Android Security: A Review

Omar M. Ahmed [1] and Amira B. Sallow[2]

[1]Department of Computer Science, Faculty of Science, Zakho University, Duhok, Kurdistan Region – Iraq
[2]Department of Computer Science and Information Technology, College of Computer Science & Information Technology, Nawroz University, Duhok, Iraq

ABSTRACT

Smartphones are used by billions of people that means the applications of the smartphone is increasing, it is out of control for applications marketplaces to completely validate if an application is malicious or legitimate. Therefore, it is up to users to choose for themselves whether an application is safe to use or not. It is important to say that there are differences between mobile devices and PC machines in resource management mechanism, the security solutions for computer malware are not compatible with mobile devices. Consequently, the anti-malware organizations and academic researchers have produced and proposed many security methods and mechanisms in order to recognize and classify the security threat of the Android operating system. By means of the proposed methods are different from one to another, they can be arranged into various classifications. In this review paper, the present Android security threats is discussed and present security proposed solutions and attempt to classify the proposed solutions and evaluate them.

KEYWORDS: Android, Security, Smartphone.

I. INTRODUCTION

Android is a platform for mobile that is designed and developed to be totally open source. An advanced level of software and hardware can be used by Android applications, in addition to local and server data and information, uncovered by the operating system in order to gets value and innovation to consumers. On the way to guarantee the security of user's application, information, and data Android platform should be having a powerful security mechanism. In order to provide security for Open source platform, it requires a powerful and severe security architecture. With multilayered security, the architecture of Android platform is designed that gives the flexibility required for an open source platform (Android, 2017).

Android applications are increasing very fast, the informal application developers, and the open source platform encourages the malicious software developers to Exploit of such application and vulnerable platform and steal the user's private data to unintentionally damage the developer reputation and the apps markets. Furthermore, because the Android platform is considered one of the access control of the device, consume excessive battery, steal private data, and even transform the smartphone from normal into a botnet zombie (Appbrain, 2017).

There are huge diversity of Android vulnerabilities are existing which they could appear in many layers of Android operating system's security, like framework layer or application layer or even Linux Kernel layer. Vulnerabilities as well occurs in benign or any applications by the unintended include of design flaws or coding faults. As mentioned previously, the flawed Android operating system gives the opportunity for hackers or attackers to use it. There is many of security threats exists on Android phones, like Denial of Service attacks, colluding, repackaging apps to inject malicious code, permission escalation, and unauthorized access between the application.

The objective of this paper is for expanding the coverage of malicious application growth, and Android security threats. In this review paper, the main proposed works in Android operating system privacy and security will be covered, and furthermost of the existing deployed tools and techniques.

1. Android Platform Security Architecture

Android is developed by Google at first and then via Open Handset Alliance (OHA) is promoted. The Android platform is located on highest of the Linux kernel which is

Academic Journal of Nawroz University (AJNU)
Volume 6, No 3(2017), 6 pages
Received 1 May 2017; Accepted 29 August 2017
Regular research paper: Published 30 August 2017
Corresponding author's e-mail: omar.m.a@gmail.com

consists of the APIs, libraries, and middleware by writing in c programming language, and above the application framework the application software is running which contains other libraries which they are Java-compatible. Like open source licenses by Google, the Android's source code is published. By means of a stack of software components, the Android operating system is based, which is organized by five major layers like appeared in Figure 1. Android operating system and platform components and layers are described and analyzed as Threats and Defenses of Android Security (Brähler, 2010).

