Using Weighted Bipartite Graph for Android Malware Classification

Altyeb Altaher
Faculty of Computing and Information Technology in Rabigh
King Abdulaziz University
Jeddah, Saudi Arabia

***Abstract*—The complexity and the number of mobile malware are increasing continually as the usage of smartphones continue to rise. The popularity of Android has increased the number of malware that target Android-based smartphones. Developing efficient and effective approaches for Android malware classification is emerging as a new challenge. This paper introduces an effective Android malware classifier based on the weighted bipartite graph. This classifier includes two phases: in the first phase, the permissions and API Calls used in the Android app are utilized to construct the weighted bipartite graph; the feature importance scores are integrated as weights in the bipartite graph to improve the discrimination between malware and goodware apps, by incorporating extra meaningful information into the graph structure. The second phase applied multiple classifiers to categorise the Android application as a malware or goodware. The results using an Android malware dataset consists of different malware families, showing the effectiveness of our approach toward Android malware classification.**

Keywords—Android malware; Bipartite graph; Classification algorithms; machine learning

I. INTRODUCTION

Smartphones have become increasingly essential part of our daily lives, leading to an exponential growth in the number of smartphone users. Android is a extensively used operating system for smartphones and represents more than 84 % of the smartphone market in the first quarter of 2016 [1]. The new advanced capabilities of the smartphones, coupled with the popularity of Android OS, have attracted many developers to offer useful applications— commonly called apps. The basic market for Android apps is Google Play, also several third-party stores are availabe. The number of hosted apps on Google Play was around two million apps in February 2016 [2].

The increasing number of both smartphone users and available apps has attracted malware developers to design malware apps for smartphones. The amount of new Android malware apps in 2015 was 884,774; this number has increased to more than three times compared to 2014 [3]. As more new sophisticated Android malware apps evolve, their detection using traditional signature based approaches become more challenging.

Android presented the system of permissions as a potential approach to restrict access to user resources. Android apps request user approval for permissions to access smartphone resources. Thus, the Android permissions have been introduced to protect users from malware apps.

There are many permission-based approaches for Android malware detection [4,5,6,7]. However, the existence of some permission is not sufficient evidence to classify the App as malware, as most of the permissions requested by goodware apps are also requested by malware apps. Moreover, the permissions stated in the Android-Manifest.xml are not necessarily employed by the App [8,9]. Several researches considered the API call used in the apps' code to differentiate between malware and goodware apps [10,11,12,13,14]. However, these methods need many API calls for malware classification.

This paper proposes an effective approach for Android app classification as malware or goodware using weighted bipartite graph mining. The contributions of this research are as follows:

- This paper, shown improved results compared to several other approaches for the problem of classifying Android malware apps. The carefully crafted weighted bipartite graph structure based on Android permissions and API calls, combined with the support vector machine classifier, achieves better performance and discriminates between the goodware and malware apps efficiently with low false positive rates. This paper aims to utilize both Android permissions and API calls in the building of the weighted bipartite graph for Android malware classification.
- This paper introduced the use of the important score of Android permissions and API calls as weights in the edges of the constructed bipartite graph. This approach improves discrimination between malware and goodware apps by signifying the association level between an Android app and its used features.

This paper is structured as follows: the related work is presented in section 2. The details of the approach for Android malware classification based on the weighted bipartite graph mining have been discussed in section 3. In Section 4, the used data set and results has been discussed. In Section 5, the conclusions of our study have been presented.

II. RELATED WORK

Dynamic and static approaches are the two main approaches for Android malware analysis [15]. In the dynamic approach, the required features are monitored and collected during the running of Android app in a mobile device or emulator. Examples for the features extracted dynamically include SMS and call information [16], and logs of the system [17]. In the static approach, the features of Android app are extracted from the decompiled Android Dalvik bytecode. API calls and permissions features could be extracted statically .Felt et al. [4] proposed a tool that provides a list of expected permissions for the application, and then it compares them with the really requested permissions. In [18], authors proposed Kirin which is a security service to protect the smartphone. The dangerous permissions requested by the app are formulated as rules in Kirin. During the installation of the app, if the requested permissions do not satisfy the security rules of Kirin, the installation of the app is disallowed. Sanz et al. [19,20] introduced an approach for Android malware categorization based on the user features and the permissions available in the AndroidManifest.xml. In the proposed approach, they employed several machine learning algorithms using a dataset contains 357 goodware and 249 malware apps; the achieved classification accuracy was 86%. [21] used the matches of the requested permissions and the behavioural characteristics, such as the method of installation, to detect Android malware.

