A Review Paper on: Malicious Application Detection in Android System

Anuja A. Kadam
Department of CSE, Government College of Engineering, Amravati, India,

Pushpanjali M. Chouragade
Professor, Department of CSE, Government College of Engineering, Amravati, India,

ABSTRACT
The electronic devices such as a mobile have become popular day by day, which is the major target of malicious applications(malapps). The detection and removal of malicious apps from android is the major issue in now days. So keep the malapps out of the app markets is an ongoing challenge. One major design points of Android security mechanism is control the permission that restricts the access of users having responsibility to the app developers with regard to accurately specifying the requested permissions and to the users with regarding to fully understanding the risk of granting certain combinations of permission. In this report, we have studied different techniques to determine themalapps in android. In permission induced risk first, analyse the risk of an individual permission and the risk of a group of corresponding permissions. Then used different feature ranking methods. Then use different methods to identify risky permission subsets. Secondly, then analyze the usefulness of risky permissions for malapps detection with subset selection. Thirdly, in depth analyze the detection results and determine the feasibility as well as the limitations of malapps detection based on permission requests.

Keywords:Android, malapps detection, permission control, Induced risk.

1. INTRODUCTION
Now day's electronic devices such as mobile become explosively popular for personal or business use. Android is the most popular mobile operating system today. To enhance the security, Android is designed to be a privilege separated operating system, in which each application runs with a distinct system identity. Android employs a quite efficient and convenient IPC mechanism .To facilitate resource accessing from isolated applications and data sharing among applications and the system, Android designs a permission-based security mechanism . Each application needs permissions to access system resources. These permissions are granted from users at install time. At runtime, each application is checked by Android before accessing sensitive resources. Any access to resources without granted permissions will be denied. Obviously, how to detect and keep the large number of malware out of the application is a challenging job. However, it imparts a significant responsibility to the app developers with regard to declaring the least-privileged set of permissions needed by designed apps, and to the app users with regard to fully understanding the risk of granting certain combinations of permissions.[8]

Popular malicious code detection methods including signature detection, behavior detection, virtual machine detection, heuristic detection, etc., have their drawbacks. Signature detection can only identify known malicious code; it does not work on unknown malicious code. Behaviors detection depends too much on program execution so its discrimination may be wrong if the execution conditions are not met. Virtual machine detection can generally detect encrypted malicious code but it needs the help of signature scanning and, furthermore, code with special instructions may escape virtual machine detection.[8] Heuristic detection can discriminate unknown malicious code, but it has complicated judgment logic, difficult development, and a high false positive ratio.[11]

It is feasible to identify malapps through analyzing the permission usage patterns, as intuitively an app's behavior is characterized by the permissions it requests. We thus see that exploring the permission-induced risk is beneficial to three parties, the Android app developers, the users, as well as the malappsdetectors [8]. To systematically detect malicious apps in existing Android Markets, we have three key design goals: Accuracy, Scalability, and Efficiency. Accuracy is a natural requirement to effectively detect malicious apps in current marketplaces with low false positives and negatives. Scalability and efficiency are challenging as we need to get the large number of apps that need to be scanned. Specifically, with our current collection of more than 300,000 apps, if it takes 6 seconds to examine a single app, a full scanning of the collection for known malware will require more than two weeks to complete.

2. SURVEY ON DIFFERENT TECHNIQUES
There are different method available for malicious application detection, which are being used are as follows:

2.1 The Droidranger Application
The study of droidranger shows that it uses the repository to detect the malicious application. The DroidRanger[11] leverages a crawler to collect Android apps from existing Android Markets and saves them into a local repository. The first detection engine is used to detect known malware. Specially, every known malware will be first pre-processed or distilled into a so-called permission-based behavioral footprint. Each footprint essentially contains necessary Android permissions requested by the malware and finally summarizes the wrongdoings. For each collected app, DroidRanger extracts fundamental properties associated with each app (e.g., the requested permissions and author information) and organizes them along with the app itself in a central database for efficient indexing and lookup. After that, in order to detect potentially malicious apps, we take an approach with two different detection engines. These footprints are necessarily the key to meet the scalability and efficiency requirements. The second heuristics-based

detection engine aims to uncover malware that has not been reported before.[10]

