ICONTECH INTERNATIONAL JOURNAL OF SURVEYS, ENGINEERING, TECHNOLOGY
ISSN 2717-7270
Journal homepage: http://icontechjournal.com/index.php/iij

K-NEAREST NEIGHBOUR CLASSIFIER USAGE FOR PERMISSION BASED MALWARE DETECTION IN ANDROID

Dr. Recep Sinan ARSLAN

Yozgat Bozok University, Vocational High School of Technical Sciences

Assist. Prof. Dr. Ahmet Hasim YURTTAKAL

Yozgat Bozok University, Vocational High School of Technical Sciences

ARTICLE INFO

Keywords
Malware Detection
Classification
Machine Learning
Android
Mobile Security
KNN

ABSTRACT

Android application platform is making rapid progress in these days. This development has made it the target of malicious application developers. This situation provides a numerical increase in malware apps, diversity in techniques, and rise of damage. Therefore, it is very critical to detect these software and escalation the security of mobile users. Static and dynamic analysis, behaviour scrutiny, machine learning methods are used to ensure security. In this study, K-nearest Neighbourhood (KNN) classifier, one of the machine learning methods, is used. Thus, it is aimed to detect malignant mobile software successfully and quickly. The tests is conducted with dataset includes 492 malware and 697 benign applications. In the proposed algorithm, neighbour number 5 and distance metric is preferred as Minkowski. 80% of dataset randomly selected is reserved for training and 20% for testing. As a result, while 94.1% accuracy is achieved, precision 91.2%, recall 92.7% recall and f1-measure is 92.4%. The high value obtained in f1-measure shows that the proposed model is successful in detecting both malware and benevolent software. The success of using KNN algorithm in classification of malicious apps in the Android has been demonstrated.

INTRODUCTION

Today mobile devices are more widely used than personal computers. The providing factors of this usage have been the convenience of the devices with their developing capacity and power, and portability. It is estimated that the number of mobile devices has increased by nearly twice as compared to 2014 [1]. Android operating system is one of the commonly used mobile OS leading this raise. This open source software managed by Google, is used in almost half of all mobile phones.

The large user base has made mobile tools using Android OS the target of malicious cybercriminals. The proof of this situation is the high number of malwares, uploaded and published on Android application markets [2]. Besides, the variety of malicious software has

Received 8 July 2020; Received in revised form 15 July 2020; Accepted 25 Julya 2020;
Available online 15 September 2020
doi: 10.46291/ICONTECHvol4iss2pp15-27

been constantly arising in the Android mobile networks, which poses a risk to end users. Unlike the IOS, users can download the applications from shared files in third-party distribution environments as well as the Play Store [1]. Malevolent application detection tools are needed to help Android users cope with these security problems. In order to develop these detection engines static, dynamic and hybrid analysis methods are preferred.

Static analysis method is one of the main techniques used by commercial antivirus programs. In this approach, it is aimed to examine the code and structure of applications without having to install them on any mobile device. Execution schemes in the code are generated. Thus, it is determined whether the purpose of the application is to perform malicious activities. In order to carry out this process statically, application source code need to be obtained with reverse engineering methods. This conversion was easy in the applications implemented with old API versions. However, reverse engineering is a difficult process due to modern compilers and libraries used in newly developed applications today. Moreover, many encryption, obfuscation and hiding code techniques that can be used by developers to prevent accessing original java and xml codes[3].

Dynamic analysis approaches have been proposed in order to overcome the shortcomings of the static analysis [4-6]. The applications are installed and run in a sandbox or real environment in this method. Their behaviour is monitored during the working process. It is aimed to catch API calls, compare them with known actions and detect the malwares. Hiding or encryption techniques are not problem. In addition to being a successful approach in the examination and classification of apps, it has some problems. The main ones are the necessity to install and run on a mobile device, keeping track of running processes reduces the system performance and demands serious pre-processing stages. Hybrid approaches are recommended in which the advantageous points of static and dynamic analysis are used [7-9]. Thus more outstanding results were achieved.

Mobile gadgets have limited resources due to their physical size. Accordingly, it is expected that the suggested malignant software detection software does not need much processing time. Therefore, machine learning techniques are widely preferred in these days and applied in certain types of decision making tasks. High success rate can be achieved for some problems [10, 11]. In the paper numbered by [12] conducted by Sunita et al, different classification algorithms such as KNN, Naive Bayes, SVM, J48 are used to detect malignant software.

