# Efficient Android Malware Scanner Using Hybrid Analysis

**K.A. Dhanya, T. Gireesh Kumar**

***Abstract--- Mobile Malicious applications are great threat to digital world as it is increasing tremendously along with benign applications. Main approaches for analysing the malware are static, dynamic and hybrid analysis. In this paper hybrid analysis is proposed with permission features accessed from applications statically, dynamic features like network activities, file system activities, cryptographic activities, information leakage etc. are dynamically accessed using Android Droid box and dynamic API calls are analysed using API Monitor tool. Separability assessment Criteria is used for relevant feature selection which had improved the performance. In this paper, hybrid features are used to characterize the malware along with learning algorithms such as Naïve Bayes, J48 and Random Forest. Random Forest classifier had produced TPR of 1, FPR of 0 with 77 best features.***

***Keywords--- Mobile Malware, Droid box, API Monitor, Hybrid Analysis, Machine learning.***

## I. INTRODUCTION

Smart phones are bringing new possibilities to our day today life and becomes more useful than personal computers. Portability, easy to use and small size etc. are the main features which attracts more users to smart phone which in turn attracts more attackers to the smart phones and give a way for introducing thousands of malicious Applications to the market [14]. Malicious features of this Apps are botnet, SMS Trojan, aggressive adware, ransomware etc., and are spread through official Application store or third party market even though they use bouncer to filter malicious Apps. These malicious Apps use clever ways for bypassing the existing security mechanism such as code obfuscations, repackaging, encryptions etc.

It is very difficult for deterministic approach to separate benign and malware applications due to the similarity in application structure [21]. Machine learning algorithms which performs supervised and unsupervised learning play a vital role in scanning mobile malware. There are mainly four types of feature set for malware scanning which are static, dynamic, hybrid and metadata. During static analysis [17] [18] [22], features like permissions, intents, opcodes, hardware components etc. are collected prior to installation and run time behavior is analyzed during dynamic analysis [19] [20]. Static analysis are weak to identify obfuscations such as encryption, dead code injection, dynamic code loading, network activity and modification of objects at run time [16]. In-the box dynamic analysis fails to intercept app data, communications but high level semantics can be analyzed. Out-of- the-box dynamic analysis uses complex emulators improves security by isolating malware apps but which may mislead dynamic analysis.

Hybrid analysis combines the advantages of the both static and dynamic approaches and addresses the issues related to the static analysis such as inability to detect dynamic code loading, obfuscated and zero-day malware, also address the issues of dynamic analysis as it cannot analyses all execution paths of application. During hybrid analysis both static and dynamic features are collected which depicts both application code features and run time behaviors.

Feature selection methods plays a vital role in malware scanning as the classification models are poor for features with higher dimension. Feature selections filter irrelevant permissions from the feature vector of over privileged and under privileged applications. Dangerous API calls are brought in to picture by the feature selection. Regarding the classification algorithm, ensemble learners outperforms naïve classifiers in terms of accuracy but computationally expensive. To validate this, with the aid of bench mark Dataset in mobile malware, in this paper we had proposed supervised learning architecture for scanning mobile malware such as Naïve Bayes[24], SVM [25], J48[26] and Random Forest [27]. By performing hybrid analysis on the dataset, a large number of features are extracted and filtered by using scatter/ Separability assessment method based on relevance.

The outline of the paper is given below. Section2 includes the survey of the existing malware analysis and detection techniques. Section3 is about design and implementation of the proposed system, Section4 compares the experimental results and finally conclude the paper.

## II. LITERATURE SURVEY

Zhenlong Yuan etal [7] proposed a hybrid malware analysis with 120 permissions, 59 sensitive API calls and 13 network activities such as network and file inputs/ outputs, information leaks, mobile phone calls etc. For each mobile application a binary feature vector is created based on the presence and absence of particular feature in the application, Deep learning is used for malware detection produced accuracy of 96.76%which outperforms shallow machine learning methods. The main shortcomings of this system are not addressing discrete and semantic based features and deep learning model was not optimised with feature vectors of different variants of mobile malwares.

Mengyu et al. [9] in 2016 proposed a shallow machine learning approach with permissions and API calls.