Analysing the system calls for Android malware detection is another research area. Schmidt et al. [22] obtained the system calls from the Android Apk file and matched them with the system calls used in malware apps to classify apps as either malware or goodware apps. Crowdroid [17] is a client server model for Android malware detection; the clients collect the system calls from the Android app and send them to the server for classification using clustering algorithms. In DroidAPIminer [23], different classifiers used to discriminate between goodware and malware apps based on feature sets consisting of API calls.

Andromaly [24] used anomaly detection techniques to classify the malware apps by monitoring different system Metrics. Bartel et al. [25] found that several Android apps declare permissions in their AndroidMainfest.xml file but those permissions are never used. Thus, exploring the permissions only in the manifest file may not provide accurate classification of malware apps. All the API calls used in an Android app are associated with corresponding permissions; therefore when an API call starts, the Android OS verifies the approval of its associated permission before executing it. Good Android malware classification results can be achieved by using features combination approach. [24] Showed that higher accuracy of malware detection can be attained when combing the permission and API Calls. Grace et al. [9] proposed a method using data and control-flow as static features that give 9% False Negative (FN).

Our work differs from the aforementioned works in that, this paper presents an effective classifier for Android malware using the weighted bipartite graph; the permissions and API Calls used in the Android app are utilized to construct the weighted bipartite graph, and to understand the benefits of incorporating additional meaningful information into the graph structure. Moreover, we employed a classification method based on efficient classifiers.

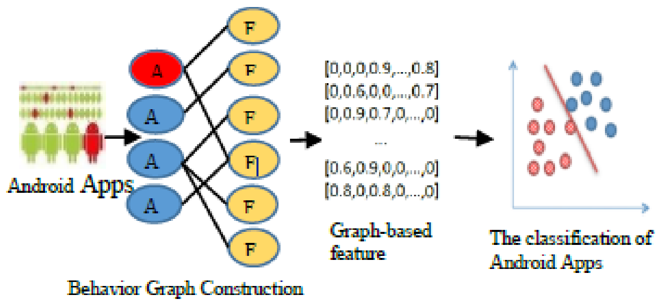


Fig. 1. The proposed approach for Android malware classification

III. Android Malware Classification based on Weighted Bipartite Graph

In this section, the proposed approach for Android malware classification is discussed. As shown in Figure 1, the proposed approach includes the following steps:

A. *Behavior Graph Construction*

First, the features extracted from the Android app to differentiate between goodware and malware apps were presented; then the method how these features are used to construct the weighted bipartite graph is described.

1) *Android app Features*

Using the most informative features, which represent the basic characteristics of Android apps, has significant impact on the classification accuracy. In this research, the Android API calls and permissions are obtained from the Android app files using apktool [26] and utilized as features for the categorization of Android apps as malware or goodware apps.

a) Android Permissions

Android introduced Permissions as a potential mechanism for security. Basically, no Application is permitted to perform any activity that affects the users. The applications can share data and resources by explicitly declaring the permissions in the application file. Therefore, the permissions based features are important features for analysing the Android apps [27]. Before the Android app is installed, Android shows the user the permissions needed by the app; Figure 2 shows examples for permissions stated in the Manifest.xml file of the bgserv malware application.