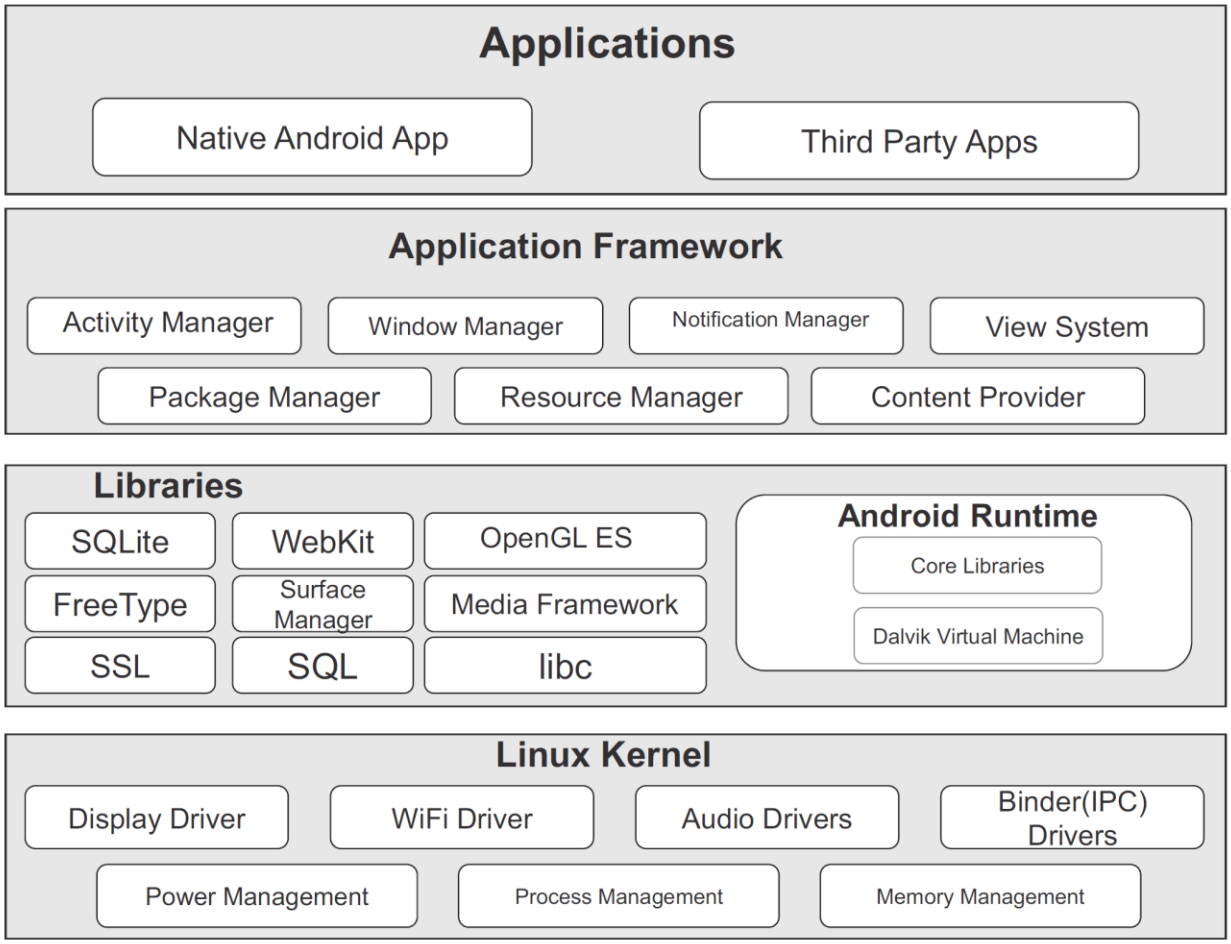

Fig 1: Android operating system architecture

The Linux kernel is the most important among all layers of the whole system. This layer for the embedded environment is customized especially containing limited resources. The entire Android operating system is constructed on top of the Linux kernel and by Google extra architectural changes has been made. This layer as well performances as a deliberation layer between the software and hardware layers. The basic system functionality is provided Linux kernel such as the management of device, memory, and process. Beside, an array of device drivers is provided by Linux kernel which makes the job simpler while communicating the peripheral devices with Android.

The native libraries of the Android on top of the Linux kernel were developed. For handling different types of data on the device this layer is enabled. Different useful libraries are provided for Android OS as the well-function. These libraries are developed for a particular hardware and were written in C or C++ language. As an example of native libraries such as for Internet security the SSL libraries are used, for providing various media codecs the Media framework is used, to show 2D or 3D graphics content the OpenGL is used, the common library libc, for storing data the SQLite database engine is used, and to display HTML content the Web browser engine WebKit is used.

The Android runtime has an element named Dalvik Virtual Machine (DVM), which is an interpreter for byte code that has been converted from Java Virtual Machine (JVM) particularly designed and enhanced for Android. From Linux core features the Dalvik takes advantage such as memory management, multitasking execution environment, and multi-threading, which is for Java language it is very essential. Dalvik provides control to applications to work as a process immediately with its own VM and the Linux kernel. While Dalvik is utilizing JVM, which gives clients a group of APIs and libraries to design and create Android application mainly via Java

programming language.

The application framework is written in Java and it is a collection of services that together make an environment for managing and running Android applications. As Java classes, the services are provided to applications. These services are allowed for application developers to use in their applications. The main services of application framework are View System, Notifications Manager, Resource Manager, Content Providers, and Activity Manager.

