Malware Detection in Android Operating System

Zinal D. Patel

Department of Computer Engineering,
Sarvajanik College of Engineering and Technology,
Athwalines, Surat, India
zinalpatel4395@gmail.com

Abstract—**Android operating system has gained a wide popularity in the recent years due to the open environmental nature of the Android framework. This has given a tremendous flux to the Malware developers to target the Android users. Thus, personal privacy theft has become a major issue. Various mobile malware detection systems are proposed in the recent years to address this issue. Operation level, Analysis technique and Accuracy are the core design aspects for any detection method. This paper focuses on surveying the existing mobile malware detection systems. Also, it provides a detailed comparison to summarize and analyze the detection methods under concern.**

Keywords- Android Operating System; Malware detection; Accuracy; Operation Level; Analysis technique

I. INTRODUCTION

Android operating system has gained a wide familiarity nowadays as smartphones have emanated as portable devices with increasingly sensing capabilities and powerful computing. The reason for considering Android operating system as a popular platform for smartphone is its open source nature and the support provided from Google.

The increasingly prevalence of Android operating system has also exerted influence on the Malware developers to target the users of smartphones having Android platform [1]. Unfortunately, the security threats have also increased prominently due to the installation of malicious software since Android allows installing applications from sources that are not duly verified. The security threats include leakage of personal information termed as personal privacy theft. Malware is defined as software that is specifically designed to damage the smartphone or gain authorized access to steal private information. Thus, detection of malware is very crucial.

There are two approaches of detecting malwares in Android operating system. The first is the signature based approach in which the derived signature of any application is checked against the malware database. Besides the signature based approach, the other is the behavioral detection. In this approach, the behavior of the application is checked at runtime with the malicious and normal behavioral profiles [1].

The security mechanism of Android for malware detection includes three types of analysis viz. Static, Dynamic and Hybrid. The detection systems are based on various analysis techniques such as feature analysis, system call analysis, network traffic analysis etc. The next section deals with the classification of the detection system based on the above criteria.

The later section of the paper provides a detailed explanation about the various existing detection systems. Moreover, a comparative analysis of the mentioned detection systems is also presented. Conclusion and future work are discussed in the last section.

II. RELATED WORK

The extant literature about detection of malware in Android operating system is presented in this section. Till date, so many techniques have been proffered. The classification of approaches for malware detection is shown in this section and depicted in Fig. 1.

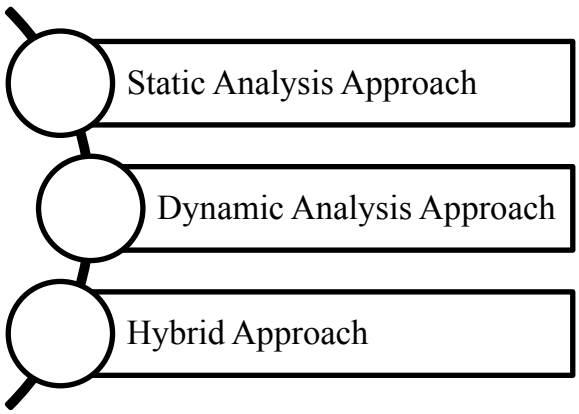

Fig. 1 Classification of existing approaches for malware detection

The dynamic analysis approach for detection of Android malware uses the features that are obtained dynamically at runtime or compile time. V.M. Afonso et al. [2] proffered a technique confiding on this approach. The features like system call traces and API calls were excerpted and then a machine learning algorithm is used for dynamically detecting whether the application is benign application or malicious application. A. Saracino et al. [3] proffered a detection system that confides upon the behavioral patterns.

ISBN: 978-1-5386-4119-4/18/$31.00 ©2018 IEEE

The patterns are distinct for different behavioral classes. The MADAM detection algorithm is utilized to detect the malware in an application based on the similarity metric defined in [3]. It is a host based detection system which can analyze the behavioral characteristics at kernel, package, user and application level simultaneously.

The hybrid approach utilizes the static features and the dynamic features for the detection of Android malware. F. Martinelli et al. [4] proffered a system named BRIDEMAID that uses Support Vector Machine (SVM) classifier for matching in static analysis while the dynamic analysis includes monitoring the behavior of user, application and the device. J. Malik et al. [5] proffered a system named CREDROID that utilizes network traffic analysis for the detection of malware in android operating system. The third party APIs are used which can detect the network address through static analysis. The Domain Name Server (DNS) queries and the data transmitted to the remote server are dynamically analyzed by a pattern based detection method.

