STUDY OF VARIOUS APPROACHES FOR MALWARE DETECTION IN ANDROID ENVIRONMENT

Rajesh Surana
School of Computing, Informatics and Decision Systems Engineering
Arizona State University
Tempe, US
rsurana@asu.edu

***Abstract*—This project covers different approaches for malware detection in Android. We broadly classified these approaches in 3 categories- static, dynamic and behavioral analysis. It also includes comparison of these approaches in terms of pros and cons.**

***Index Terms*—malware analysis, android, dynamic, static, behavioral.**

I. INTRODUCTION

Day-by-day the number of android users are increasing and people have started to store really valuable data in their 'smartphones'. Report of International Data Corporation states that in the third quarter of 2013, 83% out of the total mobiles shipped were Android smartphones. This made android very tempting target for the attackers and hackers. Notion behind this is that even one malicious program could get them large user base to victimize. Security of Android phones has become even more difficult due to the limitations of mobile devices such as less battery, low processing power, limited memory, and size. Malware analysis involves studying applications for their malicious intent and stopping them in advance. In this report, we discuss three different approaches for malware detection in android environment and asses each of them for their strengths and weaknesses.

Our first approach is static analysis in which we study various methods and discuss classification and data mining approaches in specific. We cover signature based analysis and how classification is performed in static analysis. In our second approach, dynamic analysis, we run the application in simulated controlled environment to generate log of all activities like network messages, SMS, phone call etc. in order to detect hidden suspicious nature of the application. In general, dynamic analysis is performed over cloud due to the limitations of smartphones mentioned before. Our third approach, behavioral analysis uses different attributes to build the android environment by collecting and storing them in the form of vectors and then employing machine learning or classification algorithms, support vector machines and Naïve Bayesian techniques to analyze the applications.

II. OVERVIEW

It is divided into 3 sections for 3 approaches-static, dynamic and behavioral-

A. Static Analysis

There are applications in the market that try to steal user financial and social networking data and using it for their benefit. Static analysis can be used as a light-weight malware detecting process as compared to behavioral and dynamic analysis [1]. It is performed before installing the application. Following are the robust and computationally lightweight static analysis approaches.

1) Classification in Malware Detection

It is very difficult to determine whether application is going to use the private data only for the purpose for which we have granted the permission. One solution is to use data mining. Techniques like classification, vector model analysis and k-nearest neighbors can be employed for this purpose. API, parameter and package level information from benign and malicious applications can be used in learning model to detect malicious applications. This is a four step process-

a) Broad static analysis- Here, we define the various features of the application to be considered in our leaning model.

b) Generating a learning model- Here, we use large number of applications to make training and testing set for building a learning model. Google Play could be the source of these applications. Only the useful attributes are kept and rest are discarded out of the total selected in step 1.

c) Learning based detection- In this phase we actually test the application for its benignity or malicious intent. Learning model could be in the form of decision tree or support vector machine.

d) Explanation-We present the result of step 3 to the user in the easily understandable format and then user can decide to keep or uninstall the application.

In all, this classification technique has very high accuracy rate and very low false positive rate. Although, it fails for obfuscation technique and tend to be inaccurate if sufficient training and testing data is not available.

2) *Static Signature Based Detection*

In static signature based detection [2], we directly inspect the sequence of code for the malicious intent of the application. Output is also used to decide the appropriate approach in the dynamic analysis for the application under inspection. Let's see one such model Static Analyzer of Vicious Executables (SAVE). Model proposed by A. H. Sung, J. Xu, P. Chavez & S. Mukkamala can be used to detect obfuscated (or polymorphic) [3] malware and mutated (or metamorphic) [4] malwares. Key here is that various variations of the malware share the same core signature which can help to detect malware's unknown variations. Suppose signature of a malware is S and that of its variant is S', then similarity measures have to be calculated to detect that both belong to the same family. Algorithm for similarity measure calculation is-

a) Windows Portable Executable (PE) [5] is decompressed and then passed to PE binary parser which produce API sequence.

b) This API sequence with already known sequences stored in a database are passed to similarity measurement module which creates similarity report. Similarity detection methods could be Euclidian distance [6] and sequence alignment [7].

In all, SAVE is better than current commercial antivirus products in the market in terms of detecting malwares with obfuscation.

