DOI: 10.21917/ijsc.2016.0186

CERTAIN INVESTIGATIONS ON VARIOUS ALGORITHMS THAT IS USED TO CLASSIFY MALWARE AND GOODWARE IN ANDROID APPLICATIONS

B.P. Sreejith Vignesh[1] and M. Rajesh Babu[2]
[1]*Department of Computer Science, Bharathiar University, India*
E-mail: [1]sreejithvigneshbp@gmail.com
[2]*Department of Computer Science and Engineering, Karpagam College of Engineering, India*
E-mail: [2]drmrajeshbabu@gmail.com

Abstract

In recent trends, the mobile devices play a very vital role in day to day activities of human beings. Google Android OS appeared lately i.e., in September 2008 in mobile market and gains more popularity. Google Android OS offers more flexibility for the users by offering N number of free downloadable applications to the users, which in turn gets changed as the superlative target for the attackers. As a result, many android applications that may contain the malware applications which are capable of stealing privacy information of users are available in market as a (.apk) file. The attackers started to target uneducated people and started stealing the information using applications. These applications request user to allow set of permissions during installation. For a new user it is difficult to identify the set of permissions that are harmful. This could be an advantage for malware intruders to access the data or infect the mobile device by introducing malware applications. Therefore, android malware detection various algorithms algorithm and Machine learning approaches is proposed to classify malware and goodware applications by analyzing the permission features.

Keywords:

Android, Malware Application, Principal Component Analysis, Cuckoo Search, Pearson Correlation Coefficient

1. INTRODUCTION

Since the invention of first Android phone in October 2008 the devices such as smart phones, PDA's and Tablets become popular. These mobile devices offer huge services and over 200,000 applications as of May 2011. There are 2,248,961 [1] apps available in the market which includes 12 % of low quality apps as on July 4' 2016.User can download application from android market. Free Apps as well as paid versions are also available in the android market. These applications had been installed in the devices for 4.5 billion times [2]. Android phones have been sold more than 60% of overall smart phones available [3]. Android possessed 82.8% of the market share in 2015 reported in global survey of the OS smart phone market, implying that the growth of the Android is increased when compared to other OS. Today The Android platform is fastest growing market and faces some critical risk.

Using machine learning algorithm Android Malware applications can be detected. There are two issues in implementing machine learning algorithm in detecting malware applications. First one includes extraction of feature representation of an application. Second, the data set may include normal and malicious application, so machine learning classifier is chosen to train the data set. First issue includes, extracting permission features from .apk files. To deal with second issue, decision tree classifiers are used in training the given normal and malware applications.

2. RELATED WORK

Au et al. [6] had made a comparison of features of current smart phone permission systems. It is clear that android smart phone permissions provide more information to the user than any other smart phone OSs. Android permissions have grouped applications that have similar permission features [7]. Wagner, D et al. [8] in their work of Android permissions Demystified, explained about more permission requests to the developers.

Enck et al. [10] proposed a security service system; called Kirin, using a set of predefined security rules it certifies the application on installation. They analyzed the malicious behaviors by analyzing the defined the rules that are configured with permission. AdDroid system separates privilege to Android platform from advertising framework [11] to prevent the library to access sensitive information of the application.

Aubrey-Derrick Schmidt et al. [6] contribute the detection of malware in two steps. First, in static analysis it extracts their function calls in Android environment using the command readel. List of Function calls are compared with malware executable for classification. It is done using PART, Prism and Nearest Neighbor Algorithms. Second, a collaborative malware detection approach is implemented to extend these results.

Asaf Shabtai et al. [12] proposed Andromal framework on Android mobile devices for detecting malware application. The framework recognizes a Host-based Malware Detection System that constantly monitors numerous features and events that are obtained by the mobile device, and then applies Machine Learning anomaly detectors to classify the collected data as benign or abnormal malicious.

Tanzirul Azim et al. [14] and Shuai Hao [15] proposed Android based dynamic analysis frameworks for emphasizing coverage and target analysis. In [15], an activity coverage rate has been included as major work for detecting malware on mobile devices. The achieved activity rate coverage is above 60%. It [14] describes coded framework for Android UI interaction to allow for flexibility and easy access.

