Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Permission based Android security: Issues and countermeasures

Zheran Fang [a,c], Weili Han [a,c,*], Yingjiu Li [b]

[a] Software School, Fudan University, Shanghai, 201203, China
[b] School of Information Systems, Singapore Management University, Singapore
[c] Shanghai Key Laboratory of Data Science, Fudan University, Shanghai, China

ARTICLE INFO

Article history:
Received 10 September 2013
Received in revised form
21 January 2014
Accepted 19 February 2014

Keywords:
Android security
Permission based security
Access control
Granularity of access control
Policy administration
Over-claim of permission
Permission escalation attack

ABSTRACT

Android security has been a hot spot recently in both academic research and public concerns due to numerous instances of security attacks and privacy leakage on Android platform. Android security has been built upon a permission based mechanism which restricts accesses of third-party Android applications to critical resources on an Android device. Such permission based mechanism is widely criticized for its coarse-grained control of application permissions and difficult management of permissions by developers, marketers, and end-users. In this paper, we investigate the arising issues in Android security, including coarse granularity of permissions, incompetent permission administration, insufficient permission documentation, over-claim of permissions, permission escalation attack, and TOCTOU (Time of Check to Time of Use) attack. We illustrate the relationships among these issues, and investigate the existing countermeasures to address these issues. In particular, we provide a systematic review on the development of these countermeasures, and compare them according to their technical features. Finally, we propose several methods to further mitigate the risk in Android security.

Introduction

Android security has been under a spotlight in information security as Android smartphones become the most popular mobile devices in the current market. Since the first Android-powered phone was delivered in October 2008 (Gozalvez, 2008), Android smartphones have grown to the largest global market share (75%) among all smartphones shipped in the first quarter of 2013 (IDC, 2013). In May 2013, Google announced that 900 million Android devices had been activated (Welch, 2013). According to F-Secure, a cyber security-related company, the number of new mobile threat families and variants continued to rise by 49% from the previous quarter; 91.3% of these threats targeted at Android devices in the first quarter of 2013 (F-Secure, 2013).

Android smartphones are protected by a permission based framework which restricts third-party applications' accesses to sensitive resources such as SMS database and external storage in Android smartphones. The accesses to sensitive resources may lead to money loss. For example, Android malware may send premium rate messages, make premium rate calls, and generate large amount of network data without users' acknowledgment. Moreover, the

* *Corresponding author*. Software School, Fudan University, Shanghai, 201203, China. Tel.: +86 (0)21 51355388.
E-mail addresses: 13212010002@fudan.edu.cn (Z. Fang), wlhan@fudan.edu.cn (W. Han), yjli@smu.edu.sg (Y. Li).
http://dx.doi.org/10.1016/j.cose.2014.02.007
0167-4048/

accesses to sensitive resources may lead to leakage of users' private information stored in smartphones such as contacts, emails, and even credit card numbers. Third-party application developers can leverage various smartphone sensors such as GPS, cameras, and microphones, then create applications that do more than what they claimed so as to collect users' private information stealthily (Fragkaki et al., 2012). In the current Android permission framework, accesses to critical resources on smartphones are controlled according to permissions given to applications at install-time. That is, each application must request for certain permissions for it to access system resources on a smartphone at install-time and the user of the smartphone should make a decision on whether or not to grant the permissions requested.[1]

Such permission based framework is criticized as coarse-grained. Many applications tend to request much more permissions than necessary. In most cases, a user has to either grant all permissions an application requests or abort the installation process, instead of granting the permissions one by one. In addition, the permission based framework is vulnerable due to insufficient control of cooperation among applications and poor documentation on how to use various permissions.

Android security has attracted much attention from both academia and industry. To the best of our knowledge, papers related to Android security appeared as early as 2008 (Enck et al., 2008; Schmidt et al., 2008), which is the same year when the first Android-powered smartphone was delivered. Along with the explosive growth of Android-powered devices in the following years, a considerable number of research papers on Android security have been published.

Given the large number of published researches on Android security, especially on Android permission framework, we provide a systematic overview of the current state of Android security. In particular, we investigate the recent advancement on Android security, identify the issues in Android permission framework, and analyze the countermeasures to address the security issues.

The rest of this paper is organized as follows: Section 2 introduces the background of Android security. Section 3 classifies the issues in Android permission framework. Section 4 investigates existing solutions to address Android security issues. Section 5 discusses the future work. Finally, Section 6 concludes the paper.

Background of Android security

Android is proposed as a software stack for mobile devices. It consists of an operating system, an application framework, and core applications. Each Android application executes in a separate Dalvik virtual machine instance running as a unique user identity assigned at install-time. Thus applications are essentially isolated. This design provides promising security for file accesses and limits potential damage due to programming flaws such as buffer overflow (Enck et al., 2011).

Android restricts accesses to critical resources using permissions. A permission is simply a unique text string which can be defined by Android or third party developers. According to the documentation for Android developers, there are currently 130 permissions (Android, 2013b), which are defined in Android operating system, ranging from access to camera (`CAMERA`), full access to the Internet (`INTERNET`), dialing a phone number (`CALL_PHONE`), and even disabling the phone function permanently (`BRICK`). According to the study of Wei et al. (2012), the number of Android defined permissions keeps increasing since the first widely-used release (API level 3). The expansion of the permission set aims at not only providing finer-grained permissions but also controlling accesses to new hardware features (Wei et al., 2012). In addition to Android defined permissions, application developers can also declare customized permissions so as to protect their own critical resources.

Permissions may be required when an application is interacting with system resources, including calling system API functions, and reading from and writing to file systems. Granted permissions are assigned to an application's sandbox and inherited by all of the application's components, while required permissions are assigned to application components (Bugiel et al., 2011a). In the manifest file of an application, which is included in the application package, the application declares the permissions which it requires to achieve its functionality, as well as defines the permissions for protecting its own components and resources. A permission can be associated with one of the following four protection levels (Android, 2013a):

- *Normal*: A low-risk permission which allows applications to access API calls (*e.g.*, `SET_WALLPAPER`) causing no harm to users.
- *Dangerous*: A high-risk permission which allows applications to access potential harmful API calls (*e.g.*, `READ_CONTACTS`) such as leaking private user data or control over smartphone device.
- *Signature*: A permission which is granted if its requesting application is signed with the same certificate as the application which defines the permission is signed.
- *Signature-or-system*: A permission which is granted only if its requesting application is in the same Android system image or is signed with the same certificate as the application which defines the permission is signed.

At install-time, a user is shown with a list of permissions which an application requests. The user must either grant or deny all of these permissions together. After the user approves the permission request and installs the application, the application owns its permissions throughout its lifetime and it does not need to request them again at run-time. Android controls Inter-Component Communication (ICC) through a reference monitor. The reference monitor provides a *Mandatory Access Control (MAC)* enforcement on how applications access components by evaluating whether the applications are granted with necessary permissions.

[1] In the recent version of Android 4.3, users can revoke an application's permissions after the application is installed.