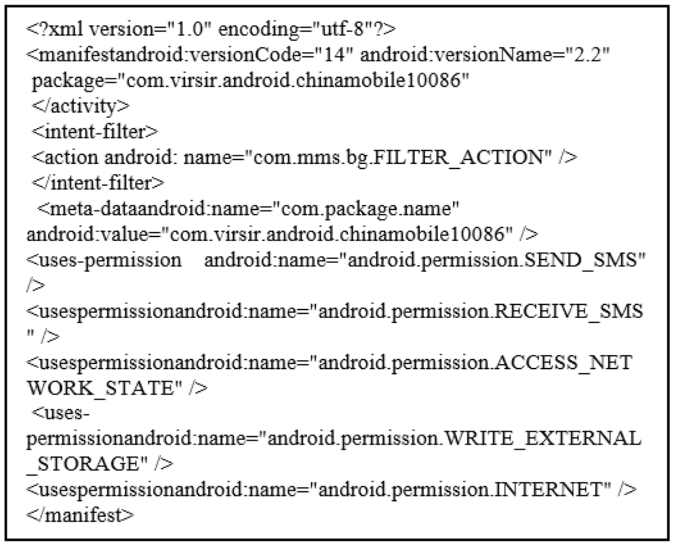
b) API Calls

It is a collection of defined functions and methods that enables the apps to communicate with each other and with the Android OS. For efferent classification of Android malware app, it is most important to focus on the APIs that frequently used by the malware, instead of considering all API calls in the app. Seo et al. [28] explored malware apps and specified the suspicious API calls that malware apps frequently use. They compared the frequency of using suspicious API calls in malware and goodware apps. The API calls extracted manually, which are similar to the suspicious API calls defined in [28], from the Android app. These API calls perform tasks such as gathering the information of the user or device, websites' accessibility, receiving and sending SMS, and app installation.

2) *Bipartite Graph Construction*

The weighted bipartite graph G ={V,U, E} contains a set of vertices V ={v1, v2, ..., vn}, a set of vertices U={u1,u2,…,un} and a set of edges E={eij | edge between vi and uj, i, j <= n}, where n denotes the vertices number. The edge E is linked with a weight w: E→R, where R represents the set of real NUMBERS. The vertices are elements of interest and relationships between the elements are represented by the edges. In this research, the V set of vertices in the constructed bipartite graph represents the Android apps and the other set of vertices U represents the permissions and API Calls used in the Android app. Because the weights can reflect potential systems properties [29], we integrated the features importance scores as edge weights to improve the discrimination between the malware and goodware apps, by signifying the association level between the Android app and the used features.

Information Gain Ratio algorithm finds the similarity score between the Android permissions and API calls; this algorithm then provides the highest weight value to the significant permissions and API calls considering the class of goodware and malware Android applications [30].



```
<?xml version="1.0" encoding="utf-8"?>
<manifestandroid:versionCode="14" android:versionName="2.2"
 package="com.virsir.android.chinamobile10086"
 </activity>
 <intent-filter>
 <action android: name="com.mms.bg.FILTER_ACTION" />
 </intent-filter>
  <meta-dataandroid:name="com.package.name"
android:value="com.virsir.android.chinamobile10086" />
<uses-permission    android:name="android.permission.SEND_SMS"
/>
<usespermissionandroid:name="android.permission.RECEIVE_SMS
" />
<usespermissionandroid:name="android.permission.ACCESS_NET
WORK_STATE" />
 <uses-
permissionandroid:name="android.permission.WRITE_EXTERNAL
_STORAGE" />
<usespermissionandroid:name="android.permission.INTERNET" />
</manifest>
```

Fig. 2. Example for AndroidManifest.xml file

The information gain ratio is calculated using the following equations:

$$gain\_r(eij, C) = \frac{gain\_r(eij,C)}{split\_info(C)} \quad (1)$$

$$split\_info(C)=\sum_i \left(\frac{|C_i|}{|C|}\right) log \frac{|C_i|}{|C|} \quad (2)$$

Where, gain r (eij,C) indicates the gain ratio of the feature eij regularity in the class C. Ci and |Ci| indicate the regularity of feature Z in class C, the subclass *i* of C and the total number of features in Ci. Let W(eij) be the weight matrix of the bipartite graph G, for each edge eij the weight W (eij) can be computed using the information Gain algorithm as follows:

$$w(e_{ij}) = \begin{cases} gain\_r(e_{ij}, C), if\ e_{ij}\ exists \\ 0, \quad otherwise \end{cases} \quad (3)$$

B. *The algorithm used for graph-based vector classification*

A support vector machine is an efficient machine learning technique for classification [31]. It finds a decision boundary with the closest training patterns and classify new patterns based on the decision boundary. The SVM algorithm has a number of advantages, which are significantly important for the classification of Android apps. For example, it has the ability to handle large feature spaces and robust to overfitting spaces. In this paper, four SVM kernel functions: polynomial, linear, radial basis functions and sigmpoid, as well as the state-of-the-art algorithm for learning linear SVM Stegasos [32] were used for graph-based vector classification.