Bose et al. [13] proposed a detection framework based on behavior by analyzing the ordering of actions performed by application. Malicious behavior is segregated from benign behavior by training SVM. Evaluation results produce 96% accuracy for both simulated mobile malwares and real world. Enck et al. [14] proposed a method for dynamic analysis of android applications. This method attempts to detect malicious application as well as detects the application that leaks the user private information without notifying the user. Intelligent Black Box Execution testing and fuzzy testing is involved in analyzing the android applications.

3. PERMISSION BASED ANALYSIS

In installing android applications, permission has a major role to deal with. App asks user permission for accessing system related entities while installing an android app and these permissions are stored in manifest.xml file during installation. These permissions restricts the application from accessing the confidential information, intruding vulnerabilities [16] etc. The user cannot select the individual permissions, instead they can either allow or deny the installation of application. The accessing of resources is based on these permissions in android applications. Based on the permissions listed in manifest.xml, malicious activities can be detected.

3.1 ANDROID APPLICATION PERMISSIONS

android.permission.Internet: Some application asks for Internet permissions upon installing. Not every application requires Internet to run. If the application that does not need Internet is requested for Internet permission, it is considered as an abnormal application. Using this, malware can send user privacy information to the applications server. Internet Permission request is one of the anonymous permission request.

android.permission.ChangeConfiguration: The application seeks request for changing the configuration of system files. By allowing this permission request, the application can change any stored files and it can be lost.

android.permission.Write_Sms: Without notifying the user, the application can write SMS by allowing this request.

android.permission.Send_Sms: By allowing this permission request, the application can send SMS without notifying the user which may cost to the user.

android.permission.Call_Phone: Some applications request for this permission request even the app don’t have anything to do with making a call. If the user allows this request the application itself can make a call without notifying the user. The user does not know how the money was reduced.

android.permission.Camera: Allowing this permission request, the application can take photos or record videos without notifying to the user.

android.permission.Location_Data: Allowing this permission request, application can track the current device location.

android.permission.Bluetooth: Allowing this permission request, application can transfer data to other devices.

android.permission.AccessNetwork_State: Allowing this permission request, application can check network connection status and also it can access the information how it is connected (Wi-Fi/3g/4g).

android.permission.Read_Phone_State: Allowing this permission request, the application can access the device serial numbers and identifiers and with added Internet permission unique profile on remote server for that device can be created. Phone identifiers can also be sent to advertisement servers.

android.permission.Access_Wifi_State: Allowing this permission request, the application can view nearby SSIDs.

4. FEATURE EXTRACTION

The following steps are followed to extract the data from selected android application.

1. Download the Malware application and Goodware application from the android play store.
2. Perform decompression to extract the contents of the downloaded application using .apk tool.
3. Permissions request features for each application are then retrieved as shown in Fig.1 and Fig.2.
4. Construct the dataset with the retrieved features in ARFF format [17].
5. AndroidManifest.xml file is used to process the extracted data.

5. CUCKOO SEARCH

Cuckoo search is an efficient heuristic method introduced by Xin-sheet et al [19]. It represents the parasitical training feature of cuckoos. Every parent cuckoos does not breed its eggs. They depend on other cuckoo birds to host its egg. In the process, the bird selects a nest randomly and lays the eggs there. The bird owning the nest, after finding the egg may damage the egg or destroy its own nest and build a new one. Then it may change the color or shape of its egg to avoid detection of the nest.

The cuckoo search algorithm describes the breeding behavior of cuckoos which can be explained in three rules: 1. A Cuckoo lays one egg during one particular moment. 2. Nests that carry high quality eggs will be sent for nest production. 3. The probability of finding the egg by a cuckoo bird in its nest is $P(0,1)$. The host bird if discovers any new egg in its nest, it may either destroy the egg or the nest and modifies or build a new nest.

5.1 BASIC CUCKOO SEARCH ALGORITHM

1. Construct n number of nests
2. Follow the below process until the condition is met
   - Select a cuckoo randomly and bring it to the nest using Levy flight distribution
   - Evaluate the quality of cuckoo (C_q)
   - Select a nest randomly
   - Evaluate the nest quality (N_q)
   - If $C_q > N_q$, find new nest and replace N
   - The Probability P of weaker nests is replaced with newer ones.
   - Evaluate the quality of nests N_q and keep the best ones.
   - Rank the nest based on their quality.
3. Calculate the result.