**Manuscript received February 01, 2019**

**K.A. Dhanya,** TIFAC-CORE in Cyber Security, Amrita School of Engineering, Coimbatore, Tamil Nadu, Amrita Vishwa Vidyapeetham, India. (e-mail: dhannyashibu@gmail.com)

**T. Gireesh Kumar,** TIFAC-CORE in Cyber Security, Amrita School of Engineering, Coimbatore, Tamil Nadu, Amrita Vishwa Vidyapeetham, India. (e-mail: t_gireeshkumar@cb.amrita.edu)

They defined 104 binary and numerical permission and 654 binary and numerical API calls features. Binary permission denotes existence of call to each permission control function and numerical count of calls to all functions controlled by each permission. Binary and numeric API calls denotes presence and counts to corresponding API calls. Relevant features are filtered and learned using Random Forest, Support vector machines and Artificial neural networks.

Dong Jie et al [8]. proposed a mobile malware detection system called Droid matusing static features like permissions, deployment of components, intent message passing, API calls etc.

Features are extracted, clustered using k-means algorithm and the best cluster is selected using Singular Valued Decomposition (SVD) Algorithm. DroidMat is effective in detecting variants of Android malware except Base Bridge and Droid Kungfu.

It is scalable and efficient due to the absence of environmental setup and manual efforts for dynamic analysis.

Shree Garg et al [10] proposed network based android malicious detection system with dynamic analysis for collecting network features such as DNS details, HTTP and TCP details etc.

This system is strong enough to detect mobile bots, Distributed DOS, sending premium messages and stealing credit card information etc. Extracted features are learned using Decision Trees, Bayesian Networks, Random Forests, Logistic Regression and K-Nearest Neighbours.

Random Forest produced best accuracy and highly scalable with less time complexity. Network traffic reflects malicious behaviour and this approach strongly identifies abnormal behavior on network with accuracy of 95.4% without hardware and software dependence.

Ali Feizollah et al. [11] scrutinize the various features of Android malware.

Machine learning methods are effective with good features with high variance among malicious and benign application.

In this survey the author analysed 100 papers and categorised features into four groups which are static, dynamic, hybrid and metadata features.

## III. PROPOSED SYSTEM

A robust malware scanning system proposed in this paper and its architecture depicted in Figure1. Detailed explanation of each step is given in following subsections.

### *3.1 Dataset Collection*

200 Benign and 200 malware application comprises the dataset for experimentation. Benign applications are randomly collected from Google play store [4] and verified in Virus Total [2] scanner and ensured that they are legitimate.

Malware applications are taken from the Drebin Dataset. [5].

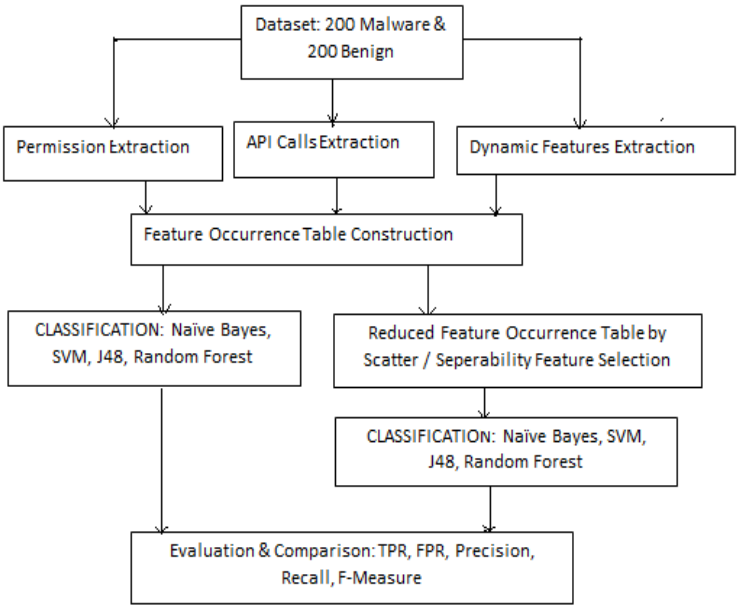


**Fig 1: Proposed Architecture**

### *3.2 Feature Extraction*

Permission Extraction using static Analysis: Manifest files of apk are extracted using Apktool [1], [15] and converted into readable format using AXMLPrinter2. Permissions are extracted from Manifest file using XML Parser and Boolean feature vector is constructed for each apk. Occurrence of each permission in Benign and malware apks sets are separately represented in a bar chart (X-axis represents permissions and y axis represents no of apks using the corresponding permission).