B. Dynamic Analysis

Due to the limitation of static analysis to detect the malwares with self-modifying code we use dynamic analysis for this purpose. Invocation of API and system call by executing applications are used in the dynamic analysis for determining malicious intent. First, we will see two sandboxes and then various contemporary platforms for dynamic analysis of malwares. In terms of computer security, Sandbox is the virtual machine in which application can be run and isolated from the actual resources.

1) Dynamic analysis sandboxes

a) TaintDroid

The motive of TaintDroid [8] is to detect whether application is sending private information over internet to the advertising server. Here, a label is assigned to each private information called as taint for tracking purpose. This process is called as taint analysis. TaintDroid uses four level tracking viz. message, variable, method, file level tracking.

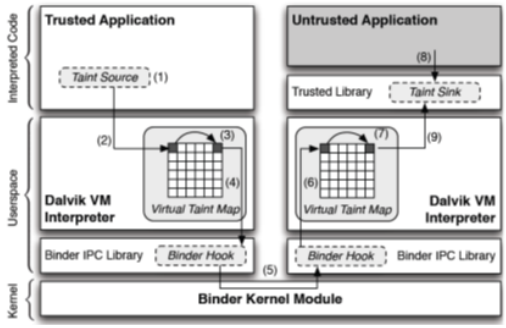

Fig.1 TaintDroid-Architecture with Android [8]

Three important parts of TaintDroid architecture are-

1. Taint Source-Location where private data is generated. E.g. GPS hardware.
2. Taint Propagation- Changes in data when it is passed to various processes or stored in a file.
3. Taint Sink- Place where data leaves the device to outside internet. E.g. API call for message sent over network. Traces are generated here.

Geographic location, phone identifiers, microphone input are some of the private data considered in the taint analysis. TaintDroid is very lightweight and have ability to differentiate between different types of private data. Although, it causes many false positives when configuration information is involved in tracking.

b) TraceDroid

(Victor van der Veen, August 2013) [9] in his Master thesis created TraceDroid which is a modified Android OS that creates traces when application is executed over it. Detailed information on app's internal process could be displayed by installing hooks in the application byte-code interpreter.

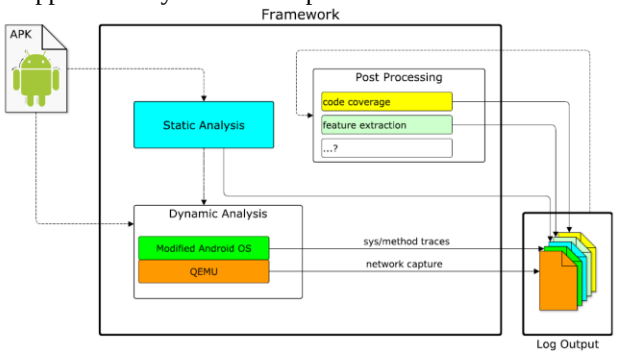

Fig. 2 TraceDroid Architecture [9]

Initially, static analysis is used to decide the approach in dynamic analysis which creates traces in log directory. These logs are post processed to represent the result in easy understandable format. SMS messages and incoming call can be emulated in the TraceDroid. Although, the coverage of TraceDroid is as good as manual testing, it is significantly low. It is 50% faster than Android's original profiler and has very nice GUI.

2) Dynamic Malware Analysis Platforms

In this section we will see two different dynamic malware analysis platforms viz. ANANAS, ANDRUBIS.

a) ANANAS framework

ANANAS [10] is divided into two parts core framework and different analysis plugins. There are several events in the framework to which plugins can register thereby deciding the configuration of the framework. Whenever particular event occurs, code in all the registered plugins is executed and result is created in the raw log file. SQL queries are used to create statistical data from logs. As every plugin has its own setting, optimizing the framework for individual need is easier. Although, there are only 6 plugins are available and scripting has to be done by human.

b) ANDRUBIS platform

ANDRUBIS [11], a hybrid Android malware analysis platform uses tools, like TaintDroid [8] and apktool [12] for dynamic analysis. This platform try to cover as many execution paths as possible. When application is loaded in the simulation environment predefined entry points are already known from the static analysis. During runtime, entry points are dynamically registered. Application Exerciser Monkey is used for simulating real time interactions such as file upload, button clicks, text input, etc. In addition to tracking Dalvik VM, it also tracks native code execution. In terms of code coverage it is better than ANANAS.

C. Behavioral Analysis

In behavioral analysis, data is collected from running applications and then passed to machine learning classifiers to alarm user when certain parameters exceed threshold.

