Journal of Advanced College of Engineering and Management, Vol. 4, 2018

MALWARE DETECTION AND CLASSIFICATION USING LATENT SEMANTIC INDEXING

Srijana Parajuli[1], Subarna Shakya[2]

[1,2]Department of Electronics and Computer Engineering, Pulchowk Campus, T.U.

email Address: srijana.parajuli7@gmail.com , drss@ioe.edu.np

Abstract

The increasing popularity of smart phones has led to the dramatic growth in mobile malware especially in Android platform. Many aspects of android permission has been studied for malware detection but sufficient attention has not been given to intent. This research work proposes using Latent Semantic Indexing for malware detection and classification with permissions and intents based features. This method analyses the Manifest file of an android application by understanding the risk level of permission and intents and assigning weight score based on their sensitivity. In an experiment conducted using a dataset containing 400 malware samples and 400 normal/benign samples, the results show accuracy of 83.5% using Android Intent against 79.1 % using Android permission. Additionally, experiment on combination of both features results in accuracy of 89.7%. It can be concluded from this research work that dataset with intent based features is able to detect malwares more when compared to permissions based features.

Keywords: *android, intent, permission, weight score, latent semantic indexing, malware detection*

1. Introduction

According to International Data Corporation (IDC), Android OS occupied 85% of the market share of smart phones in 2017 whereas iOS occupied a market share of 14.7% [1]. Android's popularity is a result of being an open source. Due to this open environment, malware authors can develop malicious apps exploiting vulnerabilities in the platform to launch malicious behaviors [2]. Trojans, worms and mobile botnets are the various forms of malware threats on mobile devices. Research studies in the Android malware detection field work in three approaches static, dynamic or hybrid. In static analysis, malware is disassembled into a source code from where specific features are extracted whereas in dynamic analysis malware is monitored at run-time in a virtual environment. The hybrid approach incorporates both static and dynamic analysis. Machine learning algorithms have been used to build classification models by training datasets with application features that are collected from static, dynamic or hybrid analysis. The malware detection framework proposed in this research work is static analysis that uses permissions and intents based features extracted from apk files of an application. The dataset is classified using Latent Semantic Indexing (LSI) to detection whether an android application is normal or a malware.

2. Literature Review

Malware detection method called DREBIN was developed that extracts a broad set of features from the app's AndroidManifest.xml and disassembled codes to generate a feature set [3]. SVM was applied on the dataset to make a classification between the two-categories of apps (benign and malicious). Drebin datasetis currently considered as the largest publicly available dataset with about 5,560 malware samples from 179 different malware families.

A detection method is proposed that combines permission and API calls from classes.dex and manifest file of an android application into a single feature set [4]. The features are combined in a single set with 1 and 0 indicating presence and absence of a feature. Random forest is used to make a distinction between normal and benign applications.

An approach that investigates effects of intent filters on android malware behavior is proposed [5]. Analysis of permission and intent patterns present in manifest file of an android application is used to identify the malicious and benign apps. This approach is also the first of its kind that classifies and android application as benign, malware and benware (showing characteristics in between malware and benign).

Cosine similarity is used to investigate the performance of Arabic language text classification [6]. Singular value decomposition (SVD) method is used to extract textual features based on LSI. The research was conducted on a corpus that contains 4,000 documents of ten topics (400 documents for each topic). This study reveals that the classification methods that use LSI (based on cosine measure) significantly outperform the other methods of classification.

A new approach is proposed that applies Latent Semantic Indexing (LSI) to identify malware application [7]. Documents consisted of a set of malwares whereas terms comprised of set of dangerous permissions. This method examines list of permissions and find a set of highly close and relevant applications from a given set.

3. Methodology

The framework of proposed system is shown in Figure 1. Manifest.xml file of an android application is parsed and intent and permissions are selected as features. Each feature is assigned a weight from 1 to 6 points corresponding to the extent of danger from normal, moderate to very high. Whole dataset is split into three parts: first two parts are used to make term document matrix and query matrices. Remaining third portion of dataset will be used for validation. The term document matrix is reduced into sets of Eigen vectors and singular values using singular value decomposition (SVD) technique. Cosine similarity is used to compute the similarity between the each element in query matrix and term document matrix in the reduced space. If the similarity between query and document is greater than or equal to a threshold value then the query will be labeled as a malware otherwise it will be labeled as benign. The result obtained on the basis of cosine similarity is used to make prediction between malware and benign applications. A joint feature vector is also created by combining permission and intent based features and these combined features were also used to make prediction between malware and benign samples. Finally a performance comparison is made on malware detection module based on each of the intents, permissions and a combination of both features.