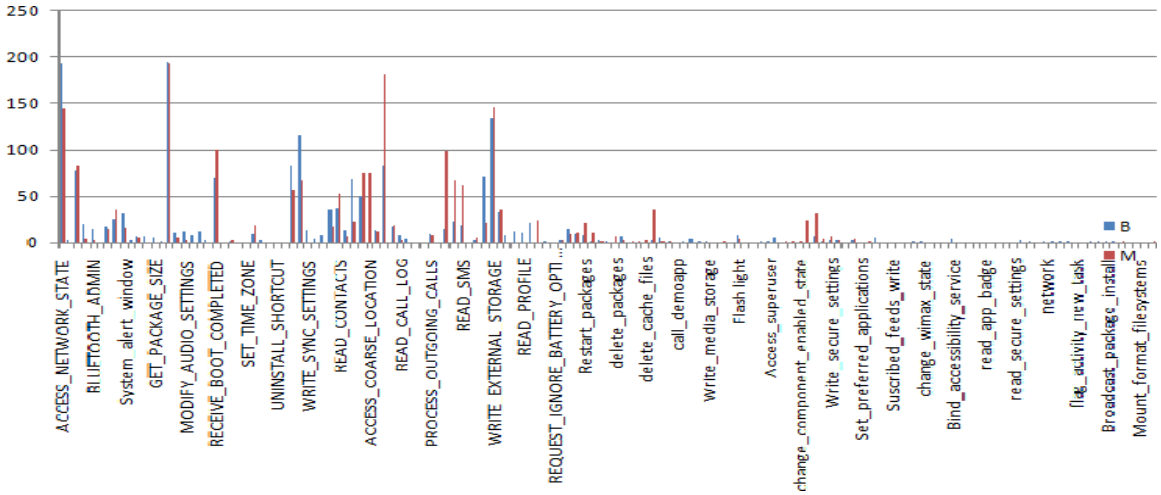


**Fig 2: Occurrence of Permissions in malware and Benign Applications**

API Calls Extraction using dynamic Analysis: New repackaged apk is generated from old using API Monitor tool. The new apk ran in droidbox and API calls are collected using adb logcat command. Occurrence of each API calls is used for constructing feature vector.

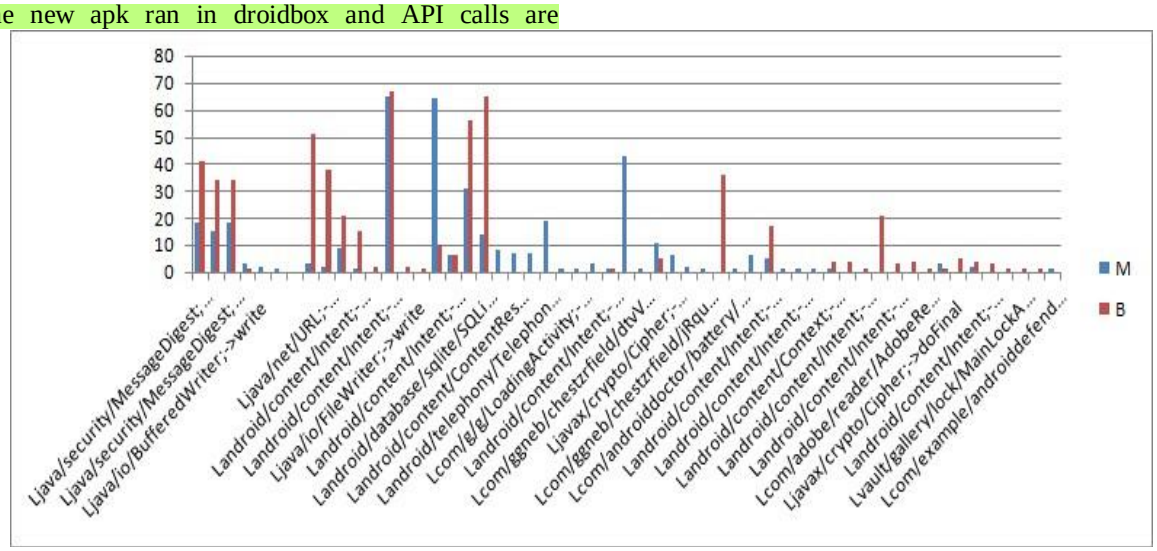


**Fig 3: Occurrence of API calls in malware and Benign Applications**

Dynamic features extraction: 14 Dynamic features of apks are extracted by executing apk in the Droidbox for 60 seconds, 14 action logs are collected and feature occurrence vector is constructed. Dynamic action behaviour of dataset is depicted in Fig4. Barchart (X-axis represents Dynamic action and Y- Axis represents No of apks performing Actions).

