Intelligent Android Malware Detection Using Radial Basis Function Networks and Permission Features

Ammar Abdulrahman , Khalid Hashem, Gaze Adnan, Waleed Ali

Information Technology Department, Faculty of Computing and Information Technology,
King Abdulaziz University, Rabigh, Saudi Arabia

Summary
Recently, the quick development rate of apps in the Android platform has led to an accelerated increment in creating malware applications by cyber attackers. Numerous Android malware detection tools have utilized conventional signature-based approaches to detect malware apps. However, these conventional strategies can't identify the latest apps on whether applications are malware or not. Many new malware apps are periodically discovered but not all malware Apps can be accurately detected. Hence, there is a need to propose intelligent approaches that are able to detect the newly developed Android malware applications. In this study, Radial Basis Function (RBF) networks are trained using known Android applications and then used to detect the latest and new Android malware applications. Initially, the optimal permission features of Android apps are selected using Information Gain Ratio (IGR). Appropriately, the features selected by IGR are utilized to train the RBF networks in order to detect effectively the new Android malware apps. The empirical results showed that RBF achieved the best detection accuracy (97.20%) among other common machine learning techniques. Furthermore, RBF accomplished the best detection results in most of the other measures.

Keywords: *Android applications, Android malware detection, Radial basis function network, Feature selection*

1. Introduction

In the last few years, the Android platform has become one of the most known mobile platforms for smart mobile devices as it is uncharged and open source. The quick development percentage of apps in the Android platform has led to a tremendous increment in creating and spreading Android malware apps by cyber attackers. Fig.1. illustrates the number of mobile malware installation packages detected by Kaspersky between the first quarter of 2019 and the first quarter of 2020 [1].

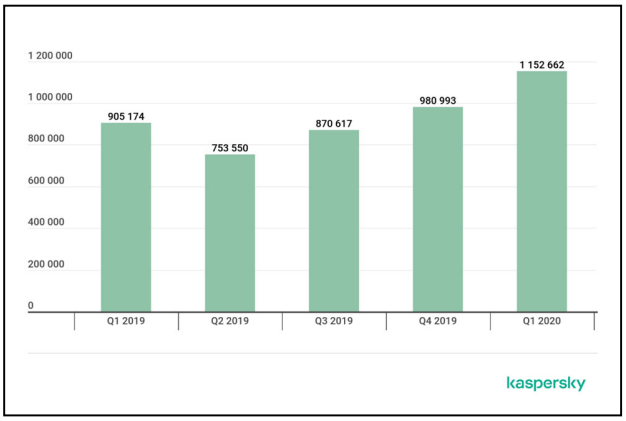

Fig.1. The number of mobile malware installation packages detected by Kaspersky between Q1 2019 and Q1 2020 [1].

Numerous Android malware detection tools have utilized the conventional signature-based approaches to detect malware apps periodically. However, these conventional strategies can't identify the latest android malware apps on whether the new applications are malware or not. Even at the official Google Play store, many new malware apps are periodically discovered but not all Android malware Apps are discovered, especially in early stages of publication in the Google Android Market [2,3]. Hence, there is a need to propose intelligent approaches that are able to recognize the latest Android malware apps.

Recently, in order to address the major obstacles of the conventional Android malware detection approaches based on the signature, some popular intelligent techniques such as artificial neural network (ANN), support vector machine (SVM), Naive Bayes, and decision trees were employed in the existing works [4-7] to distinguish effectively the latest Android malware apps.

This study proposes utilizing one of the fast neural networks called Radial Basis Function (RBF) networks to detect malware applications. The RBF networks have several attractive advantages such as simple structure, good generalization, strong tolerance to input noise, and online learning ability [8]. These good characteristics of RBF

Manuscript received June 5, 2021
Manuscript revised June 20, 2021
https://doi.org/10.22937/IJCSNS.2021.21.6.37

networks make it more appropriate machine learning to be utilized in Android malware detection. In addition, Information Gain Ratio (IGR) was used to select the most important permission features of Android apps. The experimental results showed that RBF achieved outstanding results in most of the detection measures and accomplished better performance compared to other common machine learning techniques.