The techniques based on static analysis approach are quite simple to implement and thus, a brief about the systems using static analysis technique is provided in the next section.

III. SUMMARYOF EXISTING SYSTEMS FOR MALWARE DETECTION

The systems that use a static approach for malware detection are described in this section.

A. *DREBIN*

DREBIN [6] is a Malware detection system that performs a broad static analysis on different sources to excerpt various feature sets that can be useful to detect the malicious activity. The method for this system is divided into four steps i.e. Broad static analysis, Embedding in vector space, Learning based detection and Explanation.

The broad static analysis step utilizes Android Asset Package Tool to excerpt the feature set from manifest file. This feature set includes hardware components, requested permissions, application components and filtered intents. Moreover, features like restricted API calls, utilized permissions, suspicious API calls and network address are also excerpted from disassembled code. These feature sets are embedded in a vector space with dimensions 0 and 1 using an indicator function I (x,s) where the value of function is 1 if an application x consists of feature s and 0 otherwise.

The third step utilizes the machine learning technique to learn separation between benign and malicious applications. Here, a linear SVM classifier is utilized to classify both classes associated with malware and benign applications respectively. A detection function F is utilized which is defined as

$$f(x) = \sum_{s \epsilon S} I(x,s) \cdot w_s = \sum_{s \epsilon x} w_s \qquad (1)$$

where I is indicator function and w_s clinches the weight. The explanation step provides an explanation for the detection results. This is achieved by the virtue of detection function which helps us to clinch the contribution of each feature in determining malicious applications.It is an efficient detection system.

B. *AndroDialysis*

AndroDialysis [7] is a malware detection system that utilizes Inter-process communication as a feature for detecting Android malwares as they are the notable features of Android framework which allow reutilization of components across process boundaries.

The architecture of the system comprises of three modules viz. Android application Framework, Detector and Utilizer Interface. The android application framework consists of the installed applications. The detector module further consists of four sub modules namely Decompiler, Extractor, Intelligent learner and Decision maker. Decompiler is utilized to decompile the Android Manifest.xml file and the dex file to generate a small version of Java code which is easily readable. The extractor sub module is utilized to excerpt implicit and explicit intent as well as intent filters and permissions from Java code using Beautiful Soap Package of python and store them in a feature database. This database is also passed to decision maker to clinch the maliciousness. The intelligent learner sub module is utilized to learn the pattern structure of data by the Bayesian network algorithm. The Bayesian network learns the network structure by using local score metrics. The local score metrics is calculated by various search algorithm like K2, Genetic-Search, Hill-Climber and LAGDHillClimber. Afterwards, the Bayesian network learns the probability tables using estimators. This sub module produces an output model for detection purpose.

The detector sub module receives the data sets from the extractor which is checked against the data set received from intelligent learner to detect whether the application is clean or malicious. This output is sent to the user interface so that visualized output can be seen by the user.

C. *ICCDetector*

ICCDetector [8] is a malware detection system that is dependent on the inter component communication which takes the communication of components within or across the application boundaries into account. The system architecture of ICC detector consists of two phase viz. Training phase and Detection phase.

The training phase deals with excerpting the ICC related features from the apk file of application. A parser named Epicc is utilized for this purpose. It excerpts components, explicit intents, implicit intents and intents filters as ICC related features. A feature vector is generated from these features which are passed to the detection phase. ICC detector utilizes any two of SVM, Decision tree and Random forest for learning the ICC patterns. Also, the Correlation based Feature Selection (CFS) method is utilized to remove redundant and irrelevant feature confiding on the correlation

between features. The output created likewise is passed to the detection model. The detection model detects the application as malicious application or benign application by comparing it with the ICC patterns obtained from training phase. Though it is based on ICC patterns, it is accurate.

D. APK Auditor

APK Auditor [9] is a malware detection system that is based on the permission features. The system architecture of APK auditor comprises of three main components viz. Android Client, Signature database and a Central server.

APK Auditor client sends an analysis request to the server for detecting the application as malicious or benign. The Malware in local applications and remote applications can be detected. Moreover, the client can also monitor permissions and the protection level by providing user interface. A resource usage limit can be set by the client and the application that goes beyond the usage limit can be killed. The APK auditor signature database excerpts the information about the permission and services from the application code and store it for detection of the testing application. APK Auditor server gets a list of applications from Play-store for analysis. The server downloads the application only if it was not previously analyzed. This analysis process is continuous and is based on Permission Malware Score (PMS) and Application Malware Score (AMS). PMS is calculated by the existence of that permission in all malware while AMS is calculated by adding up the permission malware score. The framework decided a limit by using logistic regression. If the AMS is above the limit then the application is reported to be malicious.

