Intent Based Security Challenges in Android-An Analysis & Recommendation

Janaki Sivakumar[†], **Ammar Yassir**[††], **P.Saravanan**[†††]
[†]Research Scholar, PRIST University, Tanjore, TamilNadu.
[††]Research Scholar, CMJ University, Shillong, India
[†††]Research Scholar, Manonmanium Sundaranar University, Trinelveli, India

Abstract
Recent years have witnessed a meteoric increase in the adoption of smart phones. The number of Android based smart phones is growing rapidly. They are increasingly used for security critical private and business applications, such as online banking or to access corporate networks. This makes them a very valuable target for an adversary. To manage such information and features, Android provides a permission-based security model that requires each application to explicitly request permissions before it can be installed to run. Several privileged permissions are unsafely exposed to other applications, which do not need to request them for the actual use. Up to date, significant or large-scale attacks have failed, but attacks are becoming more sophisticated and successful. Thus, security is of paramount importance for both private and corporate users. This paper gives a short, yet comprehensive overview of the major Android security mechanisms.

Keywords
Android OS, Seepage, Manifest file, Smart phone infections, Android APPs, Broadcast receiver, Security tools.

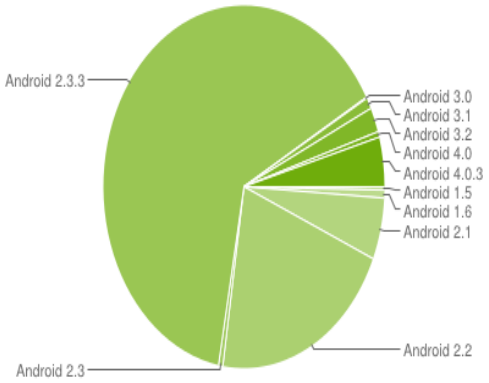
1. INTRODUCTION

According to data from IDC [1], smart phone manufacturers shipped 100.9 million units in the fourth quarter of 2010, compared to 92.1 million units of PCs shipped worldwide. For the first time in history, smart phones are outselling personal computers. Their popularity can be partially attributed to the incredible functionality and convenience smart phones offered to end-users.

In fact, existing mobile phones are not simply devices for making phone calls and receiving SMS messages, but powerful communication and entertainment platforms for web surfing, social networking, GPS navigation, and online banking. The popularity of smart phones smart phones is also spurred by the proliferation of feature-rich devices as well as compelling mobile applications (or simply apps). In particular, these mobile apps can be readily accessed and downloaded to run on smart phones from various app stores. For example, it has been reported [2] that Google's Android Market already hosts 150,000 apps as of February 2011 and the number of available apps has tripled in less than 9 months. Not surprisingly, mobile users are increasingly relying on smart phones to store and handle personal data. Inside the phone, we can find current (or past) geo-location information smart phones about the user, phone call logs of placed and received calls, an address book with various contact information, as well as cached emails and photos taken with the built-in camera. The type and the volume of information kept in the phone naturally lead to various concerns [3, 4, 5, 6] about the safety of this private information, including the way it is managed and accessed. Smart phones In this paper, we give an overview of the current state of the art of android security with the widespread use of smart phones both in private and work related areas, securing these devices has become of paramount importance.

Owners use their smart phones to perform tasks ranging from every day communication with friends and family to the management of banking accounts and accessing sensitive work related data.

Android versions distributed as of May 2012[7]



2. ANDROID AN OVERVIEW

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. There is a business alliance (Open Handset Alliance-OHA) which consists of **47** companies to develop

Manuscript received January 5, 2013
Manuscript revised January 20, 2013

open standards for mobile devices. Figure 1: shows the Android software stack Architecture.