In this study, a model has been developed for acquiring features of mobile applications with the static code analysis and classifying them with KNN algorithm. Malicious software is detected using machine learning, and the following contributions are given to the literature.

- Using proposed architecture, it is possible to specify malwares with an accuracy of 90% and above.
- In order to be used in model training and testing, 697 benign and 492 malicious mobile software is collected and an up-to-date and genuine dataset is created.

Received 8 July 2020; Received in revised form 15 July 2020; Accepted 25 Julya 2020;
Available online 15 September 2020
doi: 10.46291/ICONTECHvol4iss2pp15-27

- Using the KNN classifier, low cost and without the need for extra processing time, rewarding results can be attained quickly.
- When comparing similar studies, better results were obtained with a relatively limited but original dataset.
- Experiments is repeated and to prove reliability of the outcomes and it is shown that a certain level of performance is achieved in all cases.

In the second part of this study, a literature review consist of the studies using KNN and other classification algorithms, is given. Next, in the Methodology section, data acquisition, getting relevant features from the Android apk files and classification process with KNN are discussed in detail. In the 4th part, the outcomes in the tests performed with the 1189 apps have been show exhaustively and evaluated comparatively with similar studies. Finally, In the light of the results, an overall evaluation and suggestions about the future are mentioned.

RELATED WORKS

There are many studies about detecting malware in the Android OS. One of the first determination regarding the weaknesses in Android and the malicious software that started to appear rapidly was made by Steve Manstifield [12]. In this research, it is predicted that the used of Android is becoming widespread, which will make it a target of cyber attacks. In addition, it is stated that Android architecture offers a good opportunity for malignant people due to its open source distribution and technical and procedural defects. Accordingly, it is explained that models that can protect users against malicious software should be urgently recommended and developed.

In the study prepared by Mark et al in 2013 [13], a model in client-server architecture has been proposed that can be used to detect malicious apps installed on Android devices. In this model, the usage statistics of malignant programs are evaluated in a remote server system and the intention of the application is tried to be determined accordingly. It is a type of dynamic and behaviour analysis. Similarly, in the study evaluating the behaviours of the applications after installation [33], to identify malicious attacks, an assessment based on popular malicious practices in this areas has been used.

Due to the increasing number of malignant application in 2016, working on detection models of malwares has accelerated. The studies based on source code analysis in static analysis type [14-16], and Sandbox usage [17], client-server structure [18], behaviour based surveillance [19], API calls [20] monitoring researches in dynamic analysis were made. By using different approaches, it was tried to achieve more robust results and to increase the safety of mobile users.

In recent years, the idea of using the features, derived from apps based on static and dynamic analysis, in classification and regression problems has been put forward. The development of machine learning techniques and the ability to come up with good results for many different

Received 8 July 2020; Received in revised form 15 July 2020; Accepted 25 Julya 2020;
Available online 15 September 2020
doi: 10.46291/ICONTECHvol4iss2pp15-27

problems, adopted the idea of using it in this areas as well. In the article is prepared by Domhnall et al [21], 23 different machine learning algorithms have been tested on well known databases. During this testing period, classification performances, accuracy and CPU usage times were compared with accepted traditional methods. It is aimed to reveal a general opinion about the results of ML techniques on standard malware datasets. In the study [22] using the method of classifying the features obtained as a result of the analysis of the network usage traffic of mobile devices with ML, a very high success rate of 99.9% was achieved. Only usage statistics regarding network usage are used. In order to get features of apps, a pre-processing in the form of dynamic analysis was applied. Similarly, in the article written by Bahtiyar et al [23], it was tried to predict the malignant apps using multi-layered Stuxnet architecture. It uses regression models. Tests were carried out by creating a sample malignant software dataset. R2 value was found to be 0.8203. Thus, it has been shown that the proposed structure can produce acceptable outcomes.

Many studies have been carried out for many years to detect malware, which is the nightmare of Android mobile users. Today, studies still continue actively in this field as we have done.

MATERIAL AND METHODS

Application permissions are the basis of security in the Android operating system. Therefore, many studies have been conducted to detect malevolent apps based on permission [25-27]. The permission requested by the applications both determine the areas of accessibility and draw the limits of their malicious activities. Considering this situation, in this study, it has been studied to get requested permissions, to transform this permission into features and to classify with KNN algorithm.