1) Analysis using Machine Learning Classifiers

This method discussed in [13], has two parts: Data processing- It vectors the data collected from resources used by running applications. Malware detection analysis- it applies classification methods such as Naïve- Bayesian, Random Forest, Support Vector Machines and Logistic Regression to vectored data and creates result. Out of these random forest method has high true positive rate and low false positive rate.

2) SMARTMAL: Service Oriented Behavioral Analysis

It works on service oriented concept. Client consists of Feature Extractor which extracts features and stores in the form of vector and sent to server using communication module. Server consists of database to stores these vectored data. Detection module classifies data based on training set and creates result. Client manager manages new clients and GUI for interaction [14].

Algorithm could be defined as [14]

1. Obtain probability distribution for the normalized data.
2. Define the observation window.
3. Select sampling window observation from observation window by sampling.
4. Calculate the internal and external distance for probability correlation of normalized data.
5. If Ext dist > Int Dist

The data vector is abnormal and is a malware, add it to abnormal distribution.

6. Else, the data vector is normal and proceed to next vector. Parallel processing can be used to share server load.

3) ANDROMALY

It is lightweight security analysis framework which chooses better feature extraction method, classifier and features to be analyzed. Like SmartMal, it has Feature Extractor and processor who detects the malicious behavior based on collected features and classifier selected. Threat weights can be used in Alert Messager for further classification using K- Means, Bayesian Networks, Naïve Bayesian, Histograms, Logistic Regression and Decision Trees. Andromaly can decide whether new device can detect malwares at the same level as that of trained device. Its result are limited due to lack of awareness of attacks on android environment.

4) Instrumentation System

Instrumentation system [15] involves automating the user and application interactions. There are four major steps –

1. GUI extraction and mapping - It is done in the instrumentation engine.
2. Event preparation and scheduling - Global event scheduler schedules the events and then passes to instrumentation engine.
3. Event Injection - Events are injected into GUI threads.
4. Logging - Logs are generated in this stage.

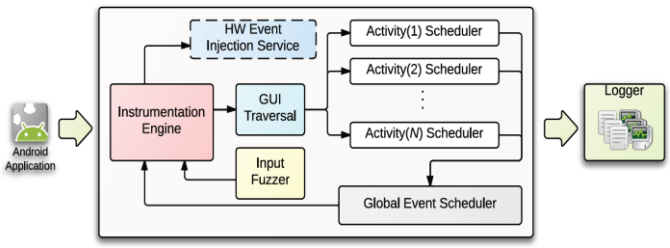

Fig.3 Instrumentation system architecture [15]

5) Intrusion detection system based on the Android Platform

Architecture [16] is divided into 3 parts –

1. Data Extraction – Extracts the features and send them to Analysis engine.
2. Data Analysis Engine- Detection algorithm like Naive Bayes Classifier is executed and suspicious activities detected.
3. Response- The result of the data analysis engine is forwarded to Response which stores it in log. Administrator would take appropriate action based on the result in the log.

6) MDTN system

MDTN system [17] is used to analyze the smartphone application before installing as well as while run time. There are three modules in the MDTN system. They are

1. Monitoring, detecting and tracking module
2. Cloud computational module
3. Notification module

Performance of the MDTN is evaluated using throughput to analyze malwares. Throughput here is total number of completed tasks.

III. My contributions

My part was to study current sandboxes used in the dynamic analysis. My analysis about two sandboxes viz. TaintDroid and TraceDroid is already explained in the dynamic analysis subsection of the overview section. Reason behind choosing these two sandboxes was that these two sandboxes are used in many current dynamic analysis frameworks. In addition, I studied the importance as well as challenges of the dynamic analysis. I also studied and presented different anti-analysis techniques used to hinder dynamic analysis by detecting the simulation environment from real one. I also presented the comparison of dynamic analysis with static analysis in order to emphasize the importance of dynamic analysis.

IV. Conclusion

Static analysis is a lightweight, high performance technique for malware analysis. Obfuscation techniques and reverse engineering can be used to thwart the analysis. Finite-based analytic model could induce imprecision in the static analysis. Also, in case of applications with multiple entry points, static analysis incurs large performance overhead. In case of signature based detection, unknown enemies cannot be detected.

Dynamic analysis overcomes the limitations of static analysis and could detect malwares with obfuscation as well as self-modifying code techniques. API and system call are used in the dynamic analysis to generate traces. As compared to static analysis, this technique is heavy. Code coverage is dependent on the events that can be simulated in the sandboxes.

