International Journal of Engineering Research and Technology. ISSN 0974-3154 Volume 10, Number 1 (2017)

Review of Inter-App Permission Leakage and Malware Characterization in Android Operating System

Nitesh A. Patil
Department of Computer Science and Engineering,
D. Y. Patil College of Engineering,
Kolhapur, Maharashtra, India
Dr. K. V. Kulhalli
Professor, Department of Information Technology,
D. Y. Patil College of Engineering,
Kolhapur, Maharashtra, India.
Dr. S. Seema
Professor, Dept. of Computer Science and Engineering,
M S Ramaiah Institute of Technology, Bangalore,
Karnataka, India

Abstract
Android operating system is extremely popular because of its openness to developers as well as freely availability of numerous apps. It also supports third party developed apps which might be developed intentionallyto grab private information of the user. Currently many scenarios have proved that because of inter app messagingmechanisms in android operating system; the user privacy is at risk.

Android app markets like Google Play Store are at the hit of malware attacks and it seriously threatening users' security. This paper presents a survey on inter apppermission leakage in android operating system. Furthermore android apppermission categories and malware characterization in androidalso highlighted.

Keywords: Android Security, Android Permissions, Inter-app permission, Android Malware Characterization, Malware Detection.

1. Introduction

All Today the boom of the android phones, the users are using more and more applications in almost all sectors suchas health, entertainment, office, college, banking etc. Android device activations are hitting near about 1.5 million perday and unfortunately, privacy leakage issues in android devices are also increasing in same flow. In app store, there aremany applications those are free but they are relying on advertisement for their income. These applications can get auser's private information easily and use it for target advertisements. Users also accept this business model, but they areunaware about his private information is being leaked by certain applications without his permission.

Android system provides sharing of data and services between apps using inter-app communication system. Androidpermission system controls access of resources of the mobile device. Hence permissions can be misused intentionally soenforcing permissions is not enough to prevent from permission violations. Android's enforcement of the permissions isat the level of individual apps, allowing multiple malicious apps to collude and combine their permissions or to trickvulnerable apps to perform actions on their behalf that are beyond their individual privileges [1].

Application components are basic logical building blocks of Android apps. Each component can run individually,either by its embodying application or by system upon permitted requests from other applications. Android apps havefour types of components: (1) Activity components provide the basis of the Android user interface. Each Applicationmay have multiple Activities representing different screens of the application to the user. (2) Service components providebackground processing capabilities, and do not provide any user interface. Playing music and downloading a file while auser interacts with another application are examples of operations that may run as a Service. (3) Broadcast Receivercomponents respond asynchronously to system-wide message broadcasts. A receiver component typically acts as agateway to other components, and passes on messages to Activities or Services to handle them. (4) Content Providercomponents provide database capabilities to other components. Such databases can be used for both intra-app datapersistence as well as sharing data across applications [1].

Privacy violations can occur even when a user grants access to protected data (e.g. contact list, exact location, etc.) to abenign app, i.e. one not trying to violate user's privacy. This holds true, since the app may either be used as a confuseddeputy [12, 13], i.e. accidentally allowing other malicious apps to use its functionality to access the resources, or bebundled with a malicious advertisement library [14], which misuses the shared permissions to violate user privacy. Also,benign Android apps tend to request more permissions than needed for their intended functionality [11].

2. Literature Survey

Hamid Bagheri, AlirezaSadeghi, et. al[1] presents novel approach, called COVERT, for compositional analysis ofAndroid inter-app permission leakage vulnerabilities. COVERT's analysis is modular to enable incremental analysis ofapplications as they are installed, updated, and removed. It statically analyzes the reverse engineered source code of eachindividual app and extracts relevant security specifications in a format suitable for formal verification.

AlexiosMylonas, MarianthiTheoharidou, and DimitrisGritzalis [2] provides taxonomy of user data found on asmartphone, their respective Android permissions and discussed ways to disclose their data. They have identified privacythreats applicable to user data, crawled apps from Google Play and used this sample to list descriptive statistics forpermission combinations that may violate user privacy.

Li Li, AlexandreBartel, et. al, [3]. In this paper, authors have proposed IccTA, a static taint analyzer to detect privacyleaks among components in Android applications. IccTA goes beyond state-of-the-art approaches by supporting inter-componentdetection. By propagating context information among components, IccTA improves the precision of theanalysis. IccTA outperforms existing tools on two benchmarks for ICC-leak detectors: DroidBench and ICC-Bench.

Li Li, AlexandreBartel, et. al [4], In this paper, author presented potential component leaks (PCLeaks), a tool toexploit potential component leaks and PCLeaksValidator, a tool which automatically generates a correspond maliciousapps to validate the results of PCLeaks. Concretely, PCLeaks first builds a precise control-flow graph for the analyzedapps. Then, it performs static taint analysis with a well-defined set of source and sink methods to identify potential activecomponent leaks and also potential passive component leaks.