Architecture of Proposed Model

In order to design the proposed model, it is necessary to complete a number of pre-processing steps such as creating original dataset, extracting source codes and determining requested permission. The purpose of this process is to obtain the input feature vectors needed in the implementation of ML methods.

Reverse engineering tools are required to reach source code of Apk application files. In this study, “JADX” is used. With the help of it, the conversion from the compiled java file “classes.dex” to readable java code has been done. In this way, it is provided to examine the java codes, to reach details about application activities and to extract application feature vectors. Application java files is scanned using code analysis approach techniques and features such as permissions, API call parameters is obtained. Thus, these feature vectors is used for classification with KNN

K-nearest Algorithm (KNN)

Received 8 July 2020; Received in revised form 15 July 2020; Accepted 25 Julya 2020;
Available online 15 September 2020
doi: 10.46291/ICONTECHvol4iss2pp15-27

The K-nearest algorithm (KNN) has been proposed by Thomas Cove and is a suggested method for classification and regression problems [24]. “k” closest training samples in the feature space is used as a input vector in both cases. The output of model will vary depending on whether the KNN approach is used for regression of classification. An test example is classified according to its closest neighbours and assigned to the class that hosts the closest neighbours. If the k value takes as 1, whatever class of its closest neighbour will be assigned to that class. In regression, the output will be the average of the next k neighbour’s feature values. It is a type of sample-based learning algorithms.

In the KNN algorithm, the Euclid distance define by the formula (1) is used to calculate the distance between the test and input data.

$$d(p,q) = \sqrt{(q1-p1)^2 + (q2-p2)^2 + \cdots + (qn-pn)^2} \qquad (1)$$
$$= \sqrt{\sum_{i=1}^{n} (qi-pi)^2}$$

In the formula given above, n denotes the number of different features in ML. The data closest to the test point is assumed to belong to that class.

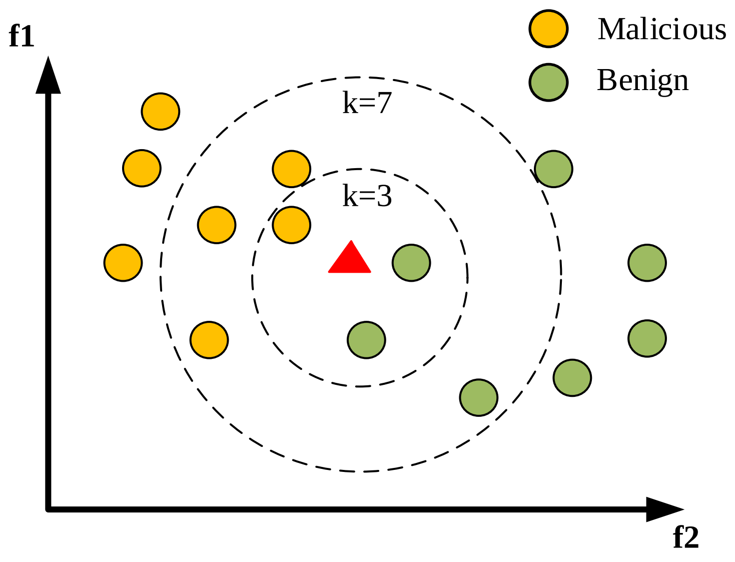


Figure 1. KNN Algorithm Working Model

A sample KNN chart is shown in Figure 1. The data market with the red triangle has been accepted as test data. Accordingly, it is tried to classify whether the test data is malicious of benevolent application. If k value is taken as 3, the 3 samples closest to the test data or k is taken as 7, 7 samples closest to the test data are included in the calculation. The distances between these input samples and test data are calculated according to the Eq.1. Hereunder, the test data is determined as belonging to the class with the smallest distance value. Due to the nature of the approach, in some cases the distance values to the two classes can be equal. In this

Received 8 July 2020; Received in revised form 15 July 2020; Accepted 25 Julya 2020;
Available online 15 September 2020
doi: 10.46291/ICONTECHvol4iss2pp15-27

case, calculation are done by adding an extra input data. In the Figure 1, the red test data will be taken into the benign class, which is green.

