IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 12, Issue 3 (Jul. - Aug. 2013), PP 39-43
www.iosrjournals.org

Android Malware: Study and analysis of malware for privacy leak in ad-hoc network

Akash Malhotra[1], Pawan Prakash Singh[2]
Software Engineering Department / Suresh Gyan Vihar University, India
Computer Science Engineering Department/ Suresh Gyan Vihar University, India

Abstract : *Smartphone's users has been increasing since last few years which provides numerous operations like accessing information through online mode, payment options, using utility applications, playing games. Smartphone's have become so powerful these days that tends to play the role of PC's. The basic operation of any mobile phone calling, storing personal details like contact information in contact book, business data, text messages etc. Since we are in new generation, where so many different varieties of devices connect together with each other giving way for security concerns. With the huge and tremendous uprising Smartphone sales in market, the chances of malicious attacks became a trouble. As malware developers tries to steal information from such devices. This paper provides a study of analyzing malware through static and dynamic means. In static analysis we are performing reverse engineering to detect malicious code and in dynamic analysis we are using tools to identify the packet structure. Further we perform white listing for safe destination address. In this paper we are highlighting the aspects of mobile malware when compared with third party applications like Lookout etc.*

Keywords *-Android malware, reverse engineering, security and protection, Anti-virus, white listing*

Submitted date 19 June 2013 **Accepted Date: 24 June 2013**

I. INTRODUCTION

People have started using smart phones, since various companies provide various utility features in devices. According to Garter, Sales of Mobile devices grew 5.6 percent in Third Quarter of 2011 whereas smart phones sales increased 42 percent. With much interest, Android Operating System itself accounts for more than 55 percent of smart phones sales since its origin. Current day mobile devices have four capabilities –, computing, communication, sensing and high utility. Besides being at a high sale rate and such capabilities, these devices have also made the malicious attackers ready to attack and steal data. This idea is complemented by Lookout Threat report which has done great effort with respect to malware.

As the sale has increased exponentially the malicious coders have also increased. Mobile malware performs malicious activities like stealing private information, sending sms, reading contacts and can even harm by exploiting data. Malware authors can cause much damage to device users as so many users use capabilities of devices such as money transfer, online bank payment etc. Recent news and survey states that android platform is the mostly attacked platform for malwares. Since the malware can enter from the network, so its users responsibility to install malware free application. Even a malicious author can even repackage a famous application. So the first right is with user to check the permissions which an application asks during its installation. Once the user allows the application, he grants the application to use the permissions mentioned completely. Otherwise the user can deny installing the application.

Our aim is to reduce the risk of malicious applications causing harm to the user of a device, by enhancing the security which performs white listing of network permission by analyzing packet using tools such as snort, wire-shark etc. As we know Android platform has a permission model, which ask the user before installing the applications. Android Applications possess permissions when they want to perform operations that may result in cost or violate confidentiality and integrity of personal information present in the device. One of the very common permission is access to the Internet. Generally, 58% of the applications that are in the Android market request this permission, can communicate and access any host on the Internet.

Permission plays an important role in android device security. Before restricting access to an application component, you need to define the set of permission in the manifest. You have to use permission tag and within it you can specify the level of access the permission will allow (dangerous, signature, normal, signature Or System).

But if the malicious author repackages the applications with malicious code, then the victim may lose his precious data. Adding access level grants security up to some level. We were solving our problem by grouping the applications, according to permissions stated in it. Applications having risky permissions are

further analyzed with static and dynamic process and the destination address being blacklisted. We are using approach for looking for malware in application such as reverse engineering in android, and comparing by installing third-party application.

This could be seen in existing literature have used tools to find the over privilege permissions, user's behavior, attention while installing applications as this the basic step before user gets influenced to harmful application on their device. Our work will define the list of sites used in the internet by the application developer

II. Related Work

The attacks on mobile devices are keeping increasing, more and more malware are affecting the user. Testing using Kirin [5] with 311 popular applications revealed that the rules flagged 10 applications for which the behavior of five was found to be questionable. Kirin entirely checks the application author's permission requests and doesn't examine how the application uses these permissions.