Drago S, Michael G. Burke, Salvatore Guarnieri, [5], Author has identifed three types of inter-application Intentbasedattacks that rely on information flows in applications to obtain unauthorized access to permission-protectedinformation. Two of these attacks are of previously known types: confused deputy and permission collusion attacks. Thethird attack, private activity invocation, is new and relies on the existence of dificult-to-detect misconfigurationsintroduced because Intents can be used for both intra-application and inter-application communication. Suchmisconfigured applications allow protected information meant for intra-application communication to leak intounauthorized applications. This breaks a fundamental security guarantee of permissions systems: that application can onlyaccess information if they own the corresponding permission.

Yajin Zhou, Xuxian Jiang [6] focuses on the Android platform and aim to systematize or characterize existingAndroid malware. Particularly, with more than one year effort, they have managed to collect more than 1,200 malwaresamples that cover the majority of existing Android malware families, ranging from their debut in August 2010 to recentones in October 2011. In addition, they systematically characterize them from various aspects, including their installationmethods, activation mechanisms as well as the nature of carried malicious payloads.

FranziskaRoesner and his team [7] has taken the approach of user-driven access control, where permission grantingis built into existing user actions in the context of an application, rather than added via manifests or system prompts. Toallow the system to precisely capture permission-granting intent in an application's context, they introduce access controlgadgets (ACGs). Each user-owned resource exposes ACGs for applications to embed. The user's authentic UI interactionswith an ACG grant the application permission to access the corresponding resource. Their prototyping and evaluationexperience indicates that user driven access control enables in-context, non-disruptive and least-privilege permissiongranting on modern client platforms.

Yi Ying Ng, Hucheng Zhou, et. al [8] presents a comprehensive study on the trustworthy level of top popular Androidapp stores in China, by discovering the identicalness and content differences between the APK files hosted in the appstores and the corresponding official APK files. First, they have selected 25 top apps that have the highest installations inChina and have the corresponding official ones downloaded from their official websites as oracle; and have collected total506 APK files across 21 top popular app stores (20 top third party stores as well as Google Play). Afterwards, APKidentical checking and APK difference analysis are conducted against the corresponding official versions. Next,assessment is applied to rank the severity of APK files. All the apps are classified into 3 severity levels: ranging from safe(identical and higher level), warning (lower version or modifications on resource related files) to critical (modifications onpermission file and/or application codes). Finally, the severity levels contribute to the final trustworthy ranking score of the21 stores.

YuryZhauniarovich, Olga Gadyatskaya and Bruno Crispo [9] present how to enable the deployment of applicationcertification service, we called TruStores, for the Android platform. In their approach, the TruStore client enabled on theend-user device ensures that only the applications, which have been certified by the TruStore server, are installed on theuser smartphone. They envisage trusted markets (TruStore servers, which can be, e.g., corporate application markets) thatguarantee security by enabling an application vetting process. The TruStore

infrastructure maintains the open nature of theAndroid ecosystem and requires minor modi_cations to Android stack.

3. Malware Characterization

3.1 What is Malware?

Malware is software which is designed to damage or disrupt a System. Malicious software is abbreviated as Malware. Generally, software is considered malware based on the intent of the creator rather than its actual features. It can be classified as Viruses, worms, Trojanhorses, rootkits, backdoors, spyware, loggersand adware.

Trojan horse is any program that invites the user to run it, concealing a harmful or malicious payload. The payload may take effect immediately and can lead to many undesirable effects, such as deleting the user's files or further installing malicious or undesirable software. Rootkits Originally, a rootkit was a set of tools installed by a human attacker on a Unix system, allowing the attacker to gain administrator (root) access. Today, the term rootkit is used more generally for concealment routines in a malicious program.Once a malicious program is installed on a system, it is essential that it stays concealed, to avoid detection and disinfection. Backdoors may also be installed prior to malicious software, to allow attackers entry.