Behavioral analysis uses classification techniques to decide whether the behavior of the application under analysis is anomalous or not. Various tools such as SmartMal and ANDROMALY could be used for behavioral malware analysis. Biggest advantage of behavioral analysis is its ability to detect unknown malwares.

V. New skills, techniques, or knowledge acquired

In the starting phase, I understood different types of malwares and their working. I also learnt how malwares of the PCs and laptops are different than that of smartphones in terms of size, working and impact on devices. To understand different approaches for malware analysis techniques in android environment, it is at the heart to understand the internal architecture of android system. Understanding the working of Dalvik virtual machine and process communication in the android was crucial in studying the dynamic analysis. I know the working of sandboxes and their typical architecture by studying contemporary sandboxes. From all the three approaches that I learnt, I found that behavioral analysis is better than the other two approaches when performance overhead is not an issue. On the other hand, when performance and time constraint comes into picture, static analysis is the best techniques. Dynamic analysis is in the middle of static and behavioral analysis in terms of performance overhead and malware detection rate.

VI. Team Members

- Rajesh Surana
- Yuli Deng
- Sagar Karri
- Tejaswini Kantheti
- Sidharth Ramesh
- Manish Trivedi

Acknowledgment

I thank Professor Yau for such a wonderful opportunity and his valuable guidance for this project.

References

[1] Kriti Sharma, Trushank Dand, Tae Oh and William Stackpole, “Malware Analysis for Android Operating,” in 8th Annual Symposium of Information Assurance, June 4-5, Albany, NY.

[2] Nwokedi Idika, Aditya P. Mathur, “A Survey of Malware Detection Techniques”, Research at Purdue University, 2007.

[3] Gerald R. Thompson and Lori A. Flynn, “Polymorphic Malware Detection and Identification via Context-Free Grammar Homomorphism”, Bell Labs Technical Journal, Volume 12, Issue 3, 2007.

[4] Ashu Sharma, S. K. Sahay, “Evolution and Detection of Polymorphic and Metamorphic Malwares”, Survey in Bits Pilani, Goa.

[5] Microsoft Portable Executable and Common Object File Format Specification, Revision 8.2, 21 September 2010.

[6] Nathan Krislock, Henry Wolkowicz, “Euclidean Distance Matrices and Applications”, Handbook of Semidefinite, Cone and Polynomial Optimization, 2010.

[7] Vivek Kumar, Sadhna K. Mishra, “Detection of Malware by using Sequence Alignment Strategy and Data Mining Techniques”, International Journal of Computer Applications, Volume 61 - Number 22, 2013.

[8] William Enck, Peter Gilbert, Byung-Gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, Anmol N. Sheth, “TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones”, 9th USENIX Symposium on Operating Systems Design and Implementation, 2010.

[9] Victor van der Veen, “Dynamic Analysis of Android Malware”, Master thesis, August 2013.

[10] Thomas Eder, Michael Rodler, Dieter Vymazal, and Markus Zeilinger. 2013. “ANANAS - A Framework for Analyzing Android Applications.”

[11] Martina Lindorfer, Matthias Neugschwandtner, Lukas Weichselbaum., “ANDRUBIS - 1,000,000 Apps Later: A View on Current Android Malware Behaviors”.

[12] Apktool. [Online]. Available: http://code.google.com/p/android-apktool/

[13] Hyo-Sik Ham, Mi-Jung Choi,“Analysis of Android Malware Detection Performance using Machine Learning Classifiers”, 2013 International Conference on ICT Convergence (ICTC), 2013, pp. 490 – 495.

[14] Chao Wang, Zhizhong Wu, Xi Li, Xuehai Zhou, Aili Wang, and Patrick C. K. Hung, “SmartMal: A Service-Oriented Behavioral Malware Detection Framework for Mobile Devices”, August 2014.

[15] Mohammad Karami, Mohamed Elsabagh, Parnian Najafiborazjani, and Angelos Stavrou,“Behavioral Analysis of malware detection in Android Applications using Automated Instrumentation”, Computer Science Department, George Mason University, Fairfax.

[16] Fangfang Yuan, Lidong Zhai, Yanon Cao.,”Research of Intrusion Detection System on Android”.

[17] Maya Louk, Hyotaek Lim, and HoonJae Lee”An Analysis of Security System for Intrusion in Smartphone Environment”,August 2014.