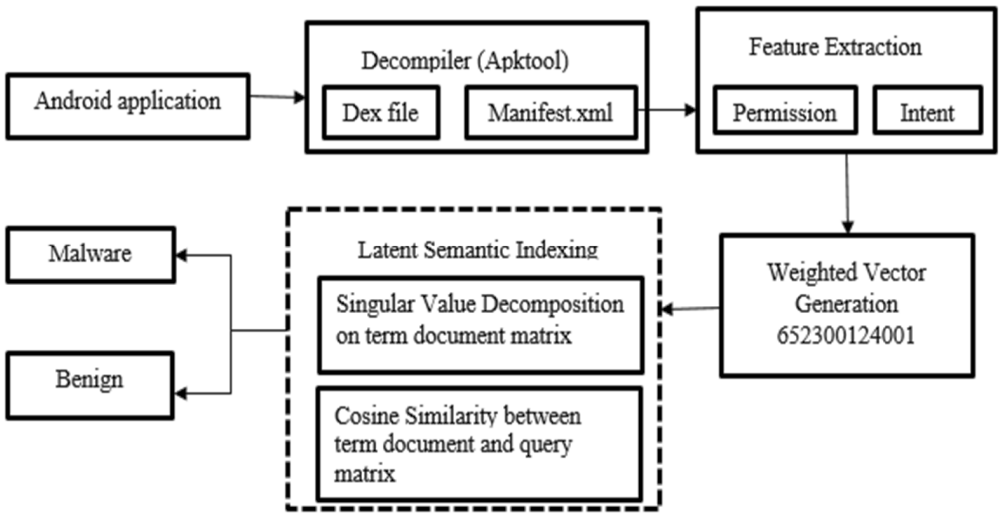

Fig.1: Malware Detection Framework

3.1 Data Collection

About 400 applications belonging to 14 different categories were downloaded from Google Play Store and Andropit to carry out research work. Similarly, 400 malware samples were downloaded from various malware repositories. About 100 malware samples were downloaded from Contagio Dump Store [8]. About 80 malware samples from were taken Kharon Dataset data [9]. About another 120 sets of malware were taken from GitHub repositories [10]. For validation of proposed module 100 malware samples were downloaded from Drebin Dataset [11]. The APK files of both 400 benign and 400 malware samples from each of the stores have been analyzed using free online malware detection tool called VirusTotal. Based on the results, applications are labeled as malware or not.

3.2 Application Feature Extraction

The malware detection approach utilizes the static features of an android application. The most significant permissions and intents that lead to efficient discrimination between the malware and benign applications are selected. The APK files of applications from data collection are decompiled into source code in forms of AndroidManiFest.xml and java classes (.dex files). The android Manifest.xml file is mined to obtain permissions and intent based features. About 48 permissions were taken to create dataset based on permission based features and 52 intents were taken to create dataset on intent based features. The weight assignment to various features range from 1 to 6 corresponding to the extent of danger or sensitivity from normal to very high. If the feature is not present it is represented by 0. Apktool is the tool used to extract permissions and intents based features from APK files [12].

3.3 Android Permission and Intent Risk Levels and Weight Score

Scores are assigned to features extracted in Feature extraction section. For permissions and intents based features, scores are assigned based on their risk level. .The permissions weight is scored on a scale from 1 to 6 points corresponding to the extent of danger from normal, moderate to very high. Weights are assigned based on the paper [13].

Table 1: Some of the Android Permissions and corresponding weights based on risk level

Permissions	Risk Level	Weight Score if Present
WRITE_APN_SETTINGS	VERY-HIGH	6
READ_CONTACTS	MEDIUM-HIGH	4
WRITE_CONTACTS	HIGH	5
GET_ACCOUNTS	MEDIUM-HIGH	4
ACCESS_COARSE_LOCATION	MODERATE-HIGH	3
CHANGE_WIFI_STATE	MODERATE	2
CALL_PHONE	MEDIUM-HIGH	4
PROCESS_OUTGOING_CALLS	VERY-HIGH	6
SEND_SMS	HIGH	5
READ_EXTERNAL_STORAGE	HIGH	5
WRITE_EXTERNAL_STORAGE	VERY-HIGH	6
WRTIE_HISTORY_BOOKMARKS	MODERATE-HIGH	3
RECEIVE_BOOT_COMPLETED	NORMAL	1
INTERNET	NORMAL	1