David Barrera [2] and colleagues have per-formed permission checks on about 1,100 Android applications and have used self-organizing maps (SOMs) to visualize the relationship between the applications and the permissions requested. But, SOMs also focus solely on the application author's permission requests and don't examine how the application uses them. In a study of Android application permission requests that included 100 paid and 856 free applications, around 93 percent of the free and 82 percent of the paid applications had at least single dangerous permission request. Internet permission is the most common and dangerous request. But, completely analyzing the permission request isn't sufficient for mobile malware detection; this should be done in parallel with static or dynamic analysis. In year 2011, Felt [8] et al. analyzed 46 pieces of iOS, Android, and Symbian malware that have spread in the wild from 2009 to 2011.Andromaly [3] which monitors both the smart phone and user's behaviors by observing several parameters, spanning from sensors activities to CPU usage, 88 features are used to describe these behaviors; the features are then pre-processed by feature selection algorithms. The malware authors developed four malicious applications to check the ability to detect anomalies. This paper contains further research and illustrates latest malwares, detection and defense techniques by referring several papers, blog posts, vendor technology and specifications.

David [2] et al. discusses permission re-delegation attacks on Android. They introduce the problem and present an attack on a vulnerable deputy. We perform a larger analysis of applications and discuss how plat- forms need to change to prevent these attacks.

Chin et al. present ComDroid, a static analysis tool that aims to help prevent developers from accidentally making components public. They also make recommendations for changes to the Android platform to reduce the rate of unintentional deputies. Although their tool and their platform recommendations would help prevent some instances of permission re-delegation, attacks on intentional deputies would still remain.

Taint Droid [4] performs dynamic taint analysis. It tracks the real-time flow of sensitive data through applications to detect inappropriate sharing. The taint source is API data, and the network is the sink. They track only data flow, but not control flow. Taint Droid [4] is complementary to IPC Inspection because they track API return values but do not prevent API calls from being made. Another tool, Scan Droid [1], uses static analysis to determine data flow through Android applications; it is intended for use similar to Taint Droid [4]. Scan- Droid, however, requires access to application source code. Kirin [5] checks application permission requirements and recommends against the installation of applications with certain permission combinations. Their rules are intended to help detect malware.

Apex [6] offer extensions to the Android framework to provide fine grained control over an app's access to potentially sensitive resources. Most of these efforts are aimed at addressing this problem on the user's phone; Risk Ranker [7], on the other hand, attempts to identify such risky behaviors at the app market, offer extensions to the Android IPC model that allow the ultimate implementer of a privileged feature to check the IPC call chain to ensure unprivileged apps cannot launch confused-deputy attacks unnoticed. Similarly, besides the above defenses, some work has been proposed to apply common security techniques from the desktop to Mobile devices. Some work has focused on malware and the overall market health. Felt et al. [8] surveyed 46 malware samples on three different smart phone platforms, discussing the incentives that motivated their creation and possible defenses against them. But, this work did not discuss how to discover this malware. Our work has a much stronger emphasis on malware detection than privacy leak detection. Mal Genome [9] aims to systematically characterize existing Android malware from various ways, including installing methods, activation mechanisms as well as the nature of carried malicious risks posed by existing in-app ad libraries. These are the various works.

III. METHODOLOGY

Detection:

Static analysis

This approach is generally used when we looking for malicious code inside the suspected application. The malware application can cause a serious harm to the user by exploiting his precious data or by stealing it. In static analysis approach, genreally de-compilation of installer file is done. An .apk file of android is analyzed with reverse engineering process. Since we are analyzing through code based approach, so it is termed as static analysis approach. We use various tools for analyzing it. Tools used are Winzip, Java Decompiler, dex2jar etc. Reverse engineering in android is done to open and view the java and xml files for any malicious code. Following the few steps, we get view of all the activities of android. We can even view the xml files of android application installer, this process exactly pin points the false code. With the following steps we get complete knowledge of a developer's intention.