Application layer which is the last layer is placed on the upper of the Android operating system stack. This layer involve both of the preinstalled application and third-party application developed and created by informal application developers, such as Contacts Manager, Browser, and Email apps.

2. Android Security Threats

Android operating system security is designed as a permission-based mechanism which manages and control the admission and approval of third-party Android apps to reach critical resources. This permission-based mechanism is extensively criticized for the inefficient permission management and controlling the application permissions, by end-users, marketers, and developers. Let's say, all permission requests from an app can either be accepted by users to install it or not. Here, the major security threats of the Android will be discussed, which makes the user's information vulnerable to leak and places the privacy at risk (Faruki et al., 2015).

2.1. Information Leakage

The design of the present Android architecture restricts applications from using or accessing other applications or resources except it is approved by the users. Before installing and using an application the user has to grant all access requests of the resources. Information leakage happens when without any restriction from OS the users grant resources. However, permission control mechanism of the Android Operating system ineffectively protect user's resource and privacy from malware.

2.2. Privilege Escalation

Privilege escalation threats are utilized by taking advantage of freely accessible Android kernel vulnerabilities to obtain higher or all access to resources which is typically protected from a user or app. Such kind of threat can be a consequence of unauthorized activities from apps with extra privileges than intended, as a result, can lead to many sensitive information leakages. To obtain access to the critical permissions the Android exported components can be exploited.

2.3. Repackaging Apps

In the Android operating system, one of the most common and important security threat is Repackaging. Repackaging by using reverse-engineering techniques can apply decompiling/disassembling of .apk files and inserting malicious code inside the main source code. By using of repackaging techniques as a normal app with distract the malicious code can be disguised. The differentiate between a normal app and a repackaged malicious code can be difficult because as the same way as the legitimate one the repackaged app typically appears to function.

2.4. Denial of Service Attack

The growing quantity of mobile devices which they are connected to the Internet as a big network which could be a stage for evolution of DoS attacks. Because the smartphones are not fortified or having less protections compared with PCs, the developers of malicious applications find it as an appropriate platform for DoS attacks. The primarily objectives of DoS attacks are targeting Overusing limited CPU, battery power, memory, and network bandwidth.

2.5. Colluding

The colluding threat is happening from the side of the users. Here, users deploy a group of applications that have the same certificate and grant various sorts of permissions could be non-sensitive or sensitive. These applications after installing it can get access to all their resources and permissions by taking advantage of a shared UID.

3. Proposed Solutions

Android security solutions separated into two kinds: 1) Static; 2) Dynamic which both can use for vulnerability assessment, analysis, and detection. Static methods are fast, yet it needs to manage false-positives sensibly. Dynamic methods, however time-consuming, are exceptionally useful when applications are extremely obscured. There are also hybrid methods that merge both dynamic and static methods together with the limitations of both.

3.1. Crowdroid

In (Burguera, Zurutuza, & Nadjm-Tehrani, 2011) they proposed a Crowrdoid system which is a behavior based malware detection. Which is having two mechanisms, a crowd sourcing application that should be deployed on user-devices and other is for malware detection which is a remote-server. The crowd sourcing application communicates with the remote server by sending the behavioral data and this data could be as an application log file. The application log file contains fundamental device info, behavioral data, and a list of installed apps. This data is processed at the remote-server to produce feature vectors which later might be analyzed by 2-means partition clustering for recognizing if the app malicious or benign. An application report is created and deposited in the remote-server's database.

3.2. AndroSimilar

In (Faruki, Ganmoor, Laxmi, Gaur, & Bharmal, 2013) they propose AndroSimilar method which is used

statistically implausible feature selection by means of similarity digest hashing mechanism. The proposed solution efficiently detects code obfuscated malware, control flow obfuscation, method renaming, and junk method addition. Signature produced through the proposed method is sufficiently solid to detect unknown samples obfuscated with several code obfuscation techniques which they are not detected by the common Antivirus application. Manually analyzing the suspected samples detected by the method in which through present Android malware signatures is verifying their similarity. This proposed signature method is strong in counter to repackaged apps.

3.3. Kirin

In (Bahman Rashidi, Fung, & Vu, 2014) they proposed Kirin method the main goal of this method is to qualifying malicious applications at install time utilizing certification process on applications. On app's requested permissions this method uses a set of previously defined security rules for finding the matched permission requests that are considered as a malicious. The defined rules are based on the sensitive permissions which lead to abusing of dangerous activities and permissions.