The rest of the study is arranged as follows. Some selected works on intelligent Android malware detection based on permission features are given in Section 2. Section 3 explains the suggested methodology of intelligent Android malware detection method based on Radial Basis Function networks and Information Gain Ratio. Section 4 discusses the performance results of Radial Basis Function network with Information Gain Ratio. Lastly, the conclusion and future work of the proposed work is given in Section 5.

2. Related Work

The study of existing works is a very important step to avoid the possibility of duplicating similar problems in our study. In this study, we concentrate on some existing intelligent detection methods that used static malware analysis, especially based on permission features, in android malware detection. In the Android malware detection methods based on static analysis, the Android malware applications are detected without executing the Android applications. Therefore, they are simpler and quicker, and require less utilization of resources compared to the dynamic analysis-based methods.

Table 1: Summary of some recent intelligent Android malware detection methods based on permission features.

Approach	Features	Machine learning	Feature selection
Mining API calls and permissions for Android malware detection [9]	Permissions and API calls	Naive Bayes and ANN	Correlation-based feature selection and information gain
Static detection of Android malware by using permissions and API calls [10]	Permissions and API calls	Naive Bayes, SVM, MLP, random forest, and J48	Information gain
Exploring Permission-induced Risk in Android Applications for Malicious Application Detection [11]	Individual permission and group of collaborative permissions.	SVM, decision trees, and random forest.	Forward selection (SFS) and principal component analysis (PCA)
A probabilistic discriminative model for Android malware detection with decompiled source code [12]	API calls and permissions	Regularized logistic regression	Information gain and Chi-square
K-ANFIS [13]	Permission features	kEFCM-based Adaptive Neuro-Fuzzy Inference System	Information gain ratio
DroidDetector [14]	Static analysis-based features and dynamic analysis -based features	Deep belief networks	Frequency analysis -based feature evaluation
EHNFC [15]	Permission features	Hybrid neuro-fuzzy classifier with evolving clustering	Information gain ratio
Identification of malicious Android app using manifest and opcode features [16]	Static features from the manifest and executable files	SVM, random forest, and rotation forests	Entropy based Category Coverage Difference and Weighted Mutual Information
Droid-HESVMGA and Droid-HESVMPSO [17]	Permission features	evolving support vector machine	Information gain ratio

In [9], the best features of permissions and API calls were selected using correlation-based feature selection and

information gain. Accordingly, the training dataset with the best features was used to train Naive Bayes and ANN in order to detect the Android malware apps.

In [10], Naive Bayes, SVM, MLP, random forest, and J48 were trained based on the best permissions and API calls features selected by information gain technique. The trained classifiers were then used to distinguish the Android malware apps from benign apps.

In [11], authors used forward selection (SFS) and principal component analysis (PCA) to select the best features of individual permission and group of collaborative permissions. Then, SVM, decision trees, and random forest were trained and subsequently used to detect the Android malware apps.

In [12], information gain and Chi-square techniques were utilized to select the best features of API calls and permissions. Regularized logistic regression was then trained and employed to detect the Android malware apps.

In K-ANFIS [13], kEFCM-based Adaptive Neuro-Fuzzy Inference System was suggested to detect the Android malware apps after training based on the best permissions features selected by information gain ratio.

In DroidDetector [14], frequency analysis-based feature evaluation was used to select the best static and dynamic features. Subsequently, deep belief networks were trained and then used to detect the Android malware apps.

In EHNFC [15], the most important permission features were identified by information gain ratio. Accordingly, hybrid neuro-fuzzy classifier with evolving clustering was trained and then used to detect the Android malware apps.

In [16], authors used Entropy based Category Coverage Difference and Weighted Mutual Information to find the best static features from the manifest and executable files. Then, they trained SVM, random forest, and rotation forests using training data with the best features in order to detect the malicious Android apps.