Similarly, for weights for intents was assigned on a scale from 1 to 6 points corresponding to its risk level. The convention followed to assign weights to intents is given below:

i. For intents that are related to permissions, same weight was assigned as that of corresponding permission as shown in Table 4.4. For example, intents like CALL require CALL_PHONE permission so CALL intent is assigned weight of 4 as like its permission counterpart.
ii. Most frequently occurring intents in the malwares are considered to be more sensitive and assigned high weights. Most malwares used BOOT_COMPLETED, SMS_RECEIVED,INSTALL_SHORTCUT,NEW_OUTGOING_CALL,TEXT,UNINSTALL_SHORTCUT, DIAL, PHONE_STATE, USER_PRESENT, intents. Hence these types of intents were assigned high weight scores [19].
iii. Most frequently occurring intents in benign applications are considered normal and assigned low weight value of 1. Intents like ACTION_SCAN, ACTION_REFRESH, BATTERY_LOW,DATE_CHANGED, HOME were present in benign samples only so these intents were assumed to be of normal risk level and assigned weight of 1.
iv. Most benign apps used CATEGORY_LAUNCEHER, VIEW, CATEGORY_DEFAULT, CATEGORY_BROWSABLE, ACTION_MAIN intents and so these intents were assigned weight of 1.
v. Remaining intents whose risks levels could not be identified were assumed to be of normal and assigned weight of 1.

Table 2: Some of the Android Intents and corresponding weights based on risk level

Intents	Risk Level	Weight Score if Present
BOOT_COMPLETED	VERY-HIGH	6
SMS_RECEIVED	HIGH	5
CONNECTIVITY_CHANGE	HIGH	5
USER_PRESENT	MODERATE	2
PACKAGE_INSTALL	VERY-HIGH	6
SCREEN_ON	MODERATE	2
CALL	MEDIUM-HIGH	4
SIG_STR	MEDIUM-HIGH	4
PACKAGE_REMOVED	MODERATE	2
ACTION_VIEW	NORMAL	1
CATEGORY_BROWSABLE	NORMAL	1
ACTION_MAIN	NORMAL	1
CATEGORY_LAUNCHER	NORMAL	1
CATEGORY_DEFAULT	NORMAL	1

3.4 Joint Features Generation by Combining Intents and Permissions

After extracting static features such as permissions and intents and assigning appropriate weights, a joint feature vector is generated by combining features. If I is the set of intents and P is the set of permissions in an android application, then a joint vector S is generated by combining P and I present

in the application .i.e. S=P+I . Now the total number of features is 100(48 permissions and 52 intents) where S_j represents a joint feature vector (S_1,S_2,S_3,…..,S_{100}) where Sj=0 or (1,2,3…,6) on the basis of presence and absence of a feature .

3.5 Malware Classification using Latent Semantic Indexing

Latent Semantic Indexing(LSI) is an information retrieval technique where a set of words are used to identify the most relevant set of documents. Queries against a set of documents that have undergone LSI will return results that are conceptually similar to search criteria. A matrix is computed in which rows correspond to documents(android applications) and columns correspond to terms(features). This matrix is then reduced using singular value decomposition (SVD) technique to find the most important set of documents. After obtaining the low-rank approximation of the term-document matrix using SVD, the computed matrices are used to project each vector in query matrix in the reduced space. The android applications are split and are represented as documents vectors and query vectors. Similarity between document vector and query vector is calculated by measuring the cosine of the angle between them. This measure of similarity is called cosine similarity. Higher value of cosine of angle means the two vectors are much closer to each other and more similar.

To make a classification using latent semantic indexing by measuring cosine similarity, the dataset is split into three parts: a set of 200 application samples are used to make term document matrix and 400 samples are used to make query matrix. Remaining 200 applications are used for validation. Each term (feature) is assigned either 0 or some weight score and a document (application) is characterized by a vector where the value of each dimension corresponds to presence or absence of the term in the document (application).

The algorithm for classification based on latent semantic indexing is represented in Figure 2.

