H., Liu, Y., Zhao, M. and Ma, J., 2019. A combination method for android malware detection based on control flow graphs and machine learning algorithms. IEEE access, 7, pp.21235–21245.

McLaughlin, N., Martinez del Rincon, J., Kang, B., Yerima, S., Miller, P., Sezer, S., Safaei, Y., Trickel, E., Zhao, Z. and Doupé, A., 2017. Deep android malware detection. In: Proceedings of the seventh ACM on conference on data and application security and privacy. pp.301–308.

Odusami, M., Abayomi-Alli, O., Misra, S., Shobayo, O., Damasevicius, R. and Maskeliunas, R., 2018. Android malware detection: A survey. In: International conference on applied informatics. Springer.pp.255–266.

Qiu, J., Zhang, J., Luo, W., Pan, L., Nepal, S. and Xiang, Y., 2020. A survey of Android malware detection with deep neural models. ACM Computing Surveys (CSUR), 53(6), pp.1–36.

Taheri, R., Ghahramani, M., Javidan, R., Shojafar, M., Pooranian, Z. and Conti, M., 2020. Similarity-based Android malware detection using Hamming distance of static binary features. Future Generation Computer Systems, 105, pp.230–247.

Wang, W., Zhao, M. and Wang, J., 2019. Effective android malware detection with a hybrid model based on deep autoencoder and convolutional neural network. Journal of Ambient Intelligence and Humanized Computing, 10(8), pp.3035–3043.

Zhang, X., Zhang, Y., Zhong, M., Ding, D., Cao, Y., Zhang, Y., Zhang, M. and Yang, M., 2020. Enhancing state-of-the-art classifiers with API semantics to detect evolved android malware. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. pp.757–770.

.