In [17], evolving support vector machine based on evolutionary algorithms called Droid-HESVMGA and Droid-HESVMPSO were trained with the best permissions features selected by information gain ratio. The trained Droid-HESVMGA and Droid-HESVMPSO were used to detect the Android malware apps.

Table 1 shows a summary of some recent intelligent Android malware detection based on permission features. As can be seen from Table 1, the existing works used some popular intelligent techniques such as MLP, SVM, decision trees, and Naive Bayes. Unlike the existing works, our study utilized one of the fast neural networks called RBF networks to detect malware applications. Besides, Information Gain Ratio was used to select the most important permission features of Android apps.

3. Materials and Methods

This section explains a suggested methodology of intelligent Android malware detection method based on Radial Basis Function (RBF) and Information Gain Ratio (IGR). Fig. 2 shows the methodology of the proposed intelligent Android malware detection approach based on the RBF network and IGR.

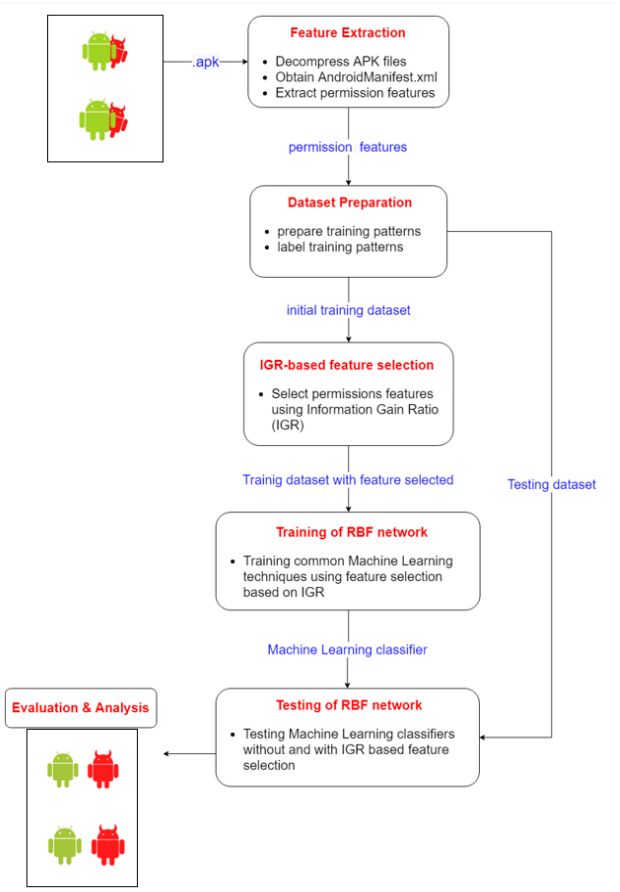

Fig. 2. The suggested methodology of intelligent Android malware detection approach based on RBF network and IGR

As can be seen from Fig. 2., the methodology includes six phases: data collection, feature extraction, dataset preparation, IGR-based feature selection, training of RBF networks, and testing of RBF networks. The first five phases are explained in the following subsections, while the evaluation and testing of RBF networks will be explained in Section 4.

3.1 Data Collection of Android Malware and Benign Applications

Many malware and benign applications were collected and analyzed by several researchers from several various sources such as Genome [18], Contagiodump [19], Certtools [20], Google Play Store [21], VirusShare [22], MitchellKrogza [23], TheZoo [24], VirusBay [25], and DasMalwerk [26].

Malgenome-215-dataset [27] used by [28] was used in this paper. The Malgenome-215-dataset includes 1260 Android malware apps and 2539 Android benign apps.