Using revere engineering:

Step 1: Rename the doubtful Malware Android installer .apk to .apk.zip

Step2: Extract to new folder for example says New_folder

Step3: Using the tool Dex2jar, convert classes.dex to classes_dex2jar.jar

Step4: Now use the tool Java Decompiler to view the classes.

Step 5: You can look for the malicious code.

This is repackaging of a malware being done.

Dynamic analysis: Dynamic analysis involves running the mobile application in an environment, such as a emulator or vm(virtual machine), so that researchers can check the dynamic behavior of android application. We are first performing static analysis of malware, and the further checking the malware for by checking its behavior. Our research study is done by using the both approach that is static and dynamic. In dynamic we are running the malware on android emulator and checking through snort tool for outgoing packets. Basically we are analyzing packets, for destination address. If the destination address is not in white list, the packet will be dropped, so any other device running in that environment is also safe as we are breaking the connection. With the help of snort tool we are generating logs for incoming and outgoing packets. These logs are further analyzed with tools such as wire shark. We can trace the packets. Testing using Taint Droid with 30 popular third-party Android applications revealed that 15 of them share user location with advertisers and seven share phone identifiers with remote servers without the user's knowledge. Although the researchers used Android Monkey (ADB Monkey) to generate inputs, this is not as efficient as testing with real users. Further, this approach hasn't been tested against malware that exhibits polymorphic behavior or code fragment encryption. Snort tool catches the incoming and outgoing packets. We are going to catch the packets and analyzes them.We are going to track the malicious code behavior at runtime by installing the suspicious malware, (specifically the one which steals information's) on android emulator. Once the malware makes HTTP Post or get request we will track it running snort on machine and trace the destination address where it is going to send our precious data, so that any other Device coming in that network, will get a warning that this URL does not fall under the white listing URL's.

Using Snort tool:

Step 1: Start snort tool

Step2: Install android malware on Android emulator

Step3: Run the command and log file will be generated.

Step3: Trace the packet from where the malware is making connection

Step 4: If it is unsafe, blacklist it or drop the packet.

Step 5: Protect other devices with alert message that this cannot make connection.

White listing and blacklisting

White listing is a list or register of entities that are being used for particular reason, and have a particular advantage, service, mobility or recognition. Only stored entities on the list will be accepted, approved, and/or recognized. White listing is the converse of blacklisting, the practice of finding entities that are denied, not recognized. In our approach we are maintain list of URL's to be blacklisted and white listed, which protects the other device installing the malicious application. In this we are adding safe URL's

A blacklist (or black list) is a list or register of entities which particularly, being denied a particular advantage, service, mobility, access or rights. Simply blacklist can mean to deny someone work in a particular field, or to stop a person from social circle. We are maintaining it under snort rules.

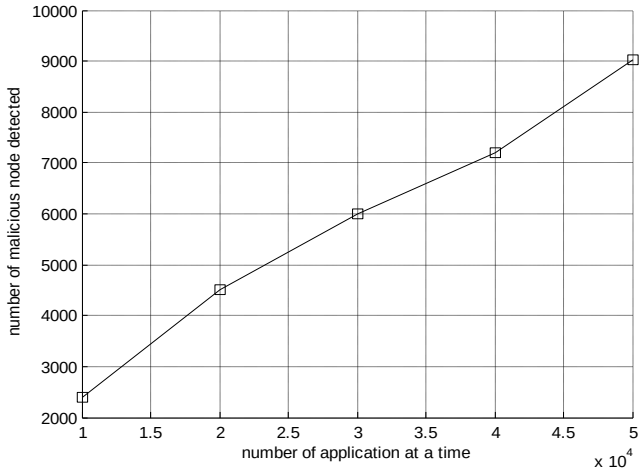
IV. COMPARISON STUDY OF THIRD PARTY APPS AND OUR METHODOLOGY

We compare our malware detection techniques between Lookout, third party application and our approach on following data[6] and found that our analysis is also effective as we can see it in graph. This graph is standard graph when tested on lookout application for 10,000 and above data.