6. FUZZY K-MEANS CLUSTERING

The FKM clustering algorithm partitions data sets into k clusters S_l (l = 1, 2,..., k) and each cluster is far located from other clusters. Each cluster is associated with cluster center C_l. The relationship between S_l and C_l is fuzzy. $u_{i,j}$ [0,1] denotes the degree of data set X_i and center of the cluster C_j. Cluster of data set is denoted as $S = \{X_i\}$. Fuzzy k-means is based on reducing the following misrepresentation.

$$J = \sum_{j=1}^{k}\sum_{i=1}^{N} u_{i,j}^{m} d_{ij} \tag{1}$$

N denotes the number of data points, k is the number of clusters, m denotes fuzzy parameter and d_{ij} denotes Euclidean distance squared between X_i and C_j. The following statement must be satisfied for $u_{i,j}$,

$$\sum_{j=1}^{k} u_{i,j} = 1 \text{ for } i = 1 \text{ to } N \tag{2}$$

The important step in FKM is assigning set of representative vectors to the best one by partitioning data points.

Algorithm

Step 1: Initiate set of cluster centers $SC_0 = \{C_j(0)\}$ and ε value. Set p = 1.

Step 2: Calculate d_{ij} for j = 1 to k and i = 1 to N. Update $u_{i,j}$

$$u_{i,j} = \left(\left(d_{ij}\right)^{\frac{1}{m-1}} \sum_{j=1}^{k} \left(\frac{1}{d_{il}}\right)^{\frac{1}{m-1}} \right)^{-1} \tag{3}$$

7. MACHINE LEARNING APPROACH

Machine Learning approaches are used to classify malware applications in Android platform. Some of the classification algorithms are C4.5 and ANFIS. The description of mentioned algorithms is as follows and best method is chosen for classification.

7.1 C4.5 ALGORITHM

C4.5 (also known as J48) is the most commonly used classifier method. It is a tree construction algorithm based on divide and conquers strategy. It uses Information gain to sort the data. This method takes collection of cases as input values where each case belongs to small number of class described by its attribute values. With these values C4.5 classifier predicts the belonging class of a new case.

Step 1: Let P be the set of cases. Compute the weighted frequency of cases in P whose class is C_i, $f(C_i, P)$ for $i \in [1, N_{class}]$.

Step 2: If all cases in P are from same class C_j, then the node is a leaf. The classification error of leaf may be the weighted sum of cases whose class is not C_j in P.

Step 3: If P has two or more classes that belongs to a case, compute the information gain for each attribute. If the attribute is discrete, information gain results in dividing cases in P into sets with specific attribute values. If the attribute is continuous it divides P into two subsets as cases with attribute value less than threshold and greater than local threshold value.

Step 4: Select the attribute for testing that possess high information gain.

Step 5: If the selected attribute is continuous, threshold is selected as the highest value (which is less than local threshold).

Step 6: Testing selected attribute produces sets P_1, P_2,..., P_m where m is the number of children of the decision node m = 2. If the selected attribute is continuous and $m = k$ if the attribute is discrete with k known values.

Step 7: The child node is set as leaf node if P_i is empty for i = [1,m] and classification error is zero.

Step 8: If P_i is not empty, perform divide and conquer on set P_i of those cases in P with unknown value of corresponding attribute. Unknown value of selected attribute will be replicated by the weights of corresponding known value of selected attribute proportionally to the cases in P_i over cases in P.

Step 9: Summing the errors in child node, the classification error is calculated. If the resulting error is larger than error of classifying all nodes in P, the decision node is set as leaf and sub-trees are discarded.

7.2 ADAPTIVE NEURO-FUZZY INFERENCE SYSTEMS [ANFIS]

An ANFIS network is implemented for classification of malware and goodware applications. In ANFIS, first model fuzzy inference system contains the fuzzy model [21] proposed by Takagi, Sugeno and Kang to generate fuzzy rules by formulating from an input output data set.

