ISSN: 2321-7782 (Online)
Volume 3, Issue 3, March 2015
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

Cloud Computing Based Forensic Analysis for Mobile Applications Using Data Mining

Aditya Parashar[1]
Department Of Computer Engineering
Padmabhooshan Vasantdada Patil Institute of Technology
Pune, Maharashtra , India

Nishant Paliwal[2]
Department Of Computer Engineering
Padmabhooshan Vasantdada Patil Institute Of Technology
Pune, Maharashtra , India

Rahul Shelke[3]
Department Of Computer Engineering
Padmabhooshan Vasantdada Patil Institute Of Technology
Pune, Maharashtra , India

Abstract: ***With the explosive increase in mobile apps, more and more threats migrate from traditional PC client to mobile device. Compared with traditional Win+Intel alliance in PC, Android+ARM alliance dominates in Mobile Internet, the apps replace the PC client software as the major target of malicious usage. In this paper, to improve the security status of current mobile apps, we propose a methodology to evaluate mobile apps based on cloud computing platform and data mining. We also present a prototype system named AndroidArmour to identify the mobile app's virulence or benignancy. Compared with traditional method, such as permission pattern based method, AndroidArmour combines the dynamic and static analysis methods to comprehensively evaluate an Android app. In the implementation, we adopt Android Security Evaluation Framework (ASEF) and Static Android Analysis Framework (SAAF), the two representative dynamic and static analysis methods, to evaluate the Android apps and estimate the total time needed to evaluate all the apps stored in one mobile app market. As mobile app market serves as the main line of defence against mobile malwares, our evaluation results show that it is practical to use cloud computing platform and data mining to verify all stored apps routinely to filter out malware apps from mobile app markets. As the future work, AndroidArmour can extensively use machine learning to conduct automotive forensic analysis of mobile apps based on the generated multifaceted data in this stage.***

Key words: ***Android platform; mobile malware detection, cloud computing, forensic analysis, machine learning, redis key-value store data mining.***

I. INTRODUCTION

Due to introduction of Android OS in smart phones, there has been a massive increase in number of android applications. There are about 800000 Android Apps in Google play market and the total download is about 48 billion as of MAY 2013. Beside Google AppStore there are different AppStores which provide Android Users an easy way of downloading application. Due to this there is increase in mobile threats as different AppStores use malware detection strategies. Some sophisticated Malwares can escape from detection and harm device. In this paper based on cloud computing and data mining, we propose a methodology to evaluate mobile apps for improving current security status of mobile apps. AndroidArmour a demo and prototype system is also proposed to identify the mobile app's virulence and benignacy.to do this we will be using two frameworks which will be responsible for dynamic and static analysis of android application. Two frameworks used in this system are Android Security Evaluation Framework (ASEF) and Static Android Analysis Framework (SAAF).

Motivation

Recently, it has been noticed that large numbers of repackaged Android apps showing up in app stores. While these apps pretend to be “free”, in the end they cost the users time and money: they are either shown various ads or they are subscribed to various premium SMS numbers.

With more and more people using their smartphones and tablets to surf the web, update social networking sites, and shop & bank online, cybercriminals and malware are increasingly targeting mobile devices – with new smartphone threats and mobile threats. New Apps are developed to harm privacy of the user as well data from the mobile device. Our aim is to detect these types of apps. The main motivation of our project is to make sure all android apps stored on app store are not goint to harm system

II. Literature Survey

Malwares affect the working of application and also hampers the security of a smart phone.

It needs some discussion about origins of malwares and their spreading. Following are the reasons for presence of large number of malwares.

I. Android platform allows users to install apps from third party market place which might have poor defense strategy against malwares.

II. It is easy to port an existing windows based Botnet client to android platform.

III. Android application developers can upload their apps without any trustworthiness.

IV. A number of apps have been modified and the malwares have been packed in and spread through unofficial repositories.

Malicious behavior of android malware includes controlling mobile device without user intervention. Malware activities are as following:

I. Privilege escalation to root

II. Leak private data and exfilterate sensitive data

III. Dial premium numbers

IV. Botnet activities

V. Backdoor triggered SMS

There are lot of already discovered malwares such as Drad.A,Geinimi,PJApps,HippoSMS,Fake Netflix, Walk & Text , Dogwars, DougaLeaker. A,Gone in 60 s.

III. Problem Statement

With the explosive increase in mobile apps, more and more threats migrate from traditional PC client to mobile device. Compared with traditional Win+Intel alliance in PC, Android+ARM alliance dominates in Mobile Internet, the apps replace the PC client software as the major target of malicious usage. With this system we are going to solve the problem of mailicious apps. We are going to build safe app store where only legitimate apps will store which are not going to harm in any way to the users mobile device.