3.4. RecDroid

In (B Rashidi, Fung, & Vu, 2014) they proposed the RecDroid method it is a framework through crowdsourcing for resource accessing permission control. RecDroid tries to help users to choose the correct choice to decide if a permission request ought to be denied or accepted. RecDroid is an approval framework that brings together the user's permission responses to the application's permission requests, also for evaluating the level of experience of users this method is utilized and to figure a proper response to the permission request even to be rejected or accepted. RecDroid depends on a slight group of seed expert users that might generate dependable recommendations for a slight group of apps. To ignore malicious responses and detecting the malicious clients this method in addition utilizes a game-theoretic Bayesian.

3.5. Aurasium

In (Xu, Saïdi, & Anderson, 2012) they proposed Aurasium technique which is a very useful technique that takes control of execution of applications, by forcing arbitrary security rules at runtime. In order to have the ability to do that, Aurasium includes code for rules enforcement to Android applications by repackages it. Aurasium can apply security rules across multiple applications not only at individual application. Any privacy and security violations are informed to the user. So, it rejects the necessity of manipulating an Android operating system to monitor application behavior. It interferes when the application accessing sensitive data like phone identifiers, messages, contacts and implementing shell-commands by requesting the user for approval regarding the same.

3.6. FireDroid

In (Russello, Jimenez, Naderi, & van der Mark, 2013) they propose FireDroid which is a framework that is policy-based by interleaving process system calls can implementing security policies. In this method an application monitor is generated for tracking all processes spawned in Android operating system and based on humanly managed policies can accept or reject them. At runtime, the FireDroid can identify if an app is executing illegitimate or potentially harmful actions by interrupting the system calls the app executes. Regardless of if the malware is repackaged or a new type of an existing one: FireDroid can detect and implement the proper security policies once the malware executes harmful system calls.The objective of FireDroid is that it is entirely transparent to the apps along with Android operating system. By this the users are not complicated with a heavy interaction.

3.7. DroidScope

In (Yan & Yin, 2012) they proposed DroidScope which is a Virtual Machine Introspection (VMI) that is for Android applications is dynamic analysis framework. Contrasting other dynamic analysis frameworks, it doesn't be located inside the emulator but creating Dalvik-level and OS-level semantics by locating outside the emulator. By this, even in the kernel, the privilege escalation attacks can be detected. It likewise makes the attackers mission of disorderly analysis hard. upon QEMU emulator the DroidScope is built, and likewise, make a set of APIs available to modify analysis needs to human analysts.

3.8. RiskMon

In (Jing, Ahn, Zhao, & Hu, 2014) they propose the RiskMon method which is integrated runtime behaviors and user's expectations of authorized apps to produce a risk assessment baseline in which can seizures suitable behaviors of apps. User's expectations on the apps are the key part of the operating system. Initially, the user's expectations of the deployed applications are gathered and the position of permission sets based on their relevancy to the corresponding app. After that, depending on the gathered data from the user, it generates or produce the risk assessment baseline for the apps. Lastly, utilizing the produced baseline, the method ranks deployed apps depending on the risk of the application's interactions.

3.9. RiskRanker

In (Grace, Zhou, Zhang, Zou, & Jiang, 2012) they propose RiskRanker is a practical active procedure to recognize zero-day Android malicious apps. It attempts to assess potential security risks caused by unauthorized applications. They design an automated system with the purpose of analyzing the harmful behavior of applications dynamically. The proposed method

achieves a two-stage risk analysis. Firstly, they recognizes applications with medium and high risk. The purpose of recognizing these applications it traces non-obfuscated implementations of applications that invoke: privacy violation attacks, illegal cost creation, and launching root exploits. Secondly, so as to determine those applications that encrypt exploit code to avoid the previous stage analysis it achieves an additional investigation by analyzing suspicious application behavior. To solve this encounter, they make a set of heuristics to map applications to associated risk types (Low, Medium, and High risk).

3.10. DroidRanger

In (Zhou, Zhou, Jiang, & Ning, 2012) they present a survey to estimate the security of applications on some existing unofficial Android application markets and Google Play. The DroidRanger has two-stage analysis to detect zero-day malware and existing known malware. They utilize a permission-based behavioral footprinting scheme so as to detect known malware. The other stage, to identify certain inherent behaviors of a zero-day malware they apply a heuristics-based filtering scheme. The summary of all previous proposed solutions is presented in Table.1.