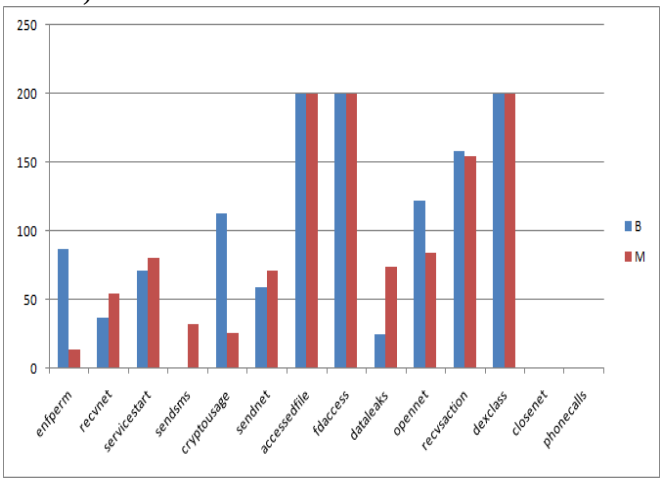


**Fig 4: Occurrence of Dynamic Features in malware and Benign Applications.**

*3.3 Scatter/ Separability Assessment for Feature Selection*

Scatter/Separability Assessment (SA) [13] is a robust attribute identification method, which selects relevant features from independent feature set. SA selects n features of N independent features set (n<<N) by optimizing class separation. Separability Assessment of feature (f) is calculated as the correlation of total variability ($V_{t,f}$) and within class variability ($V_{w,f}$). Between class variance ($V_{b,f}$) and within class variance ($V_{w,f}$) are summed to produce the total variability ($V_{t,f}$). Total variability is calculated using Equation( 1)

$$V_{t,f} = V_{w,f} + V_{b,f} \quad (1)$$

Separability score Ψ fof a feature (f) is computed as in Equation (2)

$$\Psi_f = \frac{Vt,f}{Vw,f} \quad (2)$$

*3.4 Feature Occurrence Table Construction*

Let R is atuple consists a collection of 332 features (143 permissions, 14 Dynamic features, 175 API calls).


Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

For eachi$^{th}$ instance of dataset, we calculate a binary vector R = r1, r2 ...rj... rn and rj = 1, if feature exists and rj=0, otherwise. In addition, class label C attached where C ∈ {Malware, Benign}. This class label C denotes -1 for malware instance and 1 for the benign application.

*3.5 Model generation and Classification*

Four different machine learning algorithms: Naïve Bayes, SVM, J48 and Random Forest from WEKA Tool [3] experimented for classification on feature vector table with and without SA. Experiments were conducted on three different size of dataset. For that we divided benign and malwareapplicationsintheratiosof200:200,200:100and100:200. In the proposed system cross validation with 10 folds is used to measure efficiency of malware scanner. In 10-fold cross validation FVT is divided into 10 partitions and during each iteration one partition is used for testing and rest nine partition is used for training. Accuracy estimate of classification model is the overall number of correct classifications from the k iterations.

*3.6 Evaluation Metrics*

For the evaluation of our dataset, we are using True Positive, False Positive, Precision, Recall and F-Measure [12]. True positive rates denote the ratio of malware that are identified correctly. False positive rates denote the proportion of benign instances wrongly identified as malware. Precision is the ratio of count of correctly classified malware instances to the count of instances classified as malware. Recall is the ratio of true positives count to malware count in dataset. And F-measure is the measure of accuracy and calculated using Equation 3. In evaluation, time is an important constraint to build a model. So here we also analyzed time.

F-measure = 2* Precision * recall / (Precision + recall) (3)

## IV. RESULTS AND DISCUSSION

Classification results without scatter/ separability analysis feature selection is shown in Table1 and with SA feature selection shown in Table2.