IV. Proposed System

Name of proposed system is AndroidArmour. System will scan the android application with the help of SAAF and ASEF as soon as the application is uploaded on AppStore by an Valid Uploader. If ASEF and SAAF give satisfactory results regarding

respective app only then that application will be allowed to get downloaded. Following data will provide details regarding Frontend and Backend of a System.

FRONTEND:

AndroidArmour has a web Frontend which is based on the SpringSource's spring framework and twitter bootstrap. It will provide facilities to upload application analyze application and check reviews of application given by system to the developer. It will also provide download facility to the smart phone users.

B**ACKEND:**

ASEF:

ASEF is an automated tool which can be used to analyze android application. When you upload an apk file of application it will start the ADB logging and traffic sniffing using TCPDUMP. Then it will launch an Android Virtual Machine (AVM) and install the application on it. After that ASEF begins to launch app to analyze and send the number of random gestures to simulate human integration on the application. Meanwhile ASEF also compares the log of android virtual machines with a CVE library and its the internet activity with Google Safe Browser API. After certain number of gestures are sent to virtual machine the test of circle is ended and the application will be uninstalled. Then ASEF will begin to analyze the log file and the internet traffic that the app generated. ASEF uses Google Safe Browsing API to find out whether the URLs the app try to reach are malicious or not. ASEF also checks the existed vulnerability list to find out whether the application has some serious vulnerability.

SAAF:

SAAF is static analyzer for android app's apk file. It can extract content of apk files and the decode the content to smali code , then it will apply program slicing on the smali code , to analyze the permission of apps , match heuristic patterns and perform program slicing for function of interest.

Work Flow of System:

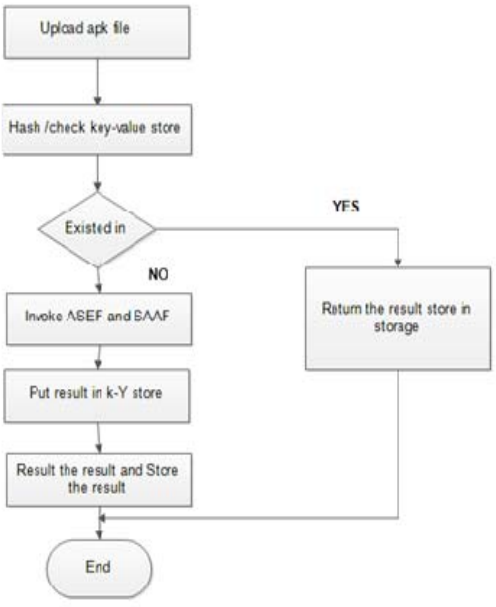

Figure 1 Basic Work Flow Of Proposed System

Storage of applications and analyzed data:

The application and analyzed data is stored in SQL database using MySQL. This database is stored on the virtual cloud.

V. ALGORITHMIC STRATEGIES

Algorithms:

In this project we are going to use two algorithms:

1. **ASEF**
2. **SAAF**

Whether it's a bandwidth-hogging app, aggressive adware or even malware, it would be interesting to know if they are doing more than what they are supposed to and if your personal information is exposed. Is there really a way to automatically evaluate all your apps - even hundreds of them - to harvest their behavioural data, analyze their run pattern, and at the same time provide an interface to facilitate a vast majority of evolving security tests with most practical solutions?

Android Security Evaluation Framework (ASEF) performs this analysis while alerting you about other possible issues. It will make you aware of unusual activities of your apps, will expose vulnerable components and help narrow down suspicious apps for further manual research. The framework will take a set of apps (either pre-installed on a device or as individual APK files) and migrate them to the test suite where it will run it through test cycles on a pre-configured Android Virtual Device (AVD).

During the test cycles the apps will be installed and launched on the AVD. ASEF will trigger certain behaviours by sending random or custom gestures and later uninstall the app automatically. It will capture log events, network traffic, kernel logs, memory dump, running processes and other parameters at every stage which will later be utilized by the ASEF analyzer. The analyzer will try to determine the aggressive bandwidth usage, interaction with any command and control (C&C) servers using Google's safe browsing API, permission mappings and known security flaws. ASEF can easily be integrated with other open source tools to capture sensitive information, such as SIM cards, phone numbers and others.

ASEF is an Open Source tool for scanning Android Devices for security evaluation. Users will gain access to security aspects of android apps by using this tool with its default settings. An advanced user can fine-tune this, expand upon this idea by easily integrating more test scenarios, or even find patterns out of the data it already collects. ASEF will provide automated application testing and facilitate a plug and play kind of environment to keep up with the dynamic field of Android Security.