Consider there are two inputs and one output in the fuzzy interface. Takagi and Sugeno's type rule contains if-then rules of [21] as follows:

If u is A and v is B then q is $f(u,v)$

where, A and B are the fuzzy sets in the backgrounds and $d = f(u,v)$ is a crumbly function in the consequent. $f(u,v)$ is a polynomial for the input u and v. When $f(u,v)$ is a constant, a zero order Sugeno fuzzy model is formed. The two rules are stated as,

Rule 1: If u is A_1 and v is B_1 then $f_1 = p_1u + q_1v + r_1$

Rule 2: If u is A_2 and v is B_2 then $f_2 = p_2u + q_2v + r_2$

Then the type-3 fuzzy inference [21] is used. In this system each rule adds the constant term with input variables to produce an output which is linear combination input. The final output is the weighted average of each rule's output. Each layer functions are described below.

7.2.1 Layer 1:

Each and every node j is adaptive with a function node in Layer 1.

$$O_j^1 = \mu_{A_j}(u) \tag{4}$$

where, v is the input to node j, μ (A_j) is membership function of A_j and A_j is the variable associated with this node function. Generally $\mu_{A_j}(u)$ is chosen between 0 and 1,

$$\mu_{A_j}(u)=\frac{1}{1+\left[\left(\frac{u-c_j}{a_j}\right)^2\right]^{b_j}} \text{ (or) } \mu_{A_j}(u)=\exp\left\{-\left(\frac{u-c_j}{a_j}\right)^2\right\} \quad (5)$$

where, u is the input and $\{a_j, b_j, c_j\}$is the parameter set.

7.2.2 Layer 2:

In this layer, each of the fixed nodes calculates the firing strength of the rule and each node produce the output (i.e. product of input entropy of file).

$$O_j^2=\omega_j=\mu_{A_j}(u)\cdot\mu_{B_j}(v),\ j = 1, 2 \quad (6)$$

7.2.3 Layer 3:

In this layer, the ratio of node is calculated with weight of the each node. It produces normalize firing strength of node.

$$O_j^3=\bar{\omega}_j=\frac{\omega_j}{\omega_1+\omega_2},\ j = 1, 2 \quad (7)$$

7.2.4 Layer 4:

Every node j in this layer is adaptive with a node function.

$$O_j^4=\bar{\omega}_j f_j=\bar{\omega}_j\left(p_j u+q_j v+r_j\right),\ j = 1, 2 \quad (8)$$

where, $\bar{\omega}_j$ is the output of Layer 3 and $\{p_j, q_j, r_j\}$is the parameter set.

7.2.4 Layer 5:

Summation of all incoming signals is the overall output that is computed by single fixed node.

$$O_j^5=overall\ output=\sum_j\bar{\omega}_j f_j=\frac{\sum_j\omega_j f_j}{\sum_j\omega_j} \quad (9)$$

Based on the final result of input entropy, the ANFIS classifier predicts the class accurately where it belongs. As a result, if the value is greater than 0 or with some floating values then those applications are considered to be malware else if the result is zero then it is goodware application.

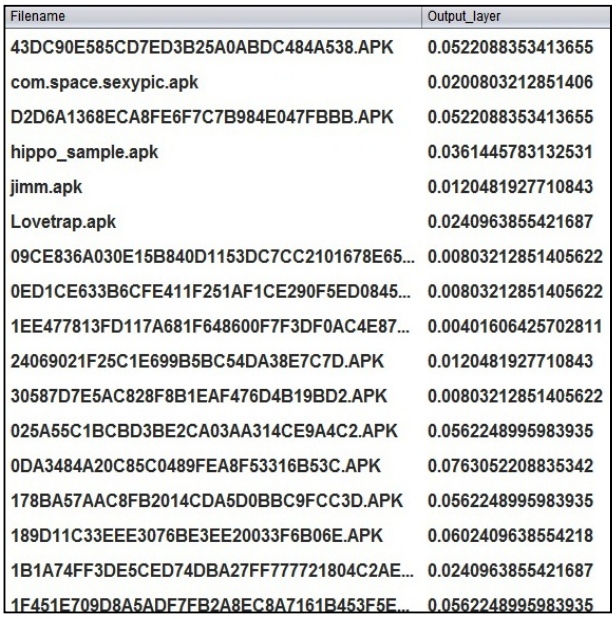
8. COMPARISON PARAMETER VALUES

The performance of the proposed work is compared with the existing system using the following 4 parameters Precision, Recall, F-Measure and Accuracy.