E. DroidNative

DroidNative [10] is the first malware detection system that utilizes the native code for detecting malware. It utilizes the control flow patterns for analysis of the native code. The main component of the system architecture comprises of Disassembler, Optimizer, Mail generator and Malware detector.

The Linear Sweep Technique and the Recursive Traversal Technique are utilized by the disassembler to disassemble the provided native code and generate the control flow patterns. There are many instructions in the native code which are not necessary for malware detection. Thus the optimizer is used to reduce the number of instructions and also build smaller and multiple Control Flow Graphs (CFGs) instead of one large CFG to reduce the runtime. Malware Analysis Intermediate Language (MAIL) generator is utilized to transform the instruction to MAIL statement. MAIL is an intermediate language that is utilized for malware analysis. The patterns present in MAIL annotate a CFG which is further utilized in pattern matching to detect malware.

The Malware Detector further comprises of components like Data Miner, Signature Generator and Similarity Detector. Behavioral signatures like Annotated Control Flow Graph (ACFG) or Sliding Window of Difference (SWOD) are generated by the Signature generator using the structural information present in MAIL which is searched by Data Miner. The training data set is initially utilized to generate a template for malicious or benign application confiding on the behavioral signatures. Then, a similarity detector utilizes a simple binary tree classifier to check the test data set by comparing the behavioral signatures of the template and classifying the application as benign application or malicious application.

F. DMDAM

DMDAM [11] is a malware detection system that utilizes the permission feature for the detection of android malware. It is a pure data mining based detection system.

The initial phase excerpts the permission feature from the manifest file using the AndroGuard tool. The <use permission> tag defines the permission and thus this tag is excerpted. A feature vector is created in which the presence of permission is denoted by 0 while its absence is denoted by 1.

The mining algorithm is applied after removing the irrelevant attributes. A feature ranking tool named Information Gain was applied to the feature set. Different classifiers like Naïve Bayer, Random Forest, PART etc. are applied to classify the application as malicious application or benign application.

G. API based system

This system utilizes a learning approach that confides on the Application Program Interface (API) for detection of Android malware [12]. The system framework consists of two phase viz. Training phase and Identification phase.

The training phase discovers sensitive data transmission paths using the data flow related APIs which are excerpted from the byte code. The byte code is first translated into an intermediate representation Jimple. Then the control flow of an application is clinched by constructing a call graph. Also, the dynamically registered broadcast receivers are discovered. All these features are utilized to construct a feature vector. Several algorithms like Bayesian Network, K-Nearest Neighbor and Logistic Regression (LR) learns the feature vector to generate and train the classifiers. The LR algorithm is utilized to calculate a malicious-weight value vector based on the weights of dataflow-related APIs. The maximum weight scores are utilized to generate feature database which further trains the classifiers. The Identification phase reports the application as malicious application or benign application by comparing the feature vector of test data set with the feature vector of training data set.

IV. COMPARATIVE ANALYSIS

An overall resemblance of the above discussed systems is presented in this section in Table 1. The comparison of the

system is advantageous to obtain an insight to the systems using static analysis approach for Android Malware detection. This in turn, is used to analyze the mentioned systems under specific concern. The metrics mentioned in the table are used to assess the performance and working of the system. A brief about the parameters is provided in this section.

TABLE 1 COMPARISON OF THE EXISTING SYSTEMS FOR MALWARE DETECTION

Parameter	DREBIN	AndroDialysis	ICCDetector	APK Auditor	DroidNative	DMDAM	API based system
Base of detection	Machine learning	Bayesian Network	Machine learning	Client-server learning approach	Machine learning	Machine learning	Machine learning
Analysis Technique	Feature	Feature	ICC	Feature	Semantic based signature	Feature	API
Accuracy	High	High	High	High	High	Medium	High
Evaluation Metric	Detection function	Local score	Feature vector	Application malware score (AMS)	Index based array	Feature vector	Standardization accuracy
Operation Level	Byte code	Byte code	Byte code	Byte code	Native code	Byte code	Byte code
Classifier/ Tool	SVM classifier	Andro-guard, Beautiful soup package	EPICC parser, SVM classifier	-	ART compiler, Decision tree classifier	Weka, Multiple classifiers	Dexpler, KNN classifier
Output of system	Decimal value	Graphical pie chart	Decimal value	Binary value	Malware tag	Malware tag	Malware tag/ Binary value
Scope of improvement	Lacks dynamic inspection and Possibility of mimicry and poisoning attacks	Lacks dynamic inspection	-	Makes system unavailable when external system dependencies are out of service	Unable to detect malware employing excessive control flow changes	-	Loss in accuracy if overhead is reduced