Apk Structure

Android applications are in a compressed file format with the extension "apk" (Android Application Package). Its structure is as shown in Figure-2. Android manifest file is in XML structure. It contains metadata, information about hardware and software features and especially the permissions of the application. Manifest.xml file used in static analysis used to detect malicious applications. "Classes.dex" contains the compiled application java files. It has an executable structure in the Dalvik virtual machine. The res file contains knowledge about the graphic and audio definitions, visual drawings and the language used. The "Meta-INF" includes certification and verification parameters for security and data integrity. The "Lib" folder is the repository of native libraries Recorded files are stored in the "Asset" folder. It is protected with the same name in the compiled package during the compilation stage. The "Resources.arsc" is a compiled binary file. It contains all application files including libraries.

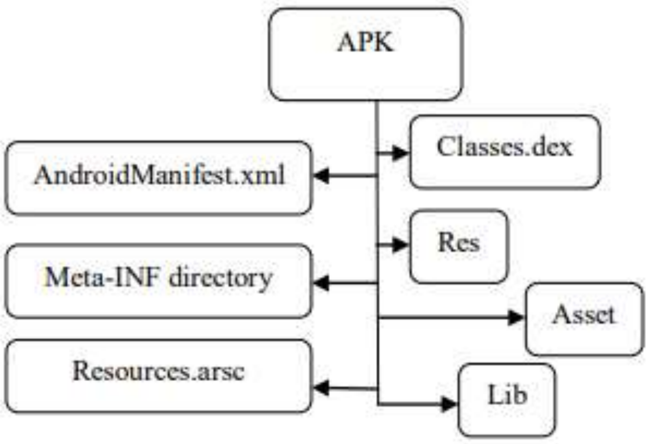


Figure 2. APK Structure [24]

Although it has a simple structure than other different application development platforms, there are critical information files and folders in the APK architecture that should be considered in terms of security. Thus, the files included in the apk package are examined in different malware detection approaches such as static and code analysis. Thereby, it is possible to estimate the purpose of the Android application and take measures accordingly.

Performance Measure

Different metrics are used to measure the performance of malware detection models and to determine the accuracy level of these measurements such as precision, recall, f1-measure etc. The value of TP (True Positive) is a result that the model predicts correctly the positive class and TN (True Negative) is a result where the model anticipate correctly the negative class. Whether a negative sample is incorrectly classified as positive is considered to be FP (False

Received 8 July 2020; Received in revised form 15 July 2020; Accepted 25 Julya 2020;
Available online 15 September 2020
doi: 10.46291/ICONTECHvol4iss2pp15-27

Positive), and if a positive sample is classified as negative is considered to be FN (False Negative). There values are shown in the confusion matrices.

The well known performance measure is the Accuracy value (Acc), which indicated the accuracy of the classification performance. It is calculated with the formula given in the Eq.2. It refers to the ratio of the number of correct recognition of malware and benign application examples to the total number of samples.

$$Acc = \frac{TP+TN}{TP+FP+FN+TN} \quad (2)$$

In addition, the equations of Precision (P) (3), Recall(R) (4) and, the harmonic mean of these values, F-measure(F) (5) are given below.

$$Precision\ (P) = \frac{TP}{TP-FP}\ (3)\ ,\ Recall\ (R) = \frac{TP}{TP+FN}\ (4)$$

$$FMeasure = \frac{2PR}{P+R} \quad (5)$$

Since the F-measure value expresses the correct classification rates of both malignant and benevolent apps, in some cases, it is preferred as a measurement metric instead of Accuracy.

EXPERIMENT AND RESULTS

Different approaches have been used in the preparation of malicious and benign software datasets to be used for classification with KNN. In the tests of the proposed model, 492 malicious sample software from Gnome malware dataset is used. Applications that will use in training and testing phase is chosen randomly. In the selection of benevolent software, 697 samples selected from different type of mobile applications such as banking, business, weather, travel is used (Figure-3). A very rigorous study was carried out in the selection of mobile apps that is scanned on “virustotal” [28] web site. It is aimed to prepare a homogenous dataset. Thus, training and testing results is provided to be more accurate.

One of the advantages of the KNN classifier is that it can achieve results fast and requires less process time. So, CPU based tensorflow library and a computer with 2.4GHz processor speed have been used to tests. Besides, numpy, pandas, sklearn libraries is used to test our proposed model.