Table.1. Parameter Comparison table

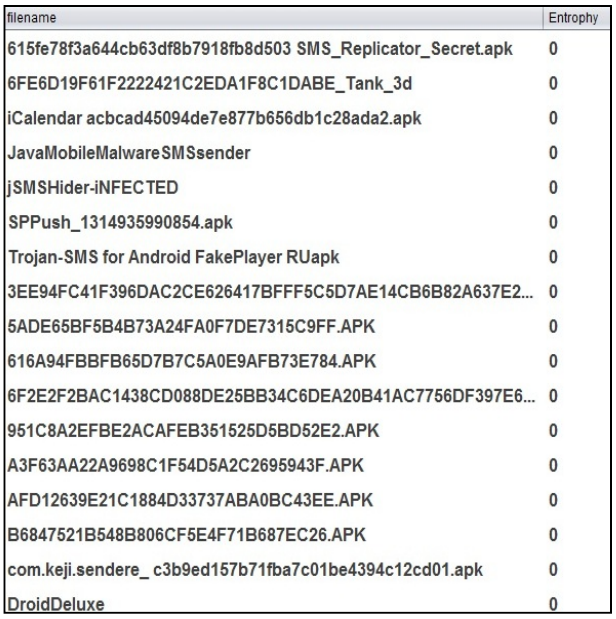
Parameters	Existing DT with GA	Proposed ANFIS
Precision	67	70
Recall	64	68.18
F-Measure	68	73.5
Accuracy	72	75

From the above table we could get a slight improvement in all the measurement parameters in the proposed ANFIS based system when compared with the existing system. In specific the variation of Recall and F-Measure has recorded a considerable improvement. So in further investigations the F-Measure value is taken as a benchmarking point and from there the further investigations are carried forward, considering that this will enable us to perform the backtracking to overcome the minute changes and improve the performance level. From that particular point a drift in performance is calculated in which the accuracy of the application as good ware is further improved.



Filename	Output_layer
43DC90E585CD7ED3B25A0ABDC484A538.APK	0.0522088353413655
com.space.sexypic.apk	0.0200803212851406
D2D6A1368ECA8FE6F7C7B984E047FBBB.APK	0.0522088353413655
hippo_sample.apk	0.0361445783132531
jimm.apk	0.0120481927710843
Lovetrap.apk	0.0240963855421687
09CE836A030E15B840D1153DC7CC2101678E65...	0.00803212851405622
0ED1CE633B6CFE411F251AF1CE290F5ED0845...	0.00803212851405622
1EE477813FD117A681F648600F7F3DF0AC4E87...	0.00401606425702811
24069021F25C1E699B5BC54DA38E7C7D.APK	0.0120481927710843
30587D7E5AC828F8B1EAF476D4B19BD2.APK	0.00803212851405622
025A55C1BCBD3BE2CA03AA314CE9A4C2.APK	0.0562248995983935
0DA3484A20C85C0489FEA8F53316B53C.APK	0.0763052208835342
178BA57AAC8FB2014CDA5D0BBC9FCC3D.APK	0.0562248995983935
189D11C33EEE3076BE3EE20033F6B06E.APK	0.0602409638554218
1B1A74FF3DE5CED74DBA27FF777721804C2AE...	0.0240963855421687
1F451E709D8A5ADF7FB2A8EC8A7161B453F5E...	0.0562248995983935