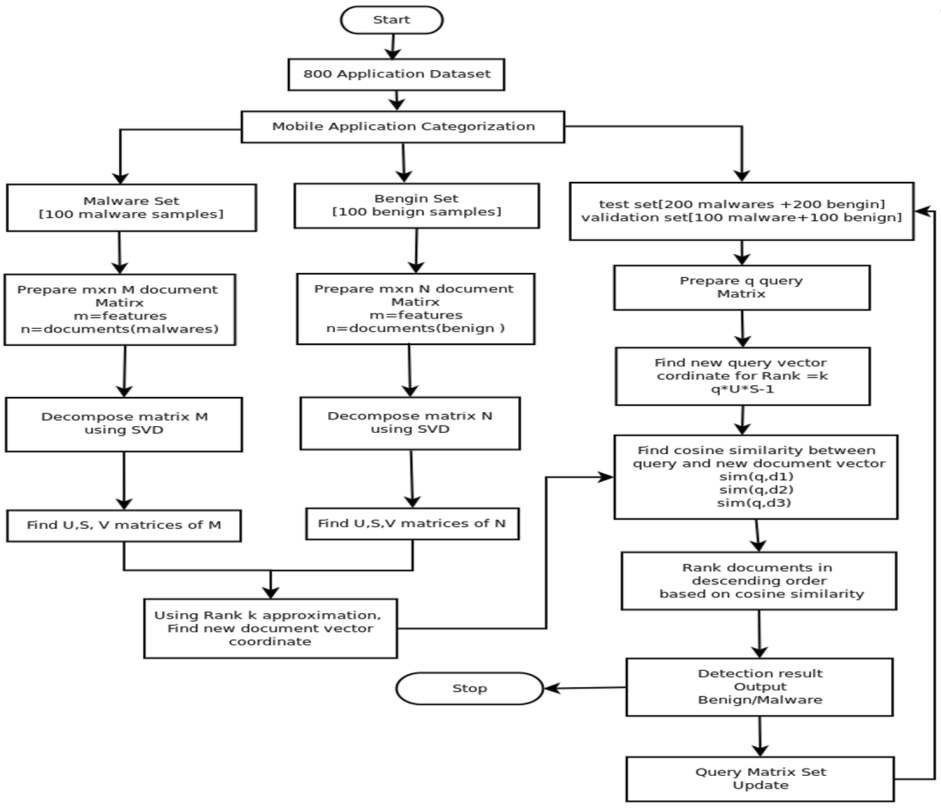

Fig. 2: LSI implementation in Malware Detection and Classification

4. Results and Discussion

The proposed algorithm was tested by applying the receiver operating characteristics (ROC) on classification output of Latent Semantic Indexing on the data. A ROC curve is a commonly used graph that summarizes the performance of a classifier over all possible thresholds.

4.1 Parameters Selection for malware classification using LSI

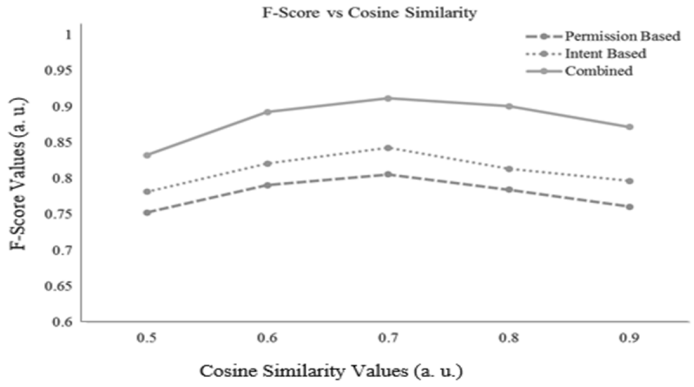

Fig. 3: Variation of F-Score with change in cosine similarity threshold

The variation of F-Score with change in cosine similarity threshold values for three types of datasets is shown in Figure 3. It is seen that when features are used separately cosine similarity of 0.7 generates highest F-score whereas when these features are combined, cosine similarity threshold of 0.8 generates highest F-Score value. Since malware detection is a classification problem, cosine similarity is used to make prediction between malware and benign samples. There exists a tradeoff between precision and recall while choosing appropriate cosine similarity threshold. High precision relates to low false positive rate and high recall relates to low false negative rate. If threshold is high, precision will be high and recall will be low and vice versa. As some sort of balance between precision and recall was needed to choose cosine similarity threshold, F score was calculated and cosine similarity threshold value was chosen with highest F Score.