IV. EXPERIMENTAL RESULTS

The experimental results and performance evaluation of the proposed Android malware classification based on weighted bipartite graph mining are presented in this section.

A. *Data Set*

To assess the proposed classifier’s performance, we conducted experiments based on dataset contains 250 malware apps and 250 goodware apps. The goodware apps have been sourced from the well-known market for Android applications, Google play; the malware apps from the Android Malware Genome Project [33].

B. *Performance metrics*

We used the following metrics for evaluating the proposed approach for Android malware classification:

True Positive Ratio (TPR): the ratio of Android malware apps that were classified correctly as malware apps.

$$TPR = \frac{TP}{TP+FN} \quad (4)$$

Where TP represents the malware apps which classified correctly and FN represents the malware apps which incorrectly classified as goodware apps.

True Negative Ratio (TNR): is the ratio of goodware apps that were classified correctly as goodware apps.

$$TNR = \frac{TN}{TN+FP} \quad (5)$$

where TN represents the goodware apps which correctly classified as goodware and FP represents the goodware apps which incorrectly classified as malware apps.

Accuracy: the ratio of malware apps which classified correctly as malware apps:

$$\text{Precision} = \frac{TP}{TP+FN} \quad (6)$$

TABLE. I. SAMPLE FROM THE WEIGHT MATRIX OF ANDROID APP BIPARTITE GRAPH

	Read_Phone_State	System_Alert_Window	Receive_Boot_Completed	Internet	Write_External_Storage
App1	0.2746	0.2529	0.2366	0.0776	0.0414
App2	0	0	0	0	0
App3	0.2746	0.2529	0.2366	0.0776	0.0414
App4	0.2746	0	0.2366	0.0776	0.0414
App5	0.2746	0	0.2366	0.0776	0
App6	0.2746	0	0.2366	0.0776	0.0414
App7	0.2746	0.2529	0.2366	0.0776	0.0414
App8	0.2746	0	0.2366	0.0776	0
App9	0.2746	0	0.2366	0.0776	0
App10	0.2746	0.2529	0.2366	0.0776	0
App11	0.2746	0.2529	0.2366	0.0776	0
App12	0.2746	0	0.2366	0.0776	0.0414
App13	0.2746	0	0.2366	0.0776	0.0414
App14	0.2746	0.2529	0.2366	0.0776	0.0414
App15	0.2746	0.2529	0.2366	0.0776	0.0414
App16	0.2746	0.2529	0.2366	0.0776	0.0414
App17	0.2746	0	0.2366	0.0776	0.0414
App18	0.2746	0.2529	0.2366	0.0776	0.0414
App19	0.2746	0.2529	0.2366	0.0776	0.0414
App20	0.2746	0.2529	0.2366	0.0776	0.0414

C. *Results and Discussion*

The dataset described in section 5.2 is utilized to investigate the performance of the introduced classifier for Android malware based on the 10-fold cross validation. The dataset of goodware and malware apps is randomly divided into 10 groups. Each time, we choose one group consisting of goodware and malware apps as dataset for testing, and the rest 9 groups are used as dataset for training.

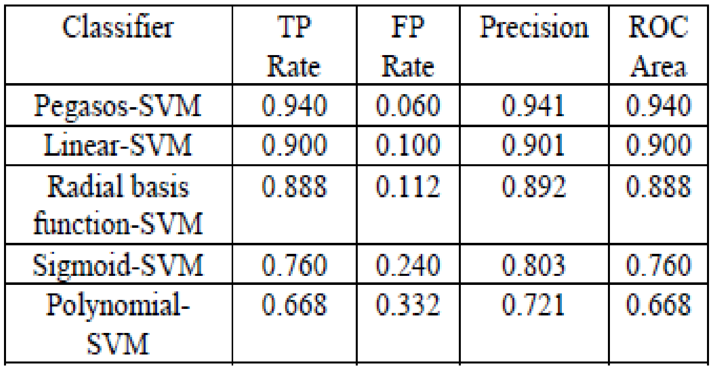
The Android permissions and API calls were extracted from each Android app in the dataset and used to construct the bipartite graph. The important scores of the Android permissions and API calls for discrimination between malware and goodware apps were computed based on the Information gain method and used as weights in the constructed bipartite graph.