3.2 Feature Extraction

The permission features are the most significant features in Android applications since they are critical in recognizing malware from benign applications. Android applications include AndroidManifest.xml as shown in Fig. 3., which is an XML file that has all the permission features required to work properly in Android applications. As seen in Fig. 3., the permission features required to access system resources are declared in the AndroidManifest.xml. The developers of Android applications usually declare all permissions required to access system resources using <uses-permission> tags in the AndroidManifest.xml. Then, some tools are used to decompress the Android application packets, and then the permission features can be obtained from AndroidManifest.xml.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="com.example.snazzyapp">

    <uses-permission android:name="android.permission.SEND_SMS"/>

    <!-- other permissions go here -->

    <application ...>
        ...
    </application>
</manifest>
```

Fig. 3. The permission features in AndroidManifest.xml that are required to use system resources [17]

3.3 Dataset Preparation

In machine learning, a dataset is usually organized as a table with rows and columns. The columns can also be referred to as features or ‘variables, while the rows refer to examples or instances.

In the Android malware detection, the rows represent Android apps in the dataset while the permission features are the columns in the dataset. The permission features of Android applications are converted into numerical form to be effectively used in the training phase. If the feature is available in the application, it will be assigned to 1. If it is not available, it will be assigned to 0. Table 2. shows an example of dataset preparation for 5 apps and 6 permission features in Android malware detection.

Table 2: An example of dataset preparation in Android malware detection

	SEND SMS	RECEIVE SMS	GET ACCOUNTS	MANAGE ACCOUNTS	USE CREDENTIALS	INSTALL PACKAGES	Class (Label)
App1	1	0	1	1	0	1	1 (Malware)
App2	0	1	0	0	0	0	0 (Benign)
App3	0	1	0	0	0	0	0
App4	1	1	1	1	1	0	1
App5	0	0	0	1	0	1	1

3.4 Information Gain Ratio Based Feature Selection

Feature selection is an extremely important step that can select only the most relevant and useful features to be used with machine learning training sets, which dramatically reduces costs and data volume [29, 30].

In this study, we used feature selection based on Information Gain Ratio (IGR) algorithm. IGR is one of the most common filter feature selections. IGR is a ratio of the amount of information gained about a random variable or signal from observing another random variable, which uses the information gained to rank the most useful feature. The filter feature selection methods rank the features independently, without training any machine learning algorithm. Therefore, they are more straightforward and faster compared to the wrapper methods. Hence, the filter feature selection methods are more suitable for mobile environment [17, 31].

3.5 Training of RBF Networks

RBF networks is a neural network with three fixed layers: the input layer, hidden layer, and output layer. In our study, the input layer in RBF networks consists of several inputs or nodes that represent the permission features of Android apps selected by IGR as explained in Section 3.4. In addition to the input layer, a number of nodes in the hidden layer is determined by experiments. In the

output layer, we have just one node that represents the class of Android app, either malware application (1) or benign application (0).

Fig. 4. shows the architecture of the RBF network with six inputs as an example. In Fig. 4. there are six input nodes, or equivalently, an input vector with six binary values. If the feature is available in the app, it will be represented by 1, otherwise, it will be represented by 0. The hidden layer consists of four hidden nodes. Each hidden node has a centroid which is a vector with the same number of values as the input. RBF centroids are usually given the symbol μ. Each hidden node has a width (sometimes called a standard deviation) which is a single numeric value. RBF widths are usually given the symbol σ.