**Table 1: Classification Results without Feature selection on 332 Features**

| Samples(B:M) | Algorithms | Time (sec) | TP | FP | Precision | Recall | F-measure |
|---|---|---|---|---|---|---|---|
| 200:200 | Naive Bayes | 0.01 | 0.825 | 0.194 | 0.830 | 0.825 | 0.823 |
| | SVM | 0.21 | 0.800 | 0.214 | 0.801 | 0.800 | 0.798 |
| | J48 | 0.06 | 0.850 | 0.153 | 0.850 | 0.850 | 0.850 |
| | RandomForest | 0.1 | 0.800 | 0.214 | 0.801 | 0.800 | 0.798 |
| 200:100 | Naive Bayes | 0.01 | 0.700 | 0.346 | 0.727 | 0.700 | 0.680 |
| | SVM | 0.13 | 0.775 | 0.265 | 0.807 | 0.775 | 0.763 |
| | J48 | 0.05 | 0.825 | 0.194 | 0.830 | 0.825 | 0.823 |
| | RandomForest | 0.24 | 0.825 | 0.194 | 0.830 | 0.825 | 0.823 |
| 100:200 | Naive Bayes | 0.01 | 0.900 | 0.092 | 0.905 | 0.900 | 0.900 |
| | SVM | 0.1 | 0.900 | 0.092 | 0.905 | 0.905 | 0.900 |
| | J48 | 0.04 | 0.900 | 0.102 | 0.900 | 0.900 | 0.900 |
| | RandomForest | 0.23 | 0.900 | 0.092 | 0.905 | 0.905 | 0.900 |

**Table 2: Classification Results with Scatter/ Separability Feature selection -77 features**

| Samples(B:M) | Algorithms | Time (sec) | TP | FP | Precision | Recall | F-measure |
|---|---|---|---|---|---|---|---|
| 200:200 | Naive Bayes | 0.01 | 0.725 | 0.316 | 0.747 | 0.725 | 0.711 |
| | SVM | 0.10 | 0.975 | 0.020 | 0.976 | 0.975 | 0.975 |
| | J48 | 0.16 | 0.850 | 0.123 | 0.888 | 0.850 | 0.849 |
| | RandomForest | 0.28 | 0.950 | 0.041 | 0.955 | 0.950 | 0.950 |
| 200:100 | Naive Bayes | 0.02 | 0.867 | 0.102 | 0.898 | 0.867 | 0.867 |
| | SVM | 0.02 | 0.967 | 0.025 | 0.969 | 0.967 | 0.967 |
| | J48 | 0.05 | 0.933 | 0.051 | 0.942 | 0.933 | 0.934 |
| | RandomForest | 0.17 | 1 | 0 | 1 | 1 | 1 |
| 100:200 | Naive Bayes | 0.01 | 0.900 | 0.150 | 0.899 | 0.900 | 0.899 |
| | SVM | 0.09 | 0.933 | 0.133 | 0.939 | 0.933 | 0.931 |
| | J48 | 0.05 | 0.933 | 0.133 | 0.939 | 0.933 | 0.931 |
| | RandomForest | 0.12 | 0.900 | 0.200 | 0.913 | 0.900 | 0.895 |

Classification Accuracy tremendously improvedwith feature selection in terms of TP, FP, Recall, precision and F-measure. Random Forest produced best result with one for true positive, zero for false positive and one for F-measure. Time complexity of random forest is high as it is an ensemble classifier, many decision trees models are constructed and majority voting is selected as classification result. The desirable characteristics of random forest are highly accurate, simple easily parallelised, strong against noise and outliers and strong mathematical estimates for error. Naïve Bayes classifier has less time complexity but poor True positive and false positive rates. Regarding the sample size 200:100 ratio produced best results as it close to real ratio of benign and malware.

## V. CONCLUSIONAND FUTURE WORK

The proposed hybrid analysis system is strong enough to distinguish benign and malware applications as it overcomes disadvantages of static and dynamic analysis. Scatter/ Separability analysis used in the system is a strong feature selection which maximizes class separability and selects best 77 features from 332 feature set. Random Forest algorithm performed the best and it provides 100% accuracy in 0.17 seconds and it ensured the quality of ensemble decision tree classifier. The scalability of the proposed system is to include network features which is to be collected from network packets and good feature selection methods which selects features that has higher correlation with class and lesser correlation with features of other class.