From the weighed bipartite graph of Android app depicted in Table 1, we can gain additional insights into the behaviour of Android malware. READ_PHONE_STATE permission is the most discriminative feature between the malware and goodware apps; this permission is essential to get the phone identification information such as device ID. This permission is also required by malware apps that attempt to achieve a financial gain by sending the phone number to a charged service. The second most discriminative feature is RECEIVE_BOOT_COMPLETED permission. This feature is used by a malware app to execute background services without the user's interference. SYSTEM_ALERT_WINDOW permission permits the app to display a window; malware developers could use the SYSTEM_ALERT_WINDOW permission to popup a window to evade the user and steal sensitive information.

The applications that are part of the Android operating system only need the SYSTEM_ALERT_WINDOW permission; an example for the alert window is the window shown when the smartphone is out of battery.

In the experiment, different SVM variants were adopted to categorize the Android apps as malware or goodware apps. The performance of our classifier is evaluated by computing the precision, true positive rate and false positive rate as explained in Table 2.

TABLE. II. PERFORMANCE EVALUATION OF DIFFERENT SVM CLASSIFIERS



Classifier	TP Rate	FP Rate	Precision	ROC Area
Pegasos-SVM	0.940	0.060	0.941	0.940
Linear-SVM	0.900	0.100	0.901	0.900
Radial basis function-SVM	0.888	0.112	0.892	0.888
Sigmoid-SVM	0.760	0.240	0.803	0.760
Polynomial-SVM	0.668	0.332	0.721	0.668

Table 2 shows the performance of variant types of SVM kernel functions. It is clear from Table 2 that the Pegasos-SVM classifier achieves the highest precision of 0.940 % with a minimum false positive rate of 0.062. Pegasos-SVM outperforms all other SVM kernel functions due to its significantly better convergence bounds. The polynomial SVM kernel function achieved the lowest precision of 0.803 with the highest false positive rate of 0.240.

The Receiver Operating Characteristics (ROC) is also utilized to compare between the different approaches. ROC compares the performance of different classifiers using the false positive rates and true positive rates. In the ROC graph, the true positive rate is displayed on the Y axis and the false positive rate is displayed on the X axis. To evaluate the classifiers performance, the area under the ROC curve (AUC) [39] is used. The AUC value is in the range [0.5, 1]. The accuracy of the classifier is 100% when AUC = 1.

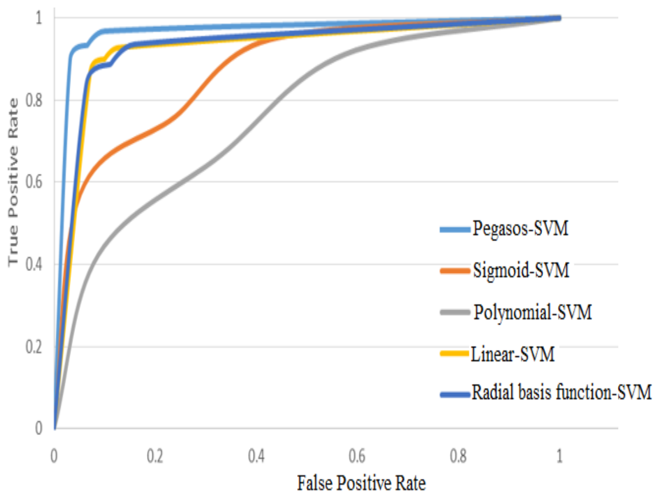


Fig. 3. The RUC curves for the Pegasos-SVM, Linear -SVM, Radial basis function -SVM, Sigmoid –SVM and Polynomial -SVM classifiers

Figure 3 shows the RUC curves for the Pegasos-SVM, Linear -SVM, Radial basis function -SVM, Sigmoid –SVM and Polynomial -SVM classifers. The AUC results confirm that the Pegasos-SVM classifier's area under the curve (AUC=0.940) is greater than all the other classifiers, namely, Linear -SVM (AUC=0.900), Radial basis function-SVM(AUC=0.888), Sigmoid–SVM(AUC=0.760) and Polynomial -SVM(AUC=0.668).