Spyware is a type of malicious software that can be installed on computers, and which collects small pieces of information about users without their knowledge. The presence of spyware is typically hidden from the user, and can be difficult to detect. Spyware programs can collect various types of personal information, such as Internet surfing habits and sites that have been visited, but can also interfere with user control of the computer in other ways, such as installing additional software and redirecting Web browser activity. Keystroke logging (often called keylogging) is the action of tracking (or logging) the keys struck on a keyboard, typically in a covert manner so that the person using the keyboard is unaware that their actions are being monitored. There are numerous keylogging methods, ranging from hardware and software-based approaches to electromagnetic and acoustic analysis. Adware, or advertising-supported software, is any software package which automatically plays, displays, or downloads advertisements to a computer. These advertisements can be in the form of a pop-up. The object of the Adware is to generate revenue for its author. Adware, by itself, is harmless; however, some adware may come with integrated spyware such as keyloggers and other privacy-invasive software. [15]

Malware writers/users go by a variety of names. Some of the most popular names are black hats, hackers, and crackers. In creating new malware, black hats generally employ one or both of the following techniques: obfuscation and behavior addition/modification in order to circumvent malware detectors [16]. Hacker is any highly skilled computer expert capable of breaking into computer systems and networks using bugs and exploits. A cracker (also known as a black hat hacker) is an individual with extensive computer knowledge whose purpose is to breach or bypass internet security or gain access to software without paying royalties.

The malware detector attempts to help protect the system by detecting malicious behavior. The malware detector may or may not reside on the same system it is trying to protect. The malware detector performs its protection through the manifested malware detection technique.

Systematic characterization of existing malware into three broad categories as follow: Installation, Activation, Carried Payloads. [17]

3.2 Installation

Android malware use to install onto user phones and generalize them into three main social engineering-based techniques, i.e., repackaging, update attack, and drive-by download. These techniques are not mutually exclusive as different variants of the same type may use different techniques to entice users for downloading.

Repackaging is one of the most common techniques malware authors use to piggyback malicious payloads into popular applications (or simply apps). In essence, malware authors may locate and download popular apps, disassemble them, enclose malicious payloads, and then re-assemble and submit the new apps to official and/or alternative Android Markets.

Update attack, second technique makes it difficult for detection. Specifically, it may still repackage popular apps. But instead of enclosing the payload as a whole, it only includes an update component that will fetch or download the malicious payloads at runtime.

The drive-by download technique applies the traditional drive-by download attacks to mobile space. Though they are not directly exploiting mobile browser vulnerabilities, they are essentially enticing users to download interesting apps.

3.3 Activation

Android malware can rely on the built-in support of automated event notification and callbacks on Android to flexibly trigger or launch its payloads.

3.4 Carried Payloads

The payload functionalities partition into four different categories: privilege escalation, remote control, financial charges, and personal information stealing.

The Android platform is a complicated system that consists of not only the Linux kernel, but also the entire Android framework with more than 90 open-source libraries included, such as WebKit, SQLite, and OpenSSL. The complexity naturally introduces software vulnerabilities that can be potentially exploited for privilege escalation.

During analysis to examine the remote control functionality among the malware payloads, authors are surprised to note that 93.0% turn the infected phones into bots for remote control. One profitable way for attackers is to surreptitiously subscribe to (attacker-controlled) premium-rate services, such as by sending SMS messages.

In addition to the above payloads, malware are actively harvesting various information on the infected phones including SMS messages, phone numbers as well as user accounts. For Android apps without root exploits, their capabilities are strictly constrained by the permissions users grant to them. Therefore, it will be interesting to compare top permissions requested by these malicious apps with top permissions requested by benign ones.

4. Permission Categories

Most device access in android is controlled by permissions. Applications can define their own extra permissions, but here the permissions defined by Android OS are considered only. There are 134 permissions in Android2.2. Permissions are categorized into following threat levels

Level 1: API calls with annoying but not harmful consequences are protected with Normal permissions. Example: accessing information about available Wi-Fi networks, vibrating the phone, and setting the wallpaper.

Level 2: API calls with potentially harmful consequences. Example: Opening a network socket, recording audio, and using the camera.

Level 3: The most sensitive operations are protected with Signature permissions. These permissions are only granted to applications that have been signed with the device manufacturer's certificate. Example: Ability to inject user events.

Level 4: This category includes signed applications and applications that are installed into the/system/app folder. Example: Preinstalled applications, applications protecting the ability to turn off the phone. During installation permission prompt is displayed to the user for level 2 permissions. Warnings are categorized according to functionality. For example, Dangerous location related permissions are included in location related warning. Level 1 permissions are hidden in a collapsed menu. Level 3 permissions are not shown at all.[10] Following table shows available android permission groups with respective permissions.