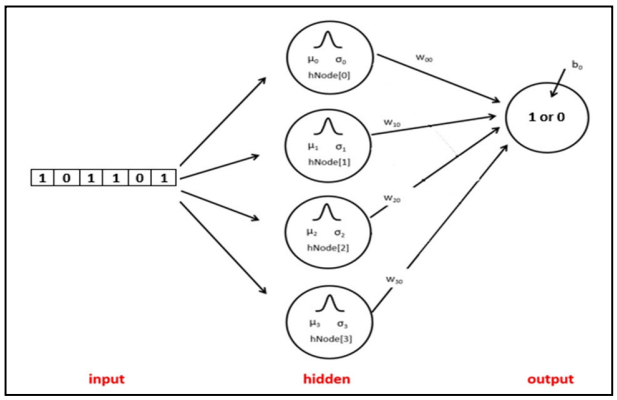

Fig. 4. An example of the architecture of RBF network with six features or inputs

The computed output of an RBF network depends on the input values, and the values of centroids, widths, and weights. Training of RBF network is the process of finding the best values of the centroids, widths, and weights.

The simplest approach for setting the RBF parameters is to select centroids randomly from training data. The weights are determined by minimizing an objective function (mean-squared error), which is the difference between the network outputs and the target values. In this paper, the method of the pseudo inverse was used to find the best weights during the training phase of RBF networks.

4. Results and Discussion

In this study, RBF used to detect malware apps was implemented by using Python language. This section discusses the performance results of RBF with IGR feature selection method. Besides, the performance of RBF will be compared with other common machine learning such as Support Vector Machine (SVM), Decision Tree (DT), Multilayer Perceptron networks (MLP), and Naive Bayes (NB).

4.1 Dataset Collection

In this study, Malgenome-215-dataset [27] used by [28] was adopted in all experiments. Table 3 shows description of Malgenome-215-dataset. The Malgenome-215-dataset dataset consists of 3799 Android applications with 215 features. The data includes 2539 benign and 1260 malware apps collected from the Android malware Malgenome project [18], which is one of the common sources for malware apps collection used by the malware research society. We extracted 109 permission features from Malgenome-215-dataset to train RBF networks.

Table 3: Details of Malgenome-215 dataset used in the experiments.

Number of Android apps	3799
Number of Android malware apps	1260
Number of Android benign apps	2539
Number of all features	215
Number of permission features	109

4.2 Performance Measures

In this study, we used hold-out validation for evaluating the performance of the RBF network in Android malware detection. In the hold-out validation, the data collected was randomly partitioned into two parts: training data (70%) and testing data (30%).

The performance in terms of correct classification rate (CCR) was calculated for testing dataset to validate and verify the performance of Radial Basis Function (RBF) against Decision Tree (DT), Naive Bayes (NB), multilayer perceptron (MLP), and Support Vector Machine (SVM).

In addition to CCR, some important measures were calculated to evaluate the performance of machine learning techniques. Table 4 shows the confusion matrix for Android malware detection. In Tables 4, TP, TN, FP, and FN represent true positive, true negative, false positive, and false negative, respectively. Based on the confusion matrix, True positive rate (TPR), false positive rate (FPR), true negative rate (TNR), false negative rate (FNR), and area under ROC curve (AUC) were also computed to validate the performance of the machine learning techniques.

Table 4: The confusion matrix for Android malware detection.

	Classified as malware	**Classified as benign**
Malware app	TP	FN
Benign app	FP	TN

The performance measures used to evaluate the proposed method are described in the following:

- CCR: The rate of malware and benign apps correctly classified with respect to all the apps as shown in Eq. (1).

$$CCR = \frac{TP+TN}{TP+FP+FN+TN} \times 100 \quad (1)$$

- TPR: The rate of malware apps classified as malware out of the total malware apps as shown in Eq. (2).

$$TPR = \frac{TP}{TP+FN} \times 100 \quad (2)$$

- TNR: The rate of benign apps classified as benign out of the total benign apps as shown in Eq. (3).