Figure 4 shows the performance comparison between the proposed approach based on Pegasos-VM and the Naïve bayes, Neural network and Decision table respectively. It is clear from Figure 4 that the proposed approach outperforms the classification algorithms: Naïve Bayes, Neural network and Decision table. The proposed approach based on Pegasos-VM achieved the highest precision of 0.941 with a minimum false positive rate of 0.06.

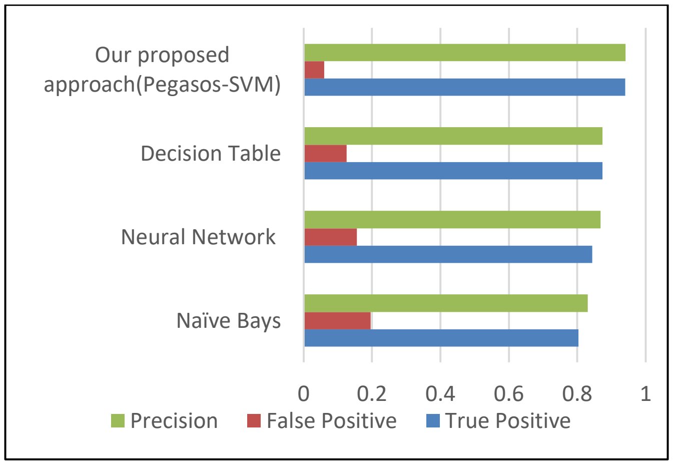


Fig. 4. Performance comparison between the proposed approach based on Pegasos-VM and Naïve bayes, Neural network and Decision table classification algorithms

The proposed classifier's performance is compared with other approaches as shown in Table 3. The results clearly show that the proposed approach achieved an improved accuracy level of Android malware classification with a minimum false positive rate of 6% compared with other approaches. The proposed approach has the capability to integrate the importance scores of Android permissions and API calls as weights in the bipartite graph to improve discrimination between the malware and goodware apps, so that it achieves the minimum false positive rate.

TABLE. III. PERFORMANCE EVALUATION OF THE PROPOSED CLASSIFIER AND OTHER APPROACHES

Classifier	Classification accuracy
K-ANFIS [34]	FP = 10%.
Apposcopy [14]	FN= 10 %
DroidMOSS [35]	FP = 7.1% to 13.3%.
DREBIN [36]	FP = 6%.
Crowdroid [17]	FP = 20%.
RiskRanker [37]	FN = 9%.
AndroSimilar [38]	FP = 7%
Andromaly [24]	FP = 10%
The proposed classifier	FP = 6 %

V. CONCLUSION

Developing efficient and effective approaches for Android malware classification is emerging as a new challenge. This paper, introduces a classifier for Android malware based on the analysis of a weighted bipartite graph constructed from the Android API calls and permissions. The importance scores of the Android API calls and permissions are integrated as weights in the bipartite graph to improve the discrimination between the malware and goodware apps. The graph-based feature vector is constructed from the weighted bipartite graph sent to a support vector machine to categorize the Android apps as malware or goodware apps. The results show a significant improvement over other malware classification approaches.

For future work, more advanced classification techniques will be considered to detect advanced Zero-day malware attacks.

ACKNOWLEDGMENT

This work was supported by the Deanship of Scientific Research (DSR), King Abdulaziz University, Jeddah, Saudi Arabia, under Grant No. (830-863-D1435). The author, therefore, gratefully acknowledges the DSR technical and financial support.

REFERENCES

[1] GARTNER, "Gartner Says Worldwide Smartphone Sales Grew 3.9 Percent in First Quarter of 2016" http://www.gartner.com/newsroom/id/3323017

[2] Statista," number of available applications in the google play store" http://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/

[3] Kaspersky, "The Volume of New Mobile Malware Tripled in 2015" http://www.kaspersky.com/about/news/virus/2016/The_Volume_of_New_Mobile_Malware_Tripled_in_2015

[4] Felt, Adrienne Porter, Erika Chin, Steve Hanna, Dawn Song, and David Wagner. "Android permissions demystified." In Proceedings of the 18th ACM conference on Computer and communications security, pp. 627-638. ACM, 2011.