Table.1. Proposed solutions

Proposed Solutions	Objective			Mechanisms	
	Assessment	Analysis	Detection	Static	Dynamic
Crowrdoid	□		□		□
AndroSimilar			□	□	
Kirin	□			□	
RecDroid	□				□
Aurasium			□		□
FireDroid			□		□
DroidScope		□			□
RiskMon	□	□			□
RiskRanker		□		□	
DroidRanger		□	□	□	□

4. Conclusion

Alongside the expanding of Android smartphones, the quantity of Android applications, as well as malware is expanding daily. Regardless of existing Android security system, malware exploits the existing security system's weakness to reach and access the granted resources. By this means, various solutions have been proposed in order to control and prevent the vulnerabilities in Android platform. In this review paper, the proposed solution is divided into two groups static and dynamic and into three goals assessment, analysis, and detection.

REFERENCES

Android. (2017). Android Security Overview. Retrieved from https://source.android.com/security/

Appbrain. (2017). Number of android applications. Retrieved July 28, 2017, from http://www.appbrain.com/stats/number-of-android-apps

Brähler, S. (2010). Analysis of the Android Architecture. *Karlsruhe Institute for Technology*, 52. Retrieved from http://os.ibds.kit.edu/downloads/sa_2010_braehler-stefan_android-architecture.pdf

Burguera, I., Zurutuza, U., & Nadjm-Tehrani, S. (2011). Crowdroid: Behavior-Based Malware Detection System for Android. *Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices - SPSM '11*, 15. https://doi.org/10.1145/2046614.2046619

Faruki, P., Bharmal, A., Laxmi, V., Ganmoor, V., Gaur, M. S., Conti, M., & Rajarajan, M. (2015). Android security: A survey of issues, malware penetration, and defenses. *IEEE Communications Surveys and Tutorials*, *17*(2), 998–1022. https://doi.org/10.1109/COMST.2014.2386139

Faruki, P., Ganmoor, V., Laxmi, V., Gaur, M. S., & Bharmal, A. (2013). AndroSimilar : Robust Statistical Feature Signature for Android Malware Detection. *Proceedings of the 6th International Conference on Security of Information and Networks*, (September 2015), 152–159. https://doi.org/10.1145/2523514.2523539

Grace, M., Zhou, Y., Zhang, Q., Zou, S., & Jiang, X. (2012). RiskRanker: Scalable and Accurate Zero-day Android Malware Detection. *10th International Conference on Mobile Systems, Applications, and Services*, 281–294. https://doi.org/10.1145/2307636.2307663

Jing, Y., Ahn, G.-J., Zhao, Z., & Hu, H. (2014). RiskMon : Continuous and Automated Risk Assessment of Mobile Applications. *Proceedings of the 4th ACM Conference on Data and Application Security and Privacy - CODASPY '14*, 99–110. https://doi.org/10.1145/2557547.2557549

Rashidi, B., Fung, C., & Vu, T. (2014). On lightweight mobile phone application certification. *Proceedings of the ACM MobiCom Workshop on Security and Privacy in Mobile Environments*, 235–245.

https://doi.org/10.1145/1653662.1653691

Rashidi, B., Fung, C., & Vu, T. (2014). RecDroid: A resource access permission control portal and recommendation service for smartphone users. *2014 ACM MobiCom Workshop on Security and Privacy in Mobile Environments, SPME 2014*, 13–17. https://doi.org/10.1145/2646584.2646586

Russello, G., Jimenez, A. B., Naderi, H., & van der Mark, W. (2013). FireDroid: hardening security in almost-stock Android. *Proceedings of the 29th Annual Computer Security Applications Conference*, 319–328. https://doi.org/10.1145/2523649.2523678

Xu, R., Saïdi, H., & Anderson, R. (2012). Aurasium: Practical Policy Enforcement for Android Applications. *Proceedings of the 21st USENIX Conference*, 27. Retrieved from https://www.usenix.org/system/files/conference/usenixsecurity12/sec12-final60.pdf%5Cnhttp://dl.acm.org/citation.cfm?id=2362793.2362820

Yan, L. K., & Yin, H. (2012). DroidScope: Seamlessly Reconstructing the OS and Dalvik Semantic Views for Dynamic Android Malware Analysis. *In USENIX Security Symposium*. https://doi.org/10.1145/2420950.2420980

Zhou, W., Zhou, Y., Jiang, X., & Ning, P. (2012). Detecting repackaged smartphone applications in third-party android marketplaces. *Proceedings of the Second ACM Conference on Data and Application Security and Privacy - CODASKY '12*, 317–326.

https://doi.org/10.1145/2133601.2133640