Received 8 July 2020; Received in revised form 15 July 2020; Accepted 25 Julya 2020;
Available online 15 September 2020
doi: 10.46291/ICONTECHvol4iss2pp15-27

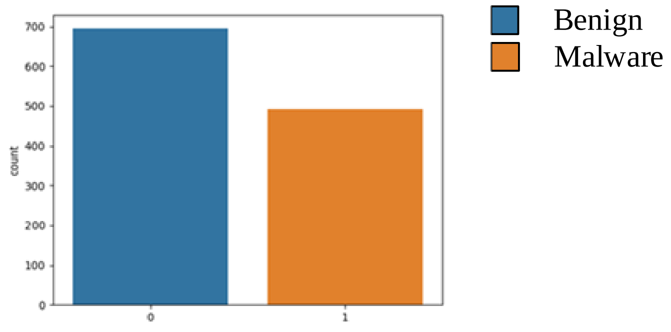
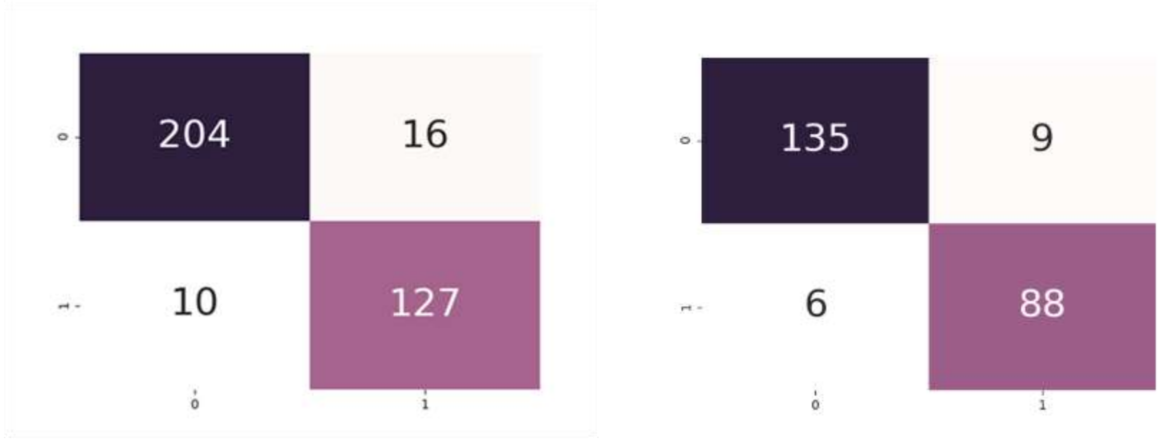


Figure 3. The distribution of Malicious and Benign Application in dataset

In the selection of training and test data, randomly selected datasets is used. Accordingly, test is carried out with both 70%-30% and 80%-20% training and test rates. These choices have been repeated many times for different experiments, and the results are described in this section. Precision, recall and f measure metrics are used for performance evaluation.

K-nearest neighbourhood (KNN) algorithm is used as classifier that is one of the supervised learning algorithms and is successful in solving classification problems. The distance metric is set at 5. Euclidean distance is used to calculate the distance of the test data with 5 sample point. Minkowski is selected as the distance metric. In total, tests is carried out with 1189 mobile applications. In Figure 4, the confusion matrices obtained for both 70%-30% and 80%-20% selections are given.



70% Training – 30% Test

80% Training – 20% Test

Figure 4. Confusion Matrices for two dataset

Successful classification results is obtained with the proposed model in both tests, and the number of FP and FN is very low. For the tests, in which the 70%-30% distribution is used as a training-test, 331 of the total 357 applications is correctly classified. The accuracy level is 91.2%, while the sensitivity and Specifity values are 95.33% and 88.81%, respectively. The fact that both sensitivity and specifity values are above a certain level indicated that the model is not good in one-way (only malicious detection or benign detection) but it is successful in

Received 8 July 2020; Received in revised form 15 July 2020; Accepted 25 Julya 2020;
Available online 15 September 2020
doi: 10.46291/ICONTECHvol4iss2pp15-27

both cases. In the classification problems, when the number of cluster elements is not evenly distributed, just measuring the accuracy level of the model is often an insufficient metric. Thence, the performance of proposed model is analysed with precision, recall and f1 measures.

Precision value is realized as 91.2%. In detecting malicious applications, selecting benevolent software as malware can have serious headaches. High value in precision shows that the model is successful in FP ranking. Furthermore, 92.7% level has been caught in Recall calculation. It shows that it gives good results in detecting malwares. The f1 value, which is a result of precision and recall values evaluated together and unbalanced cluster distributions can be observed, is 92.4%. Wonderful results is obtained in this metric, where all costs is evaluated.