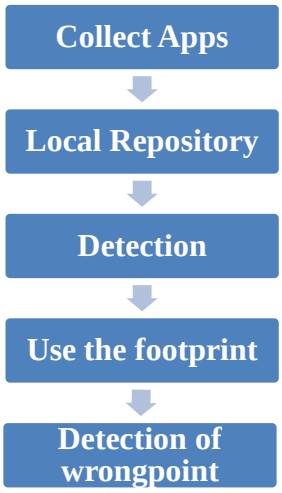

Figure 1: The overall architecture of DroidRanger

DroidRanger recognizes Suspicious Behaviors from all possible malicious applications and detects the Android features that may be misused.

2.2 Vetdroid Overview

The VetDroid System works as follows in which different layers are used.

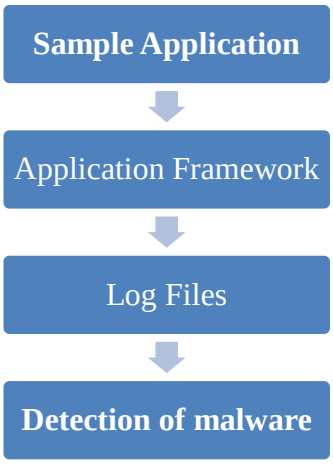

Figure 2: The VetDroidto detect malapps with permission use analysis.

In VetDroid[12] methods are used where Sample applications are first loaded into Application Framework, which automatically executes the Application in our Sandbox. During the Execution, Permission Use Analysis moduleidentifiesall the permission use points and their relations. These behaviors are recorded by Log Tracer with runtime information into a log file. The log file is offline processed by BehaviorProfiler to automatically construct behavior representations.[9]

2.3 Permission-Reduced Risks And The Detection Of Malicious Applications

In this method used for exploring permission-induced risk in Android applications first, we implement three feature ranking techniques to analyze the risk of granting each permission, based on which the permissions are ranked from most to least risky. Second, Permission sets, instead of individual permission, are analyzed by feature subset selection methods for investigation ofthe risk introduced by the collaboration of several permissions. Third, the detection of malapps based on risky permissions is calculated as a classification problem and executed by building classifiers.

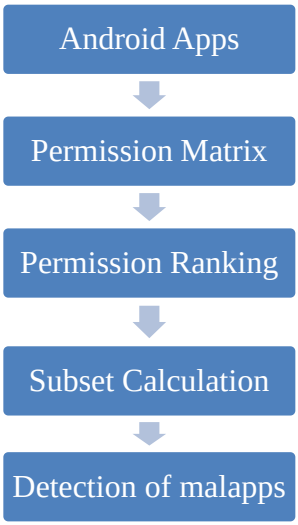

Figure. 3: The process of exploring permission-reduced risks and the detection of malicious applications in three levels.

Last, in order to explicitly characterize the risk caused by permission requests and use it to report malapps, detection rules are extracted from malapps detector.[8]

2.3.1 Range of data

The range of data is considered according to different analysis as begine apps and malapps. The begineapplications areless dangerous as compared to the malicious apps. As the more number of fraud are occurs at the malapps. The samples are ranges much more amount of users to the malapps as ranges to the different categories as VS-apps, Comm-1 Apps, Mal_Zhou apps and so on, whereas the begine apps contains the go ogles pay apps, which are ranges from 310926.