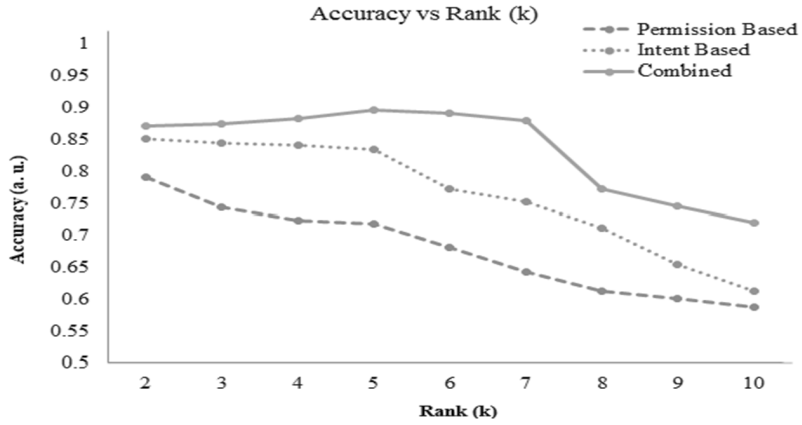

Fig. 4: Variation of Accuracy for different values of Rank (k)

The accuracy of three types of datasets for different values of Rank (k) is shown in Figure 5.4. It can be observed that prediction accuracy changes when k changes. This is because k directly represents the dimension of co-ordinates of new document vectors. The range is chosen from 1 to 10. Accuracy

is highest when rank 2 is used for dataset where intent and permission based features and rank 5 is taken for combination of features.

Table 3: Parameters chosen for classification using LSI

	Permission based Features	**Intent based Features**	**Combined Features**
Size of Term Document Matrix	48×200	52×200	100×200
Size of Query Matrix (test set)	48×400	52×400	100×400
Size of Query Matrix (validation set)	48×200	52×200	100×200
Rank Approximation (k)	2	2	5
Cosine similarity threshold (Sim)	0.7	0.7	0.8

Table 4: Class Prediction for Malware Detection based on Cosine Similarity (sim)

Predicted Class	**Permission Feature**	**Intent Feature**	**Combined Feature**
0	If sim < 0.7	If sim< 0.7	If sim < 0.8
1	If sim >= 0.7	If sim >= 0.7	If sim >= 0.8

4.2 Performance Metrics Calculation

The evaluation of a classifier is most often based on its predictive accuracy. In this study, different performance metrics are calculated to evaluate the performance of malware classification module using LSI. True positive rate (TPR) also known as "Recall" is the ratio of correctly classified malicious apps to the total number of malicious apps in the dataset.False positive rate (FPR) is the ratio of incorrectly classified benign apps to the total number of benign apps in the dataset. Accuracy is the percentage of test instances that are correctly classified by the learning algorithm. Precision is the ratio of true positives to the total positives whereas F-Score is harmonic mean between precision and recall. The calculation of Accuracy, Precision, Recall and F-Score for LSI using android permission, intents and combined features for test set and validation set is shown in Table 5.

Table 5: Calculation of Performance Metrics

Features	**TPR (Recall)**	**FPR**	**Accuracy**	**Precision**	**F-Score**
Permission(test)	0.779	0.187	0.791	0.854	0.806
Permission(validation)	0.741	0.171	0.768	0.897	0.7931
Intent (test)	0.821	0.142	0.835	0.891	0.843
Intent (validation)	0.805	0.165	0.817	0.885	0.827
Combined (test)	0.875	0.101	0.897	0.952	0.912
Combined (validation)	0.869	0.112	0.884	0.941	0.903

4.3 Performance Evaluation Using Receiver Operating Characteristic (ROC) Curve

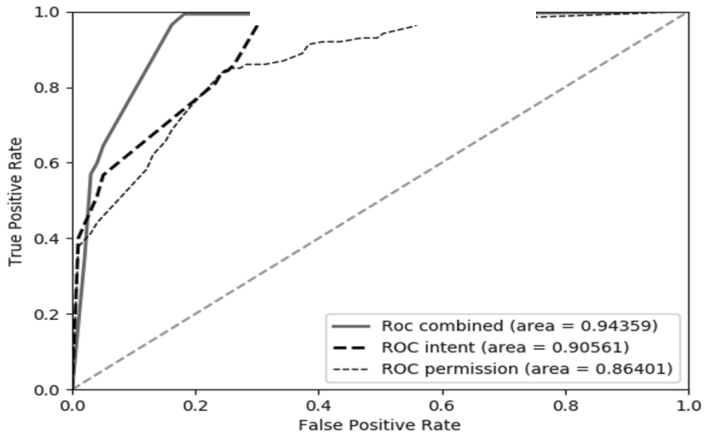

Fig. 5: ROC Curve for classification based on permission, intent and combined features