[5] Aung, Zarni, and Win Zaw. "Permission-based android malware detection."International Journal of Scientific and Technology Research 2, no. 3,pp. 228-234, 2013

[6] C.-Y. Huang, Y.-T. Tsai, and C.-H. Hsu, “Performance Evaluation on Permission-Based Detection for Android Malware” in Advances in Intelligent Systems and Applications-Volume 2, pp. 111–120, Springer, 2013.

[7] Liu, Xing, and Jiqiang Liu. "A two-layered permission-based android malware detection scheme." In Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2014 2nd IEEE International Conference on, pp. 142-148. IEEE, 2014.

http://dx.doi.org/10.1109/mobilecloud.2014.22

[8] Sharma, Akanksha, and Subrat Kumar Dash. "Mining api calls and permissions for android malware detection." In International Conference on Cryptology and Network Security, pp. 191-205. Springer International Publishing, 2014.

[9] Grace, Michael C., Yajin Zhou, Zhi Wang, and Xuxian Jiang. "Systematic Detection of Capability Leaks in Stock Android Smartphones." In NDSS, vol. 14, p. 19. 2012.

[10] Aafer, Yousra, Wenliang Du, and Heng Yin. "DroidAPIMiner: Mining API-level features for robust malware detection in android." In International Conference on Security and Privacy in Communication Systems, pp. 86-103. Springer International Publishing, 2013.

[11] Zhang, Mu, Yue Duan, Heng Yin, and Zhiruo Zhao. "Semantics-aware android malware classification using weighted contextual api dependency graphs." In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pp. 1105-1116. ACM, 2014.

[12] Huang, Jianjun, Xiangyu Zhang, Lin Tan, Peng Wang, and Bin Liang. "AsDroid: detecting stealthy behaviors in Android applications by user interface and program behavior contradiction." In Proceedings of the 36th International Conference on Software Engineering, pp. 1036-1046. ACM, 2014.

[13] Yerima, Suleiman Y., Sakir Sezer, Gavin McWilliams, and Igor Muttik. "A new android malware detection approach using bayesian classification." InAdvanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on, pp. 121-128. IEEE, 2013.

[14] Feng, Y., Anand, S., Dillig, I. and Aiken, A. Apposcopy: Semantics-based detection of android malware through static analysis. InProceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (pp. 576-587). ACM, 2014.

[15] Egele, Manuel, Theodoor Scholte, Engin Kirda, and Christopher Kruegel. "A survey on automated dynamic malware-analysis techniques and tools." ACM Computing Surveys (CSUR) 44, no. 2 (2012): 6.

[16] Lindorfer, Martina, Matthias Neugschwandtner, Lukas Weichselbaum, Yanick Fratantonio, Victor Van Der Veen, and Christian Platzer. "Andrubis--1,000,000 apps later: A view on current Android malware behaviors." In2014 Third International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS), pp. 3-17. IEEE, 2014.

[17] Burguera, Iker, Urko Zurutuza, and Simin Nadjm-Tehrani. "Crowdroid: behavior-based malware detection system for android." In Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices, pp. 15-26. ACM, 2011.

[18] Enck, William, Machigar Ongtang, and Patrick McDaniel. "On lightweight mobile phone application certification." In Proceedings of the 16th ACM conference on Computer and communications security, pp. 235-245. ACM, 2009.

[19] Sanz, Borja, Igor Santos, Carlos Laorden, Xabier Ugarte-Pedrero, Pablo Garcia Bringas, and Gonzalo Álvarez. "Puma: Permission usage to detect malware in android." In International Joint Conference CISIS’12-ICEUTE´ 12-SOCO´ 12 Special Sessions, pp. 289-298. Springer Berlin Heidelberg, 2013.