Similarly, a high Accuracy rate of 93.7% is reached in the test where 80%-20% training-test data were selected. While the specifity increased to 90.72%, the sensitivity ascended to 95.74%. Precision, recall and f1 measure values is 90.7%, 93.6% and 92.1% in turn.

Comparing the test results made with the two data selection methods, increasing the training data enables more powerful results in all measurement metrics, especially system classification accuracy. This shows that it will be possible to catch higher levels of success can be achieved with datasets containing more malicious and benign applications.

The ROC probability curve of the model is shown in Figure-5. It shows that the capacity of the model distinguish between malevolent and benevolent programs. High AUC value indicated that the model performance is better. 0.94 AUC is obtained with proposed model. This situation, which is very close to 1, supports that the model is prosperous.

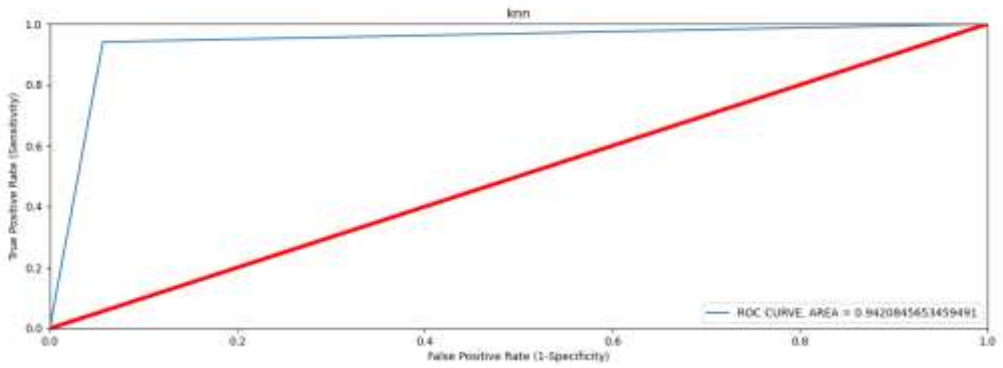


Figure 5. ROC Curve of Proposed Model

The effect of the k value change, which is an important parameter in the KNN algorithm for the accuracy level of the model and F1-measurement, is shown in Table 1. According to table, the most successful results is obtained when the k value is determined as 5 for that 70%-30% training-test distribution. In this situation, Accuracy value is 94.9%, Recall is 93.7%, Precision is 95.0% and F-measure is 92.4%.

Received 8 July 2020; Received in revised form 15 July 2020; Accepted 25 Julya 2020;
Available online 15 September 2020
doi: 10.46291/ICONTECHvol4iss2pp15-27

Table 1. Effect of k value change on model recognition performance

K value	Accuracy (%)	Recall (%)	Precision (%)	F-measure (%)
1	91.0	90.5	86.7	88.6
2	93.8	88.3	95.3	91.7
3	93.5	93.4	90.1	91.8
4	93.2	91.2	91.2	91.2
5	**94.1**	**91.2**	**92.7**	**92.4**
6	93.8	92.7	91.4	92.0
7	92.9	93.4	88.9	91.1
8	92.7	92.7	88.8	90.7
9	93.8	93.4	90.8	92.1
10	93.1	91.2	93.4	92.3

In the results showed in Table 1, the recognition performance never fell below 90%. This indicates that the proposed model has high recognition rate. Changes in the parameters do not effect the results to go below a certain level. However, with the best selection, the best results is produced compared to other models. The results have shown the effect of using the KNN classifier for malware detection.

DISCUSSION AND CONCLUSION

In this study, a model that detects malware for Android mobile devices using K-nearest neighbourhood(KNN) classification method is proposed. Nowadays, the use of machine learning techniques has become widespread, and many studies have been conducted using the KNN model. The paper numbered [29] where KNN model was used, different accuracy levels of k value according to the change were obtained. On average, 92.63% correct classication rate was catched. In the study prepared by Kedziora et al. [30], tests were carried out with a total of 1958 application including 996 malicious software. In the classification modeled with KNN, performance values varying between 78.3% and 80% were obtained for different metrics. Similarly, In the research tested with 10747 sample applications, 81.57% performance rate was obtained with the use of KNN [31] and AUC value was 87.36%. In another study, in the malware detection tools where KNN algorithm is used as classifer, Hamming, Euclidean and Chebyshev algorithms are given comparatively as distance calculation metric. According to this, although the results are close to each other, the lowest error rate were achieved with Euclidean distance [32].