The roc plot for LSI for three types of feature set is shown in Figure 5. For permission based classification, the mean area under curve for LSI 0.864 and for intent based classification the mean area under curve for LSI was found to be 0.9056. As higher the value of area under curve, better the classifier. Both the intents and permission detect android malwares with sufficient accuracy. However, it can also be seen that LSI performs better when intents are used as features than permissions. All the permissions declared in manifest file may not be used by the applications whereas intent reflects the actual intentions of applications resulting directly from activities. This indicates that Intent is more effective for malware detection. Similarly, after combining intent and permission based features and training LSI module, the mean area under curve was found to be 0.943 which is highest among these three. This is due to the fact that increase in types of features increases classification accuracy as there will be two kinds of features describing an application. The module also works well for validation data set as seen in Table 5. The results obtained from validation affirm that the proposed approach can identify malicious mobile applications with sufficient accuracy.

5. Conclusion

In this research work, android permissions and intents are explored for malware detection using Latent Semantic Indexing. The advantage of this method is that it uses only manifest files to detect malware. Manifest files are required in all Android applications, and thus, the proposed method is applicable to all Android applications. Moreover, the cost of analyzing only the manifest file is extremely low. The results show that the use of Android Intent in our approach achieves higher accuracy than permissions. The results are also validated by testing the model with malware samples taken from a well-known and verified data source. In conclusion, it can be use of intents increased true positive rates and decreased false positive rates when compared with permissions only. The research also combines both permissions and intents and make malware detection approach more precise.

6. Suggestions and Recommendations

There are many possibilities to exploit and extend the approach used in this research .The proposed module uses only a small number of samples; only 800 samples in total. In future, additional samples can be collected to obtain more precise results and code based features like API Calls can also be incorporated to make data. A standard algorithm can be used to calculate the value of k rank and

cosine similarity instead of the empirical approach as done in this study. A hybrid approach that combines both static and dynamic analysis to tackle mobile malware can be done. In addition, a graphical user interface can be developed to show list of applications that are considered malware or normal.

References

1. *"Android OS Statistics"*, http://www.idc.com/promo/smartphone-market-share/os (Accessed on 18th Feb, 2018)
2. Sahs J., Khan L. *"A Machine Learning Approach to Android Malware Detection"*, European Intelligence and Security Informatics Conference, 2012
3. Arp D., Spreitzenbarth M., Hubner M., Gascon H., Rieck K. and Siemens C., *"DREBIN: Effective and explainable detection of android malware in your pocket."* In Procedings of Annual Symposium on Network and Distributed System Security (NDSS) Symposium, San Diego, 2014
4. Yang M., Wen Q. Y. ,*" Detecting Android malware with intensive feature engineering"*, 7th IEEE International Conference on Software Engineering and Service Science (ICSESS),2016
5. Idrees F. , Rajarajan M., *"Investigating the Android Intents and permissions for Malware Detection*",10th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications,2014
6. Al-Anzi F. S. and AbuZeina D., *"Toward an enhanced Arabic text classification using cosine similarity and Latent Semantic Indexing"*, Journal of King Saud University – Computer and Information Sciences,2017
7. Shahriar H., Islam M. and Clincy V. (2017) *"Android Malware Detection Using Permission Analysis "*, 8th IEEE International Conference,2017
8. *"Contagio Dump Source"*, https://www.dropbox.com/sh/ i6ed6v32x0fp94z/ AAA QvOsOvbWrOs8T3ZTXqQya?dl. (Accessed on 5th April 2017)
9. *"Kharon Dataset"*, http://kharon.gforge.inria.fr/dataset/index.html. (Accessed on 5th April 2017)
10. *"Malware Source"*, https:// github.com/ashishb/android-malware. (Accessed on 5th April 2017)
11. *"Drebin Data Source"*, https://www.sec.cs.tu-bs.de/~danarp/drebin/download.html. (Accessed on 20th April 2018)
12. *"Reverse engineering Android APK files"*, https://ibotpeaches.github.io/Apktool/. (Accessed on 20th April 2018)
13. Duc N. V., Giang P. T., Vi P. M., *"Permission Analysis for Android Malware Detection",* The proceedings of the 7th vast aist workshop research collaboration: review and perspective, 2016