[20] B.Sanz, I.Santos, C.Laorden, X.Ugarte-Pedrero, J.Nieves, P.G.Bringas, G.Álvarez Marañón, MAMA: manifest analysis for malware detection in android, Cybern. Syst. 44, 6–7,2013

[21] Zhou, Yajin, and Xuxian Jiang. "Dissecting android malware: Characterization and evolution." In 2012 IEEE Symposium on Security and Privacy, pp. 95-109. IEEE, 2012.

[22] Schmidt, A-D., Rainer Bye, H-G. Schmidt, Jan Clausen, Osman Kiraz, Kamer A. Yuksel, Seyit Ahmet Camtepe, and Sahin Albayrak. "Static analysis of executables for collaborative malware detection on android." In2009 IEEE International Conference on Communications, pp. 1-5. IEEE, 2009.

[23] Aafer, Yousra, Wenliang Du, and Heng Yin. "DroidAPIMiner: Mining API-level features for robust malware detection in android." In International Conference on Security and Privacy in Communication Systems, pp. 86-103. Springer International Publishing, 2013.

[24] Shabtai, Asaf, Uri Kanonov, Yuval Elovici, Chanan Glezer, and Yael Weiss. "“Andromaly”: a behavioral malware detection framework for android devices." Journal of Intelligent Information Systems 38, no 1, pp.161-190, 2012.

[25] Bartel, Alexandre, Jacques Klein, Yves Le Traon, and Martin Monperrus. "Automatically securing permission-based software by reducing the attack surface: An application to android." In Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering, pp. 274-277. ACM, 2012.

[26] APKTool, Reverse Engineering with ApkTool, Available: https://code.google.com/android/apk-tool

[27] A. Feizollah, N. B. Anuar, R. Salleh, and A. W. A. Wahab, “A review on feature selection in mobile malware detection,” Digital Investigation, vol. 13, pp. 22 – 37, 2015.

[28] S.-H. Seo, A. Gupta, A. M. Sallam, E. Bertino, and K. Yim, “Detecting mobile malware threats to homeland security through static analysis,” Journal of Network and Computer Applications, vol. 38, no. 1, pp. 43–53, 2014.

[29] Watts, Duncan J., and Steven H. Strogatz. "Collective dynamics of ‘small-world’networks." nature 393, no. 6684 (1998): 440-442.

[30] Mori, Tatsunori. "Information gain ratio as term weight: the case of summarization of ir results." In Proceedings of the 19th international conference on Computational linguistics-Volume 1, pp. 1-7. Association for Computational Linguistics, 2002.

[31] Cortes, Corinna, and Vladimir Vapnik. "Support-vector networks." Machine learning 20, no. 3 (1995): 273-297.

[32] Shalev-Shwartz, Shai, Yoram Singer, Nathan Srebro, and Andrew Cotter. "Pegasos: Primal estimated sub-gradient solver for svm." Mathematical programming 127, no. 1 ,pp. 3-30,2011.

[33] Android Malware Genome Project, 2014. http://www.malgenomeproject.org/

[34] Abdulla, Shubair, and Altyeb Altaher. "Intelligent Approach for Android Malware Detection." KSII Transactions on Internet and Information Systems (TIIS) 9, no. 8, 2015.

[35] Zhou, Wu, Yajin Zhou, Xuxian Jiang, and Peng Ning. "Detecting repackaged smartphone applications in third-party android marketplaces." In Proceedings of the second ACM conference on Data and Application Security and Privacy, pp. 317-326. ACM, 2012.

[36] Arp, Daniel, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, and Konrad Rieck. "DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket." In NDSS. 2014.

[37] Grace, Michael, Yajin Zhou, Qiang Zhang, Shihong Zou, and Xuxian Jiang. "Riskranker: scalable and accurate zero-day android malware detection." InProceedings of the 10th international conference on Mobile systems, applications, and services, pp. 281-294. ACM, 2012.

[38] Faruki, Parvez, Vijay Laxmi, Ammar Bharmal, M. S. Gaur, and Vijay Ganmoor. "AndroSimilar: Robust signature for detecting variants of Android malware." Journal of Information Security and Applications ,22 pp.66-80,2015.

[39] T. Fawcett, "An introduction to ROC analysis," Pattern recognition letters, vol. 27, pp. 861-874, 2006.