VI. MATHEMATICAL MODULE

Set theory

Let S be the Set of System,

S={s, e, X, Y, Fs, success, failure}

Where,

S = s is the Start State of the System

E = e is the end State of the System

X = Set of Input Parameter

X= {X1}

Where,

X1= {input to the system is number of applications given or uploaded by the user}

Fs = {Functions used in the Program}

Fs = {F_1, F_2, F3, F4}

Let, F_2 be used to the recognize and decode the speech

F_1 be used for ASEF algorithm

F3 be the for SAAF

F4 function when malicious app is found

Y = {Output of the System}

Success Case: Malicious App is found and not uploaded to the system.

Failure Case: Any app will upload.

VII. State Diagram

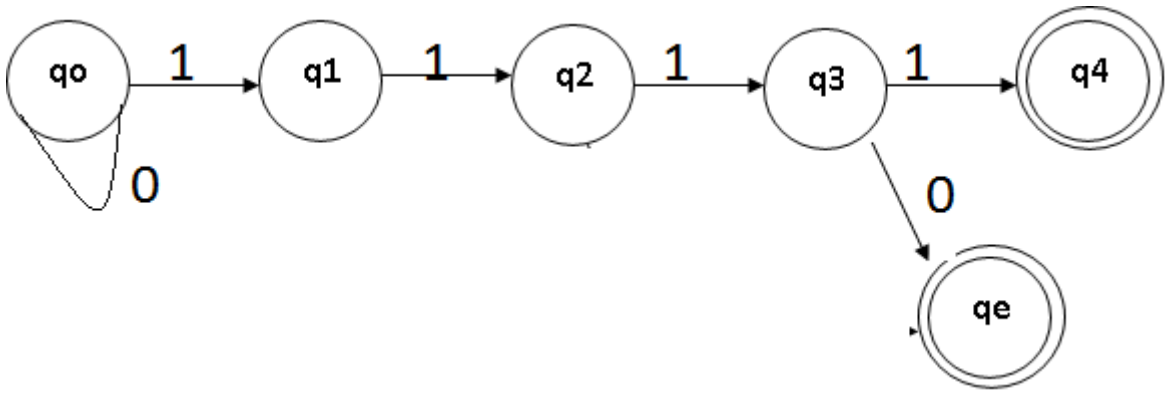

Figure 2 State Diagram

Where,

q → Initial State(app sender)

q1 → hash value is calculated.

q2 → App is compared with database whether exist or not.

q3 → Perform ASEF and SAAF.

q4 → Desired output.

qe → Any error state.

End State: {q4, Se}

VIII. Experimental Analysis

Analysis of SAAF:

We apply SAAF to 25 Android apps downloaded from different AppStore for static smali code analysis, to evaluate the performance of this tool. We found that the most time consuming step of SAAF is the slicing step, and the second is the permission categorizing step. The average time of analyzing one app consumed by SAAF in one Linux virtual machine, which runs on Intel-i5 four-core CPU with 4 GB of memory, is about 33.93 s.

We found that the analyzing of different apps will consume different times, and the total time depends on the complexity of apps, such as the amount of methods etc. But for most apps, SAAF will finish the analysis in an acceptable period.

Analysis Of ASEF:

We have calculated the time required for ASEF to analyze an app. We have analyzed 20 applications and recorded time of analysis .Following bar graph will provide further details.