Received 8 July 2020; Received in revised form 15 July 2020; Accepted 25 Julya 2020;
Available online 15 September 2020
doi: 10.46291/ICONTECHvol4iss2pp15-27

The protection of mobile devices from malignant activities is an issue that the researchers are working on, because of that the use of the Android OS on these devices are increasing day by day. In this work, an original dataset is created to distinguish malicious apks files from benign ones. A balanced distribution is attempted in the number of benign and malevolent applications in this dataset. A feature vector containing permissions and other components is created and classified with KNN which is one of the machine learning technique. Although different results are attained in this test, the performance level is determined as 94.1%. Compared to other similar level studies using KNN, better accuracy rate is achieved. The reason for this is thought to be a good model and dataset, and well-chosen parameters. In the to the Accuracy value, 92.4% is also obtained in the F-measure. It shows that this model can successfully distinguish both malware and benign classes.

In the static and dynamic analysis approached used from the past to the present, it can be quite difficult to achieve more thatn 90% success level. However, with machine learning techniques, these performance rates can be accessed with smaller datasets. At this point, it is necessary to work with larger and more comprehensive databases in order to provide safer mobile device usage environments to users, especially decreasing FP rates. Accurate detection levels of models trained with more and more applications for malignant and benovolent applications will increase rapidly. Moreover, if different classifiers such as J48, Random Forest or ANN are used, different results will be possible. If work continues in this area in the future, it will be possible for people to use more secure Android mobile devices.

REFERENCES

[1] Rahim Taheri, Meysam Ghahramani, Reza Javidan, Mohammad Shojafar, Zahra Pooranian, Mauro Conti, Similarity-based Android malware detection using Hamming distance of static binary features, Future Generation Computer Systems, Volume 105, 2020, pp. 230-247.

[2] Y. Zhou, X. Jiang, “Dissecting android malware: characterization and evalution”, 2012 IEEE Symposium on Security and Privacy (SP), (2012), pp. 95-109.

[3] Manel Jerbi, Zaineb Chelly Dagdia, Slim Bechikh, Lamjed Ben Said, “On the use of artificial malicious patterns for android malware detection”, Computer and Security, (2020), 92, 1-22.

[4] C. Willems, T. Holz and F. Freiling, “Toward automated dynamic malware analysis using cwsandbox”, IEEE Secur. Privacy, 5 (2), (2007)

[5] K. Rieck, T. Holz, C. Willems, P. Düssel and P. Laskov,” Learning and classification of malware behavior” , International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Springer (2008), pp. 108-125.

[6] H. Cai, N. Meng, B. Ryder, D. Yao, Droidcat: Effective android malware detection and categorization via app-level profiling, IEEE Trans. Inf. Forensics Secur. 14 (6) (2019) 1455–1470.

[7] Fei Tong, Zheng Yan, "A hybrid approach for mobile malware detection in Android", Journal of Parallel Distributed Computing", (2016), 103, 22-31.

[8] Shamsul Huda, Jemal Abawajy, Mamoun Alazab, Mali Abdolalihian, Refiqul Islam, John Yearwood, "Hybrids of Suppor vector machine wrapper and filter based framework for malware detection", Future Generation Computer Systems, (2016), 55, 376-390.

[9] Sitalaskhmi Venkatraman, Mamoun Alazab, R. Vinayakumar, "A hybrid deep learning image-based analysis for effective malware detection", Journal of Information Security and Applications, (2019), 47, 377-389.

[10] Daniel Gilbert, Carles Mateu, Jordi Planes, "The rise of machine learning for detection and classification of malware: Research development, trends, and challenges", Journal of Network and Computer Applications, 153, (2020), 1-22.

[11] Sen Chen, Minhui Xue, Lingling Fan, Shuang Hao, Lihua Xu, Haojiin Zhu, Bo Li, " Automated poisoning attacks and defenses in malware detection systems: An adversarial machine learning approach, Computer & Security, (2018), 73, 326-344.

[12] Steve Mansfield-Devine, "Android malware and mitigations", Network Security, (2012), 11, 12-20.

[13] Mark Guido et al. "Automated idenfitication of installed malicious Android applications", Digital investigation, (2013), 10, 96-104.