2.3.2 Techniques used

In this methodology for exploring permission-induced risk in Android apps. There are three feature ranking techniques to evaluate the risk of granting each permission. First based on which the permissions are arranged from most to least risky. Secondly, permission sets are evaluated by feature subset selection methods for investigating the risk introduced by the combination of several permissions. Third, the detection of malicious data based on risky permissions is formulated as a problem and executed by building classifiers. Lastly, the permission are ranked from different sets and then the risk can be determined by the detection rules are extracted from malapps detectors.[8]

3. CONCLUSIONS

In this paper, we provide a systematic study on the different techniques of malicious application detection in android mobiles. The exploration of permission-induced risk in Android apps on a large-scale in three levels. First upon rank all the individual permissions w.r.to their potential risk with different methods. Then, categorize subsets of risk permissions. Then using several algorithms detect the malapps based on the identified subsets of risky permissions.

4. REFERENCES

[1] W. Enck, D. Octeau, P. McDaniel, and S. Chaudhuri, "A study of Android application security," in Proc. 20th USENIX Secur.Symp., 2011.

[2] Yajin Zhou, Xuxian Jiang" Dissecting Android Malware: Characterization and Evolution" in 2012 IEEE Symposium on Security and Privacy© 2012.

[3] John P. Murphy, Vincent H. Berk, Ian Gregorio-de Souza"Decision Support Procedure in the Insider Threat Domain" in IEEE, 2012.

[4] H. Peng et al., "Using probabilistic generative models for ranking risks of Android apps, in Proc. ACM Conf. CCS, 2012.

[5] B. P. Sarma, N. Li, C. Gates, R. Potharaju, C. Nita-Rotaru, and I. Molloy, "Android permissions: A perspective combining risks and benefits," in Proc. 17th ACM SACMAT,2012.

[6] R. Pandita, X. Xiao, W. Yang, W. Enck, and T. Xie, "WHYPER: Towards automating risk assessment of mobile applications," in Proc.22nd USENIX Secur. Symp., 2013.

[7] JianlinXu, Yifan Yu, Zhen Chen_, Bin Cao, Wenyu Dong, Yu Guo, and Junwei Cao "MobSafe: Cloud Computing Based Forensic Analysis for Massive Mobile Applications Using Data Mining", inISSNll1007-0214ll10/10llpp418-427,Number 4, August 2013.

[8] Wei Wang, Xing Wang, DaweiFeng, Jiqiang Liu, Zhen Han, and Xiangliang Zhang "Exploring Permission-Induced Risk in Android " in IEEE transactions on information forensicsand security, no. 11, november 2014.

[9] Yuan Zhang, Min Yang, Zhemin Yang, GuofeiGu, PengNing and BinyuZang "Permission Use Analysis for Vetting Undesirable Behaviors in Android Apps" in IEEE transactions on information forensics and security, vol. 9,no. november 2014.

[10] Yajin Zhou Zhi Wang Wu Zhou Xuxian Jiang"Detecting Malicious Apps in Official and Alternative Android Markets"in Department of ComputerScienceNorth Carolina State University 2011.

5. AUTHOR BIOGRAPHY

Anuja A. Kadam has received her Diploma in Computer Engineering from Government Women's Polytechnic, Yavatmal, India, in 2010, the B.E. degree in Computer Science and Engineering from Babasaheb Naik College of engineering, Pusad, India in 2013. Her research interest includes Android, Data Mining, Web Mining. At present, she is persuingMaster of Technology in department of Computer Science and Engineering at Government College of Engineering, Amravati, India.

Pushpanjali M. Chouragade has received her Diploma in Computer Science and Engineering from Government Polytechnic, Amravati, India, in 2007, the B.Tech. degree in Computer Science and Engineering from Government College of Engineering, Amravati, India in 2010 and her M.Tech. in Computer Science and Engineering from Government College of Engineering, Amravati, India, in 2013. She was a Lecturer with Department of Computer Science & Engineering, in Government College of Engineering, Amravati, in 2010-11. Her research interest includes Data Mining, Web Mining, Image Processing. At present, she is an Assistant professor with department of Computer Science and Engineering at Government College of Engineering, Amravati, India, since 2011.