Permission Group	Permission
android.permission.group.CALENDAR	**android.permission.READ_CALENDARandroid.permission.WRITE_CALENDAR**
android.permission.group.STORAGE	**android.permission.READ_EXTERNAL_STORAGEandroid.permission.WRITE_EXTERNAL_STORAGE**
android.permission.group.SMS	**android.permission.SEND_SMSandroid.permission.RECEIVE_SMSandroid.permission.READ_SMSandroid.permission.RECEIVE_WAP_PUSH android.permission.RECEIVE_MMSandroid.permission.READ_CELL_BROADCASTS**
android.permission.group.SENSORS	**android.permission.BODY_SENSORS**
android.permission.group.CAMERA	**android.permission.CAMERA**
android.permission.group.CONTACTS	**android.permission.READ_CONTACTSandroid.permission.WRITE_CONTACTSandroid.permission.GET_ACCOUNTS**
android.permission.group.LOCATION	**android.permission.ACCESS_FINE_LOCATIONandroid.permission.ACCESS_COARSE_LOCATION**
android.permission.group.MICROPHONE	**android.permission.RECORD_AUDIO**
android.permission.group.PHONE	**android.permission.READ_PHONE_STATEandroid.permission.CALL_PHONEandroid.permission.READ_CALL_LOGandroid.permission.WRITE_CALL_LOGcom.android.voicemail.permission.ADD_VOICEMAILandroid.permission.USE_SIPandroid.permission.PROCESS_OUTGOING_CALLS**

Table 1 :Android Permissions

Conclusion

During this review, it was observed that there is need of more study on detection of inter-app permission leakage withany dynamic strategy which will guide users to identify hidden malwares in android operating system. This paper also focuses on a huge demand of new solution for user centric risks control and giverequisite information to the user about app behavior with user centric risks available into it.

References

[1] G. Hamid Bagheri, AlirezaSadeghi, Joshua Garcia and Sam Malek, "COVERT: Compositional Analysis ofAndroid Inter-App Permission Leakage", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, 2015.

[2] AlexiosMylonas, MarianthiTheoharidou, and DimitrisGritzalis, "Assessing Privacy Risks in Android: A UserCentricApproach", DOI: 10.1007/978-3-319-07076-6_2, Springer International Publishing Switzerland 2014.

[3] Li Li, AlexandreBartel, Tegawende F. Bissyand,"IccTA: Detecting Inter-Component Privacy Leaks in Android Apps", 37th International Conference on Software Engineering (ICSE 2015), ITALY,2015.

[4] Li Li, AlexandreBartel, Jacques Klein, Yves le Traon, "Automatically Exploiting Potential Component Leaks in Android Applications", IEEE Conference, Sept 2014.
[5] Drago S, Michael G. Burke, Salvatore Guarnieri, "Automatic Detection of Inter-application Permission Leaksin Android Applications", IEEE Conference, Nov.2013 .
[6] Yajin Zhou, Xuxian Jiang, "Dissecting Android Malware: Characterization and Evolution", Security and Privacy (SP), IEEE, May 2012.
[7] FranziskaRoesner, Tadayoshi Kohno, Alexander Moshchuk, Bryan Parno, "User-Driven Access Control: Rethinking Permission Granting in Modern Operating Systems", IEEE ,2012.
[8] Yi Ying Ng , Hucheng Zhou , ZhiyuanJi , HuanLuo, "Which Android App Store Can be Trusted in China?",Computer Software and Applications Conference (COMPSAC), IEEE, July 2014.
[9] YuryZhauniarovich, Olga Gadyatskaya, and Bruno Crispo, "Trustore: Implementing A Trusted Store ForAndroid", DISI - Via Sommarive 5 - 38123 Povo - Trento (Italy), May 2014.
[10] Hari H. Rajai, Prof. SachinBojewar, "Study Of Permissions And Risk Communication Mechanisms In Android",International Research Journal of Engineering and Technology (IRJET),2015
[11] Felt, A., Chin, E., Hanna, S., Song, D., Wagner, D., "Android permissions demystified.", Proceedings of the 18thACM Conference on Computer and Communications Security, pp. 627–638. ACM (2011).
[12] Felt, A., Hanna, S., Chin, E., Wang, H.J., Moshchuk, E. "Permission redelegation: attacks and defenses" In: 20thUsenix Security Symposium (2011).
[13] Grace, M., Zhou, Y., Wang, Z., Jiang, X. "Systematic detection of capability leaks in stock Androidsmartphones.", Proceedings of the 19th Network and Distributed System Security Symposium (2012).
[14] Pearce, P., Felt, A.P., Nunez, G., Wagner, D. "Android: privilege separation for applications and advertisers in android.",Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security, pp.71–72. ACM (2012) G.Y.Goldstein."StrategicInnovationManagement:Trends, Technology, Practice:AMonograph".Taganrog:PublishingHouseTRTU, 2002.
[15] http://www.idconline.com/technical_references/pdfs/informationtechnology/Malware%20and%20its%20types.pdf
[16] NwokediIdika, Aditya P. Mathur, "A Survey of Malware Detection Techniques", February 2, 2007.
[17] Yajin Zhou, Xuxian Jiang "Dissecting Android Malware: Characterization and Evolution", North Carolina State University, 2013