$$TNR = \frac{TN}{TN+FP} \times 100 \quad (3)$$

- FPR: The rate of benign apps classified as malware out of the total benign apps as shown in Eq. (4).

$$FPR = \frac{FP}{FP+TN} \times 100 \quad (4)$$

- FNR: The rate of malware apps classified as benign out of the total malware apps as shown in Eq. (5).

$$FNR = \frac{FN}{FN+TP} \times 100 \quad (5)$$

- AUC: This metric measures the trade-off between TPR and FPR as shown in Eq. (6).

$$AUC = \frac{1+TPR-FPR}{2} \times 100 \quad (6)$$

4.3 Performance of RBF with Changing Parameters

To ensure the best performance for the RBF network, we performed many experiments with changing the number of hidden nodes and the value of Sigma(σ) until we reached the best result for the RBF network. Table 5 shows the best performance measures of the RBF network with the corresponding number of hidden nodes and the value of Sigma.

Table 5: The best performance of the RBF Network with the corresponding number of hidden nodes and Sigma(σ).

Type of features used in the training	Permission features
# Hidden nodes	380
Sigma(σ)	0.09
CCR	97.20 %
TPR	94.80 %
TNR	98.40 %
FPR	1.60 %
FNR	5.20 %
AUC	96.60 %

4.4 Comparison of RBF against Other Common Machine Learning Techniques

This section discusses the performance of RBF compared to machine learning before IGR-based feature selection. Table 6 displays the performance measures of RBF, DT, NB, SVM, and MLP used in Android malware detection for testing data before applying IGR-based feature selection.

Table 6: A comparison of RBF performance against other machine learning before applying IGR-based feature selection.

	CCR	TPR	TNR	FPR	FNR	AUC
DT	96.00	95.60	96.20	3.80	4.40	95.90
NB	60.20	98.70	40.50	59.50	1.30	69.60
SVM	96.80	92.70	98.90	1.10	7.30	95.80
MLP	96.70	95.10	97.50	2.50	4.90	96.30
RBF	97.20	94.80	98.40	1.60	5.20	96.60

From Table 6, we can notice the following important remarks:

- In terms of CCR measure, DT, SVM, MLP, and RBF achieved good accuracy. The best accuracy (97.20%) was achieved by RBF while the worst accuracy (60.20%) was produced by NB.
- In terms of TPR measure, NB performed the best TPR (98.70%) while the worst TPR (92.70%) was produced by SVM.
- In terms of TNR measure, the best TNR nearly (99%) was accomplished by SVM and RBF while the worst TNR (40,50%) was produced by NB.
- In terms of FPR measure, the best FPR nearly (1%) was accomplished by SVM and RBF while the worst FPR was produced by NB (59.50%).
- In terms of FNR measure, the best FNR was accomplished by NB (1,30%) while the worst FNR was produced by SVM (7,30%).
- In terms of AUC measure, the best AUC was accomplished by RBF (96,60%) while the worst AUC was produced by NB (69,60%).

From Table 6, we can also observe that RBF accomplished the best results in most of the measures. That means RBF can detect correctly both malware and benign apps. On the other hand, NB classified most of the Android apps as malware apps since NB produced the highest TPR (98.70%). However, NB was not able to correctly identify the benign apps since NB produced the worse TNR (40.50%). Accordingly, NB performed the worst CCR (60.20%) and AUC (69.60%).

4.5 Comparison of RBF against Other Machine Learning After Applying IGR-Based Feature Selection

The performance of RBF and other machine learning were compared after applying the top thirty features selected by IGR in this section. Table 7 shows the performance of RBF, DT, NB, SVM, and MLP in Android malware detection for testing data after applying the top thirty features selected by IGR.

Table 7: A comparison of RBF performance against other machine learning with the top thirty features selected by IGR.

	CCR	TPR	TNR	FPR	FNR	AUC
DT	91.70	79.50	97.90	2.10	20.50	88.70
NB	58.60	99.70	37.50	62.50	0.30	68.60
SVM	91.60	80.30	97.30	2.70	19.70	88.80
MLP	91.80	80.10	97.70	2.30	19.90	88.90
RBF	92.00	79.50	98.40	1.60	20.50	89.00

From Table 7 we can observe the following points:

- In terms of CCR, all RBF, DT, SVM, and MLP achieved good accuracy but not more than (92%). The best accuracy was achieved by RBF while the worst accuracy was produced by NB (58.60%).
- In terms of TPR, NB performed the best TPR (99.70%) while the worst TPR was produced by RBF and DT (79.50%).
- In terms of TNR, the best TNR was accomplished by RBF (98.40%) while the worst TNR was produced by NB (37.50%).
- In terms of FPR, the best FPR was accomplished by RBF (1.60%) while the worst FPR was produced by NB (62.50%).
- In terms of FNR, the best FNR was accomplished by NB (0.30%) while the worst FNR was produced by RBF and DT (20.50%).
- In terms of AUC, the best AUC was accomplished by RBF (89%) while the worst AUC was produced by NB (68.60%), as shown in Table 7.

From Table 7, we can observe NB classified most of the Android apps as malware apps since NB produced the highest TPR (99.70%). However, NB was not able to correctly identify the benign apps since NB produced the worse TNR (37.50%). Accordingly, NB performed the worst CCR (58.60%) and AUC (68.60%). On the other hand, RBF accomplished the best results in most of the measures. This indicates that RBF can detect correctly both malware and benign apps.

5. Conclusions and Future Work

In Android malware detection, most of the conventional methods are unable to make decisions accurately and perfectly on whether the new Android apps are malware or benign. This study aims to develop intelligent Android malware detection using Radial Basis Function Network and information gain ratio in order to overcome the limitations of the conventional Android malware detection approaches. The results demonstrated RBF accomplished better results in most of the measures compared with other common machine learning techniques. The suggested intelligent RBF-based Android malware detection method can be used to contribute to detect the newly developed Android malware applications in order to protect Android users and devices from Android malware applications.

Although all objectives established for this study have been accomplished, there are a few limitations. Due to time constraints, in our study, we only used hold-out validation for evaluating the RBF network. K-fold cross-validation can be also used for more validation. Besides, centroid values of RBF are generated randomly in our study. K-Means clustering algorithm can be used to find the best values of the centroid in order to produce better performance.

References

[1] Kaspersky. IT threat evolution Q1 2020. Statistics. Accessed: Nov. 19, 2020. [Online]. Available: https://securelist.com/it-threat-evolution-q1-2020-statistics/96959/

[2] Buchanan, W. J., Chiale, S., Macfarlane, R.: *A methodology for the security evaluation within third-party Android Marketplaces*. Digital Investigation, 23, 88-98(2017).

[3] Dini, G., Martinelli, F., Matteucci, I., Petrocchi, M., Saracino, A., Sgandurra, D.: *Risk analysis of Android applications: A user-centric solution*. Future Generation Computer Systems, 80, 505-518(2018).

[4] Abdullah, T., Ali, W., Abdulghafor, R.: *Empirical Study on Intelligent Android Malware Detection based on Supervised Machine Learning*. International Journal of Advanced Computer Science and Applications (IJACSA), 11(4), 215-224(2020).

[5] Wang, W., Li, Y., Wang, X., Liu, J., Zhang, X.: *Detecting Android malicious apps and categorizing benign apps with ensemble of classifiers*. Future

generation computer systems, 78, 987-994(2018).
[6] Idrees, F., Rajarajan, M., Conti, M., Chen, T. M., Rahulamathavan, Y.: *PIndroid: A novel Android malware detection system using ensemble learning methods*. Computers & Security, 68, 36-46 (2017).
[7] Yerima, S. Y., Sezer, S., McWilliams, G.: *Analysis of Bayesian classification-based approaches for Android malware detection*. IET Information Security, 8(1), 25-36(2014).
[8] Yu, H., Xie, T., Paszczynski, S., Wilamowski, B. M.: *Advantages of radial basis function networks for dynamic system design*. IEEE Transactions on Industrial Electronics, 58(12), 5438-5450(2011).