[14] Recep Sinan ARSLAN, İbrahim Alper Doğru, Necaattin Barışçı, "Permission-based malware detection system for Android using machine learning techniques", International Journal of Software Engineering and Knowledge Engineering, (2019), 29(1), 43-61.

[15] Muhammad Amin et al. " Static malware detection and attribution in android byte-code through an end to end deep system", Future Generation Computer Systems, (2020), 112-126.

[16] Guruswamy Nellaivadivelu, Fabio Di Troia, Mark Stamp, "Black box analysis of android malware detectors", Array, (2020), 6, 1-9.

[17] Muzzamil Noor, Haider Abbas, Waleed Bin Shahid, "Countering cyber threats for industrial applications: An automated approach for malware evasion detection and analysis", Journal of Network and Computer Applications" (2018), 103, 249-261.

[18] Shanshan Wang et al. " A mobile malware detection method using behaviour features in network traffic", Journal of Network and Computer Applications", (2019), 133, 15-25.

[19] Elmouatez Billah Karbab, Mourad Debbabi, "MalDy: Portable, data-driven malware detection using natural language processing and machine learning techniques on behavioral analysis reports", Digital Investigation, (2019), 28, 77-87.

[20] Moutaz Alazab et al. "Intelligent mobile malware detection using permission requests and API calls", Future Generation Computer Systems, (2020), 107, 509-521.

Received 8 July 2020; Received in revised form 15 July 2020; Accepted 25 Julya 2020;
Available online 15 September 2020
doi: 10.46291/ICONTECHvol4iss2pp15-27

[21] Domhnill Carlin, Philip O'Kane, Sakir Sezer, "A cost analysis of machine learning using dynamic runtime opcodes for malware detection, Computers & Security, (2019), 85, 138-155.

[22] Zhenxiang Chen et al. "Machine learning based mobile malware detection using highly imbalanced netwok traffic", Information Sciences, (2018),433, 346-364.

[23] Şerif Bahtiyar, Mehmet Barış Yaman, Can Yılmaz Altıniğne, "A multi-dimansional machine learning approach to predict advanced malware", Computer networks, (2019=, 160, 118-129.

[24] Cover, Thomas M., Hart, Peter E. (1967). "Nearest neighbor pattern classification", IEEE Transactions on Information Theory. 13(1), 21–27.

[25] Recep Sinan Arslan, İbrahim Alper Doğru, Necaattin Barışçı, "Android Mobil Uygulamalar için İzin Karşılaştırma Tabanlı Kötücül Yazılım Tespiti", Politeknik Dergisi, (2017), 20(1), 175-189.

[26] Karakuş A. T., Doğru İ.A., Çetin A., "APK Auditor: Permission-based Android Malware Detection System", Digital Investigation, Vol. 13 (1), pp. 1–14, 2015.

[27] Utku A., Doğru İ.A., "Permission Based Detection System for Android Malicious Software", J. Fac. Eng. Arch. Gazi Univ., Vol. 32 (4), pp. 1015-1024, 2017.

[28] https://www.virustotal.com/gui/home

[29] Udayakumar N, Subbulakshmi.T, Ayush Mishra, Shivang Mishra and Puneet Jain, "Malware Category Prediction using Knn and Svm Classifiers", International Journal of Mechanical Engineering and Technology (IJMET) Volume 10, Issue 02, February 2019, pp. 787-797

[30] Michal Kedziora, Paulina Gawin, Michal Szczepanik and Ireneusz Jozwiak, "Malware detection using machine learning algorithms and reverse engineering of Android Java Code", International Journal of Network Security & Its Applications (IJNSA), (2019), 11, 1-14.

[31] Michał Jacek Kruczkowski, Ewa Niewiadomska-Szynkiewicz, "Comparative study of supervised learning methods for malware analysis", Journal of Telecommunications and Information Technology, (2014), 4, 1-11.

[32] G. Baldini and D. Geneiatakis, "A Performance Evaluation on Distance Measures in KNN for Mobile Malware Detection," 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT), Paris, France, 2019, pp. 193-198

[33] Mansour Ahmadi, Ashkan Sami, Hossein Rahimi, Babak Yadegari, "Malware detection by behavioural sequential patterns", Computer Fraud and Security, (2013), 8, 11-19.

Received 8 July 2020; Received in revised form 15 July 2020; Accepted 25 Julya 2020;
Available online 15 September 2020
doi: 10.46291/ICONTECHvol4iss2pp15-27