(a) Malware



filename	Entrophy
615fe78f3a644cb63df8b7918fb8d503 SMS_Replicator_Secret.apk	0
6FE6D19F61F2222421C2EDA1F8C1DABE_Tank_3d	0
iCalendar acbcad45094de7e877b656db1c28ada2.apk	0
JavaMobileMalwareSMSsender	0
jSMSHider-iNFECTED	0
SPPush_1314935990854.apk	0
Trojan-SMS for Android FakePlayer RUapk	0
3EE94FC41F396DAC2CE626417BFFF5C5D7AE14CB6B82A637E2...	0
5ADE65BF5B4B73A24FA0F7DE7315C9FF.APK	0
616A94FBBFB65D7B7C5A0E9AFB73E784.APK	0
6F2E2F2BAC1438CD088DE25BB34C6DEA20B41AC7756DF397E6...	0
951C8A2EFBE2ACAFEB351525D5BD52E2.APK	0
A3F63AA22A9698C1F54D5A2C2695943F.APK	0
AFD12639E21C1884D33737ABA0BC43EE.APK	0
B6847521B548B806CF5E4F71B687EC26.APK	0
com.keji.sendere_ c3b9ed157b71fba7c01be4394c12cd01.apk	0
DroidDeluxe	0

(b) Goodware

Fig.1. Malware identification via ANFIS

From the above parameter comparison chart we could find the overall accuracy of the application is constantly improving in terms of good ware when an application is checked with in terms of Ensemble ANFIS algorithm implementation.

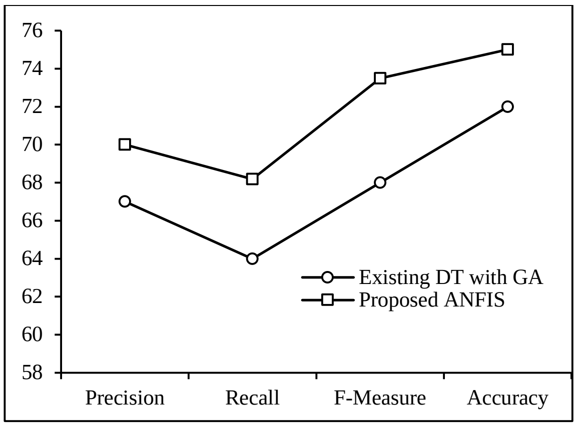


Fig.2. Parameter Comparison Chart

9. RESULTS AND DISCUSSION

The results obtained from various existing techniques are compared and results are discussed below. The integration method is implemented using java netbeans 7.1 and MYSQL 5.1. The application files (.apk) are downloaded from android play store and collected in a dataset. The detection of malware application is done in four phases: Feature analysis, Feature extraction, Feature selection and Classification. The parameters that are used in comparison are precision, recall, accuracy, detection time, detection rate, false positive rate and CPU utilization.

10. CONCLUSION

Mobile phones become the most familiar communication device because of its mobility, computational speed and Internet access. They have started to replace traditional computers because of its features. Android has gathered attention and its necessity is growing exponentially each year. Due to its popularity malware writers possess a threat to android platform by introducing malware applications in the android store which is a trusted one for downloading applications. In general only alpha and beta testing is carried out before uploading the application in the google play store, it never checks for the permission based features. Hence an approach is proposed using machine learning methods to detect the malware applications. The experimental results show that the ANFIS method detects malware application with high accuracy. The detection rate is also high comparing with other techniques. This method can efficiently detect more malware applications in the android smart phones.

REFERENCES

[1] AppBrain, Available at: http://www.appbrain.com/stats/number-of-android-apps

[2] Android: Momentum, Mobile and More at Google I/O, Available at: http://googleblog.blogspot.com/2011/05/android-momentum-mobile-and-more-at.html

[3] Gartner Says Sales of Mobile Devices Grew 5.6 Percent in Third Quarter of 2011; Smartphone Sales Increased 42 Percent, Available at: http://www.gartner.com/it/page.jsp?id=1848514.

[4] Asaf Shabtai, "Malware Detection on Mobile Devices", *Proceedings of 11th International Conference on Mobile Data Management*, pp. 289-290, 2010.

[5] Rafael Fedler, Julian Schutte and Marcel Kulicke, "On the Effectiveness of Malware Protection on Android: An Evaluation of Android Antivirus Apps", Technical Report, Fraunhofer AISEC, pp. 1-35, 2013.

[6] Kathy Wain Yee Au, Yi Fan Zhou, Zhen Huang, Phillipa Gill and David Lie, "Short Paper: a Look at Smart Phone Permission Models", *Proceedings of the 1st Workshop on Security and Privacy in Smart Phones and Mobile Devices*, pp. 63-67, 2011.