[9] Sharma, A., Dash, S. K.: *Mining API calls and permissions for Android malware detection*. In Cryptology and Network Security. Cham, Switzerland: Springer Int., pp. 191–205(2014).
[10] Chan, P. P., Song, W. K.: Static detection of Android malware by using permissions and API calls. In Proc. Int. Conf. Mach. Learn. Cybern., Lanzhou, pp. 82–87(2014).
[11] Wang, W., Wang, X., Feng, D., Liu, J., Han, Z., Zhang, X.: *Exploring permission-induced risk in android applications for malicious application detection*. IEEE Transactions on Information Forensics and Security, 9(11), 1869-1882(2014).
[12] Cen, L., Gates, C. S., Si, L., Li, N.: *A probabilistic discriminative model for android malware detection with decompiled source code.* IEEE Transactions on Dependable and Secure Computing, 12(4), 400-412(2014).
[13] Abdulla, S., Altaher, A.: *Intelligent Approach for Android Malware Detection*. KSII Transactions on Internet and Information Systems, 9(8): 2964 – 2983(2015).
[14] Yuan, Z., Lu, Y., Xue, Y.: *Droiddetector: android malware characterization and detection using deep learning*. Tsinghua Science and Technology, 21(1), 114-123 (2016).
[15] Altaher, A.: *An improved Android malware detection scheme based on an evolving hybrid neuro-fuzzy classifier (EHNFC) and permission-based features*. Neural Computing and Applications, 28(12), 4147-4157(2017).
[16] Varsha, M. V., Vinod, P., & Dhanya, K. A.: *Identification of malicious android app using manifest and opcode features*. Journal of Computer Virology and Hacking Techniques, 13(2), 125-138(2017).
[17] Ali, W.: *Hybrid Intelligent Android Malware Detection Using Evolving Support Vector Machine ased on Genetic Algorithm and Particle Swarm Optimization*. International Journal of Computer Science and Network Security (IJCSNS), 19(9), 15-28 (2019).
[18] Genome. Android Malware Genome Project. Accessed: February. 14, 2021. [Online]. Available: http://www.malgenomeproject.org
[19] Contagio. Contagio Mobile: mobile malware mini dump. Accessed: February. 14, 2021. [Online]. Available: http://contagiominidump.blogspot.co.uk
[20] GitHub. certtools. Accessed: Nov. 20, 2020. [Online]. Available: https://github.com/certtools/malware_name_mapping
[21] Google Play. Google Play Store. Accessed: Nov. 20, 2020. [Online]. Available: https://play.google.com/store?hl=en
[22] VirusShare. VirusShare.com. Accessed: Nov. 20, 2020. [Online]. Available: https://virusshare.com
[23] GitHub. Mitchellkrogza. Accessed: Nov. 20, 2020. [Online]. Available:https://github.com/mitchellkrogza/The-Big-List-of-Hacked-Malware-Web-Sites
[24] TheZoo. The Zoo aka Malware DB. Accessed: Nov. 20, 2020. [Online]. Available: http://ytisf.github.io/theZoo
[25] Virusbay. Virusbay.com. Accessed: Nov. 20, 2020. [Online]. Available: https://beta.virusbay.io/
[26] Dasmalwerk. DAS MALWERK // malware samples. Accessed: Nov. 20, 2020. [Online]. Available: https://dasmalwerk.eu/
[27] Figshare. Android malware dataset for machine learning 1. Accessed: Nov. 19, 2020. [Online]. Available: https://figshare.com/articles/Android_malware_dataset_for_machine_learning_1/5854590/1
[28] Yerima, S. Y., & Sezer, S.: Droidfusion: *A novel multilevel classifier fusion approach for android malware detection*. IEEE transactions on cybernetics, 49(2), 453-466(2018).
[29] Ali, W.: *Phishing Website Detection based on Supervised Machine Learning with Wrapper Features Selection*. International Journal of Advanced Computer Science and Applications (IJACSA), 8(9), 72-78(2017).
[30] Ali, W., & Ahmed, A. A.: *Hybrid intelligent phishing website prediction using deep neural networks with genetic algorithm-based feature selection and weighting*. IET Information Security, 13(6), 659-669(2019).
[31] Yerima, S. Y., Sezer, S., Muttik, I. *High accuracy android malware detection using ensemble learning*. IET Information Security, 9(6), 313-320(2015).