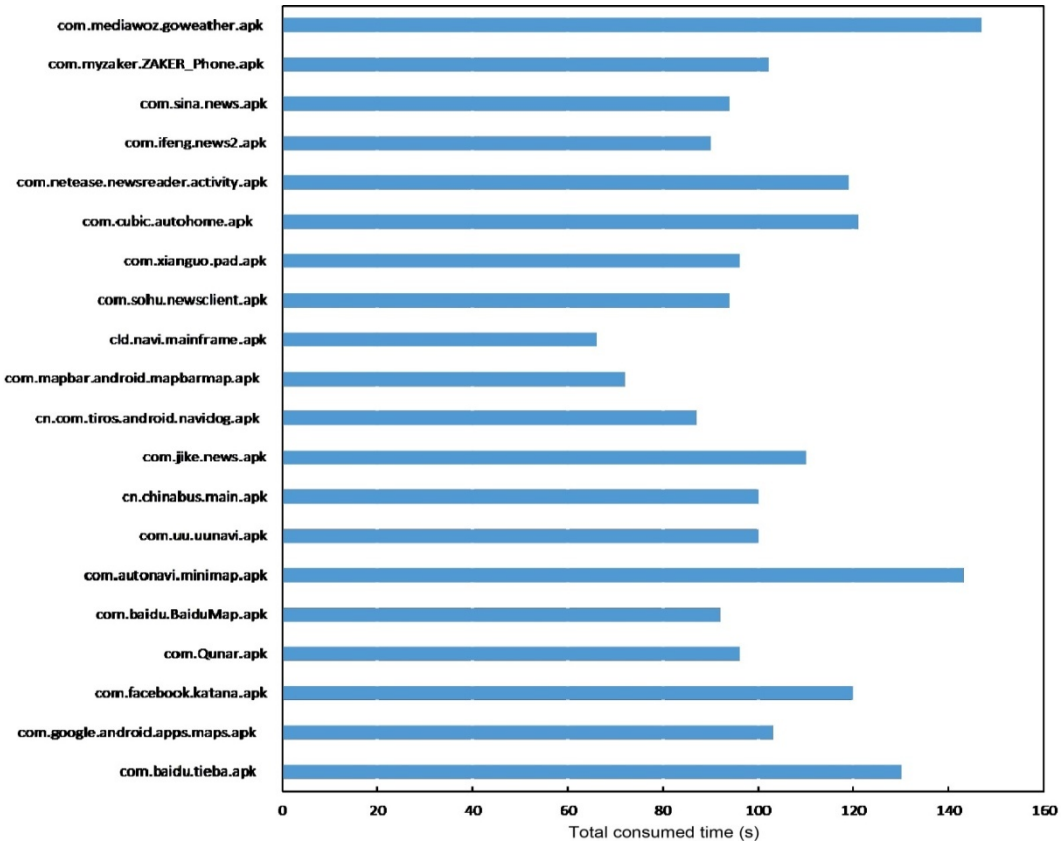

Fig.3 ASEF : The total consumed time of each app.

Also while analyzing apps we found out that there are 6 steps in ASEF which are preparing , starting log service, installing app ,testing app , ending process and analyzing the results. We also found that in this steps installation and testing process consumes 80% of total time so in order to make ASEF work faster we will have to reduce the time taken by these two steps

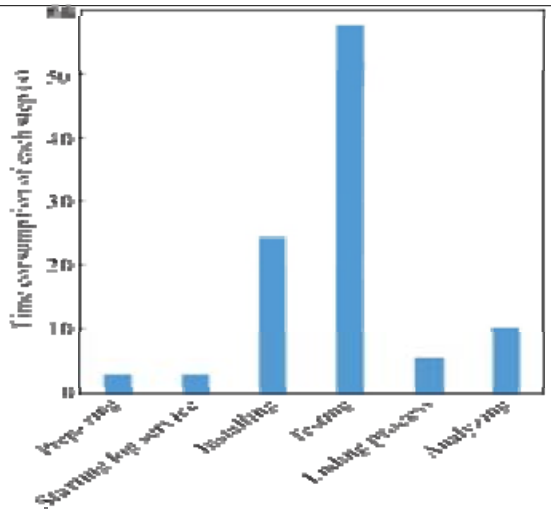

Figure 4 Step wise Time Consumption Of ASEF

Average number of apps installed in one Android device is about 30, it costs about 1 hour to use ASEF and SAAF to finish the analysis in one virtual machine and AVM. But if we can distribute the installed apps into separated individual VMs or AVMs, the whole time can be less than one minute, which is acceptable for user's experience in security check.

IX. Future Work

We can add user review analysis to this system in order to study the reviews given by the user and generate ratings for respective application. We can extend this system to a complete AppStore.

X. Conclusion

In this paper we propose a methodology to evaluate security of android mobile apps based on cloud computing platform. We also implement a prototype system that is AndroidArmour for automated forensic analysis of mobile apps using ASEF and SAAF. In this system,mobile app market will serve as the main line of defense against mobile malwares.

References

1. R. Lawler, Mary Meeker's 2013 Internet Trends report, http://techcrunch.com/2013/05/29/mary-meeker-2013internet-trends/, May 29, 2013.
2. J. Wu, On Top of Tides (Chinese Edition), Beijing: China PublishingHouseofElectronicsIndustry,January8,2011.
3. S. Q. Feng, Android software security and reversing engineering analysis(ChineseEdition),Beijing: Postsand Telecom Press, Feb. 2013.
4. Gartner, http://www.gartner.com/it/page.jsp?id=2153215, September 11, 2012.
5. List of mobile software distribution platforms, http://en.wikipedia.org/wiki/List of digital distribution platforms for mobile devices, July 19 2013.
6. D. Barrera, H. G. Kayacik, P. C. van Oorschot, and A. Somayaji, A methodology for empirical analysis of permission-based security models and its application to Android,inProc.17thACMConferenceonComputerand Communications Security, Chicago, USA, 2010, pp. 7384.
7. W. Enck, D. Octeau, P. McDaniel, and S. Chaudhuri, A study of android application security, in USENIX Security Symposium, San Francisco, USA, 2011.
8. A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, Android permissions demystified, in Proc. 18th ACM Conference on Computer and Communications Security, Chicago, USA, 2011, pp. 627-638.