[7] David Barrera, H. Gunes Kayacik, P.C. Van Oorschot and Anil Somayaji, "A Methodology for Empirical Analysis of Permission-Based Security Models and Its Application to Android", *Proceedings of the 17th Conference on Computer and Communications Security*, pp. 73-84, 2010.

[8] Adrienne Porter Felt, Erika Chin, Steve Hanna, Dawn Song and David Wagner, "Android Permissions Demystified", *Proceedings of the 18th Conference on Computer and Communications Security*, pp. 627-638, 2011.

[9] Peter Hornyack, Seungyeop Han, Jaeyeon Jung, Stuart Schechter and David Wetherall, "These Aren't the Droids You're Looking for: Retrofitting Android to Protect Data from Imperious Applications", *Proceedings of the 18th Conference on Computer and Communications Security*, pp. 639-652, 2011

[10] Justin Sahs and Latifur Khan, "A Machine Learning Approach to Android Malware Detection", *European Intelligence and Security Informatics Conference*, pp. 141-147, 2012

[11] J. Zico Kolter and Marcus A. Maloof, "Learning to Detect and Classify Malicious Executables in the Wild", *Journal of Machine Learning Research*, Vol. 7, pp. 2721-2744, 2006.

[12] G.J. Tesauro, J.O. Kephart and G.B. Sorkin, "Neural Networks for Computer Virus Recognition", *IEEE Expert*, Vol. 11, No. 4, pp. 5-6, 1996.

[13] Abhijit Bose,Xin Hu, Kang G. Shin and Taejoon Park, "Behavioral Detection of Malware on Mobile Handsets", *Proceeding of the 6th International Conference on Mobile Systems, Applications, and Services*, pp. 225-238, 2008.

[14] Tanzirul Azim and Iulian Neamtiu, "Targeted and Depth-First Exploration for Systematic Testing of Android Apps", *Proceeding of International Conference on Object Oriented Programming Systems Languages and Applications*, Vol. 48, No. 10, pp. 641-660, 2013.

[15] Shuai Hao, Bin Liu, Suman Nath, William G.J. Halfond and Ramesh Govindan, "PUMA: Programmable UI-Automation for Large-Scale Dynamic Analysis of Mobile Apps", *Proceeding of 12th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 204-217, 2014.

[16] Bruce C. Moore, "Principal Component Analysis in Linear Systems: Controllability, Observability and Model Reduction", *IEEE Transactions on Model Reduction and Automatic Control*, Vol. 26, No. 1, pp. 17-32, 1981.

[17] Myoung Soo Park, Jin Hee Na and Jin Young Choi, "PCA-Based Feature Extraction using Class Information", *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, Vol. 1, pp. 341-345, 2005.

[18] Yiteng Huang, Jacob Benesty and Jingdong Chen, “Using the Pearson Correlation Coefficient to Develop an Optimally Weighted Cross Relation based Blind SIMO Identification Algorithm”, *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3153-3156, 2009.

[19] Laura Elena Raileanu and Kilian Stoffel, “Theoretical Comparison between the Gini Index and Information Gain Criteria”, *Annals of Mathematics and Artificial Intelligence*, Vol. 41, No. 1, pp. 77-93, 2004.

[20] Zhang Yun, Zhou Quan, Sun Caixin, Lei Shaolan, Liu Yuming and Song Yang, “RBF Neural Network and ANFIS-based Short-Term Load Forecasting Approach in Real-Time Price Environment”, *IEEE Transactions on Power Systems*, Vol. 23, No. 3, pp. 853-858, 2008.

[21] Hao Ying, “Sufficient Conditions on Uniform Approximation of Multivariate Functions by General Takagi-Sugeno Fuzzy Systems with Linear Rule Consequent”, *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 28, No. 4, pp. 515-520, 1998.

[22] B.P. Sreejith Vignesh and M. Rajesh Babu, “Research Study on Various Malwares its Classification, Detection and Avoidance Techniques applied in Android Mobile Devices”, *International Journal of Applied Engineering Research*, Vol. 10, No. 20, pp. 20184-20187, 2015.