## REFERENCES


1. Apktool, https://ibotpeaches.github.io/Apktool/.
2. Virustotal, https://www.virustotal.com, accessed Feb2, 2017
3. WEKA, http://www.cs.waikato.ac.nz/ml/weka/.
4. Google play, https://play.google.com/store.
5. Drebin Dataset. http://user.cs.uni-goettingen.de/~darp/drebin/, Accessed 2 Jan 2017
6. DroidboxTool, https://github.com/pjlantz/droidbox, Accessed Feb 15, 2017
7. Zhenlong Yuan, Yongqiang Lu, and YiboXue., Droid Detector: Android Malware Characterization and Detection Using Deep Learning, ISSN1007-021410/10pp114-123 Volume 21, Number 1, February 2016.
8. Dong-Jie Wu, Ching-Hao Mao, Te-En Wei, Hahn-Ming Lee, Kuo-Ping Wu, DroidMat: Android Malware Detection through Manifest and API Calls Tracing, 2012 Seventh Asia Joint Conference on Information Security, DOI 10.1109/AsiaJCIS.2012.18.
9. MengyuQiao, Andrew H. Sung, and Qingzhong Liu, Merging Permission and API Features for Android Malware Detection, 2016 5th IIAI International Congress on Advanced Applied Informatics, DOI 10.1109/IIAI-AAI.2016.237.
10. Shree Garg, Sateesh K. Peddoju, Anil K. Sarje, Network-based detection of Android malicious apps, Int. J. Inf. Secur. (2017) 16:385–400 DOI 10.1007/s10207-016-0343-z.
11. Ali Feizollah, Nor BadrulAnuar, RosliSalleh, A inuddin Wahid Abdul Wahab. A review on feature selection in mobile malware detection. Digital Investigation 13(2015) 22-37.
12. M. V. Varsha, P. Vinod & K. A. Dhanya, Identification of malicious android app using manifest and opcode features, Journal of Computer Virology and Hacking Techniques, Volume 13.
13. P. Vinod, P. Viswalakshmi. "Empirical Evaluation of a System Call-Based Android Malware Detector", Arabian Journal for Science and Engineering, 2017
14. Narudin, Fairuz Amalina, Ali Feizollah, Nor BadrulAnuar, and Abdullah Gani. "Evaluation of machine learning classifiers for mobile malware detection", Soft Computing, 2016.
15. Varsha M. V., Vinod P. and Dhanya K. A., “Heterogeneous Feature Space for Android Malware Detection," In Proceedings of 8th IEEE International Conference on Contemporary Computing (IC3-2015), 20-22 August, 2015.
16. Kimberl T Tam, AliFeizollah, NorBadrulAnuar, RosliSalleh, LorenzoCavallaro., The Evolution of Android Malware and Android Analysis Techniques., ACM Computing Surveys, Vol. 0, No. 0, Article 00, Publication date: 0
17. Daniel Arp, Hugo Gascon, Konrad Rieck, et al.: DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket, NDSS ’14, 23-26 February 2014, San Diego, CA, USA.
18. Dong-Jie,Wu, Ching-Hao Mao, Te-En Wei, Hahn-Ming Lee, Kuo-Ping WC., Droid Mat: Android Malware Detection through Manifest and API call Tracing., Information Security (ASIA JCIS), 2012.
19. Alessandro Reina, Aristide Fattori, Lorenzo Cavallaro., A System Call-Centric Analysis and Stimulation Technique to Automatically Reconstruct Android Malware Behaviors., EuroSec ’13, April 14 2013, Prague, Czech Republic.
20. Mingshen Sun , John C. S. Lui., Droid Analytics: A Signature Based Analytic System to Collect, Extract, Analyze and Associate Android Malware. 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), 2013.
21. Paul, TG Gregory, and T. Gireesh Kumar. "A Framework for Dynamic Malware Analysis Based on Behavior Artifacts." Proceedings of the 5th International Conference on Frontiers in Intelligent Computing: Theory and Applications. Springer, Singapore, 2017.
22. R.Vinayakumar ; K.P.Soman ; Prabaharan Poornachandran., Deep android malware detection and classification., International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2017.



*Published By:*
*Blue Eyes Intelligence Engineering*
*& Sciences Publication*