The metrics are defined as follows:

1. *The Base of Detection* parameter defines the core design technique used for detection of malware.
2. *Analysis Technique*represents the technique that is used for static analysis. The different analysis techniques are feature analysis, API call analysis, Network traffic analysis, etc.
3. *Accuracy* parameter defines the performance of the system as how accurate the system is in detecting the Android malware.
4. *Evaluation Metric* parameter defines the metric used in the system for detecting the Android malware.
5. *Operation level* parameter defines code level at which the system works. There are two operation levels viz. Byte code level and Native code level
6. *Classifier/Tool* parameter defines the tool/s and classifier/s that are used during any phase of the systemfor performing malware detection.
7. *Output of system* parameter defines the way in which the results are presented to the user.
8. *Scope of improvement* parameter defines the strategies or the restrictions that can be further used to improve the overall performance of the system.

V. CONCLUSION

Malware Detection in Android Operating System is considered as a crucial requirement to protect the users of smartphones from personal privacy theft. Over the past few years, many efficient techniques for malware detection are designed. A detailed survey of some accurate systems based on static analysis approach is presented in this paper. Each system is explained and analyzed. Moreover, all the systems are resembled for gaining a brief about the techniques.

Research work in this field is still going on so that more accurate and reliable systems can be developed in the future.

REFERENCES

[1] A. Bose, X. Hu, K.G. Shin and T. Park, "Behavioral detection of malware on mobile handsets," in *Proceedings of the 6th international conference on Mobile systems, applications, and services*, Breckenridge, USA, 2008, pp. 225-238.

[2] V.M.Afonso, M.F. de Amorim, A.R.A.Grégio, G.B.Junquera and P.L. de Geus, "Identifying Android malware using dynamically obtained features," *Journal of Computer Virology and Hacking Techniques*, vol. 11, no. 1, pp. 9-17, Feb. 2015.

[3] A.Saracino, D.Sgandurra, G.Dini and F. Martinelli, "MADAM: Effective and Efficient Behavior-based Android Malware Detection and Prevention,"*IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 1, pp. 83-97, Feb. 2018.

[4] F.Martinelli, F. Mercaldo and A. Saracino, "BRIDEMAID: An Hybrid Tool for Accurate Detection of Android Malware," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*,Abu Dhabi, United Arab Emirates, pp. 899-901, 2017.

[5] J. Malik and R. Kaushal, "CREDROID: Android malware detection by network traffic analysis," in *Proceedings of the 1st ACM Workshop on Privacy-Aware Mobile Computing*, Paderborn, Germany,pp. 28-36, 2016.

[6] D. Arp, M.Spreitzenbarth, M.Hubner, H. Gascon and K Rieck, "DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket," in *NDSS*, San Diego, USA, 2014.

[7] A.Feizollah, N.B.Anuar, R.Salleh, G. Suarez-Tangil and S. Furnell, "AndroDialysis: Analysis of Android Intent Effectiveness in Malware Detection," *Computers & Security*, vol. 65, pp. 121-134, Mar. 2017.

[8] K.Xu, Y. Li and R.H. Deng, "ICCDetector: ICC-Based Malware Detection on Android," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 6, pp. 1252-1264, Jun. 2016.

[9] T. Chen, R. Xu, Y. He and X. Wang, "APK Auditor: Permission-based Android malware detection system," *Digital Investigation*, vol. 13, pp. 1-14, Jun. 2015.

[10] S.Alam, Z.Qu, R. Riley, Y. Chen and V.Rastogi, "DroidNative: Automating and optimizing detection of Android native code malware variants," *Computers & Security*, vol. 65, pp. 230-246, Mar. 2017.

[11] A Bhattacharya and R.T.Goswami, "DMDAM: Data Mining Based Detection of Android Malware," in *Proceedings of the First International Conference on Intelligent Computing and Communication*, Singapore, vol. 458, pp. 187-194, 2017.

[12] S. Wu, P. Wang, X. Li and Y Zhang, "Effective detection of android malware based on the usage of data flow APIs and machine learning," *Information and Software Technology*, vol. 75, pp. 17-25, Jul. 2016.