Dataset 1:
Large volume of data, Standard graph-LOOKOUT

Table1

s/n	Total number of dataset	Total number of malicious node(standard)
1	10000	2400
2	20000	4502
3	30000	6001
4	40000	7202
5	50000	9020



Dataset 2:

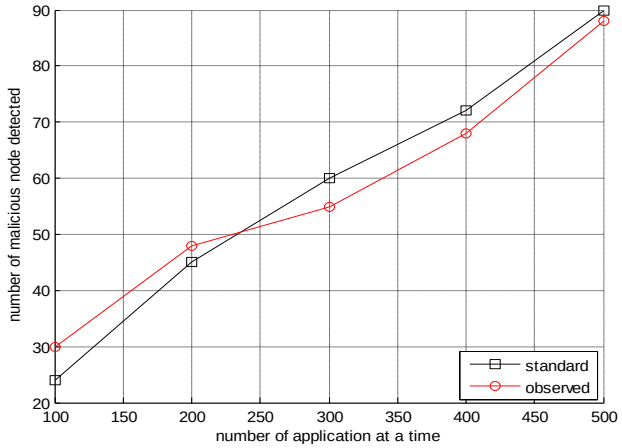
This graph is comparison Lookout and our approach here dataset used is small in number, so we can observe, our approach is also effective in finding the malware specially which uses internet permission.

Observed graph with respect to standard graph (small volume of data).

Table2

Total number of dataset	Total number of malicious dataset (standard)	Total number of malicious dataset (observed)
100	24	30
200	45	48
300	60	55
400	72	68

500	**90**	**88**



V. RESULT & CONCLUSION

We can calculate a result by using true and false positive ratio and then by finding out total accuracy
(1)TPR=TP /TP+FN
(2)FPR=FP /FP+TN
(3)Total Accuracy=TP+TN /TP+TN+FP+FN
In our paper we have presented our methodology to understand how we are able to analyze malwares through reverse engineering(code based) and packet analysis(behavior based).We have used various tools to analyse them and compared the dataset with Lookout ,Third Party Application. Our approach is quite efficient as it is providing white listing of safe URL's.

REFERENCES

[1]. Adam, P. F ,Chaudhuri, A., & Foster, J. S. (2009). ScanDroid: Automated security certification of android applications. In IEEE symposium of security and privacy.
[2]. D. Barrera, H. G. Kayacik, P. C. van Oorschot, and Somayaji. A methodology for empirical analysis of permission-based security models and its application to android. In the Proceedings of the 17th ACM conference on Computer and communications security, CCS '10, pages 73–84, NY, USA, 2010. ACM
[3]. A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, Y. Weiss: Andromaly: a behavioral malware detection framework for android devices. Journal of Intelligent Information Systems 38(1) (January 2011) 161.
[4]. Enck, W., Gilbert, P., Chun, B.-g., Cox, L. P.,Jung, J., McDaniel, P., and Sheth, A. N.TaintDroid: An Information-Flow Tracking System for Real-time Privacy Monitoring on Smartphone's. In the Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (2010),USENIX OSDI '10
[5]. Contagio mobile malware mini dump. https://contagiominidump.blogspot.com/
[6]. M. Nauman, S. Khan, and X. Zhang. Apex: Extending Android Permission Model and Enforcement with User Defined Runtime Constraints. In the Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, ASIACCS '10, 2010
[7]. Risk Ranker https://www.csc.ncsu.edu/faculty/jiang/pubs/MOBISYS12.pdf
[8]. Felt, A. P., Chin, E., Hanna, S., Song, D., and Wagner. Android Permissions Demystified. In the Proceedings of the 18th ACM Conference on Computer and Communications Security (2011), CCS 11.
[9]. MalGenome: Proceedings of the 33rd IEEE Symposium on Security and Privacy San Francisco, CA, May 2012
[10]. Felt, A. P., Wang, H. J., Moshchuk, A., Hanna, S., and Chin, E. Permission Re-Delegation: Attacks and Defenses. In the Proceedings of the 20th USENIX Security Symposium (2011), USENIX Security '11.