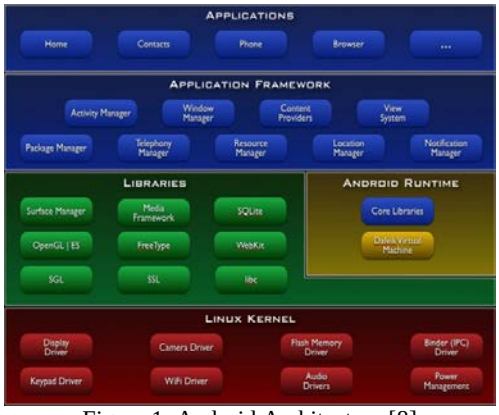


Figure 1: Android Architecture [8]

Android provides a set of core applications, which are written in Java Language:

- Email Client
- SMS Program
- Calendar
- Maps
- Browser
- Contacts

Android software stack includes a set of C/C++ libraries used by components of the Android system. Various libraries enable apps to implement graphics, encrypted communication or databases easily. The standard Library ("bionic") is a BSD derived library for embedded devices.
Android Software stack is developed in Java, executed in a virtual machine, called Dalvik VM and relying on Linux kernel 2.6 for core system services like memory and processor management, network stack, Driver model and Security.
There are many features related to Android software stack

Features	*Role*
View system	Used to build an application, including lists, grids, text boxes, buttons, and embedded web browser
Content Provider	Enabling applications to access data from other applications or to share their own data
Resource Manager	Providing access to non-code resources (localized strings, graphics, and layout files)
Notification Manager	Enabling all applications to display customer alerts in the status bar
Activity Manager	Managing the lifecycle of applications and providing a common navigation backstack

3. INTENTS

Android provides a sophisticated message passing system, in which Intents are used to link applications. Intent is a message that declares a recipient and optionally includes data; an Intent can be thought of as a self-contained object that specifies a remote procedure to invoke and includes the associated arguments [9]. Applications use Intents for both inter-application communication and intra-application communication. Intents can be sent between three of the four Android components: Activities, Services, and Broadcast Receivers. Intents can be used to start Activities; start, stop, and bind Services; and broadcast information to Broadcast Receivers.
Intents can be used both for explicit or implicit communication.

3.1 Explicit Intent

An explicit Intent specifies that it should be delivered to a particular application specified by the Intent. An explicit Intent identifies the intended recipient by name.

3.2 Implicit Intent

An implicit Intent requests delivery to any application that supports a desired operation. An implicit Intent leaves it up to the Android platform to determine which application(s) should receive the Intent

For example, consider an application that deals Client's Email information. When the user clicks on a sender's contact address, the Email application needs to ask another application to display a map of that location. To achieve this, the Email application could send an explicit Intent directly to Google Maps, or it could send an implicit Intent that would be delivered to any application that says it provides mapping functionality (e.g., Yahoo! Maps or Bing Maps). Using an explicit Intent guarantees that the Intent is delivered to the intended recipient, whereas implicit Intents allow for late runtime binding between different applications.

4. ANDROID COMPONENTS

Intents are delivered to application components, which are logical application building blocks. Android defines four types of components:
4.1 Activities: provide user interfaces. Activities are started with Intents, and they can return data to their invoking components upon completion [10]. All visible portions of applications are Activities.

4.2 Services: run in the background and do not interact with the user. Downloading a file or decompressing an archive are examples of operations that may take place in a Service [10]. Other components can bind to a Service, which lets the binder invoke methods that are declared in the target Service's interface. Intents are used to start and bind to Services.

4.3 Broadcast Receivers: receive Intents sent to multiple applications. Receivers are triggered by the receipt of an appropriate Intent and then run in the back- ground to handle the event. Receivers are typically short-lived; they often relay messages to Activities or Services [10]. Operating system sends

Intents to applications as event notifications. Some of these event notifications are system-wide events that can only be sent by the operating system. We call these messages system broadcast Intents.

There are three types of broadcast Intents:

a) Normal
b) Sticky
c) Ordered

a) Normal broadcasts are sent to all registered Receivers at once, and then they disappear.

b) Ordered broadcasts are delivered to one Receiver at a time; also, any Receiver in the delivery chain of an ordered broadcast can stop its propagation. Broadcast Receivers have the ability to set their priority level for receiving ordered broadcasts.

c) Sticky broadcasts remain accessible after they have been delivered and are re-broadcasting to future Receivers.

4.4 Content Providers: are databases addressable by their application-defined URIs. They are used for both persistent internal data storage and as a mechanism for sharing information between applications.

5. ANDROID SECURITY

Android's security model differs significantly from the standard desktop security model. Android applications are treated as mutually distrusting principals; they are isolated from each other and do not have access to each other's private data. We focus on Android because it has the most sophisticated application communication system. The complexity of Android's message passing system implies it has the largest attack surface. To provide theoretical background, we give a short explanation of fundamental Android security measures provides an overview of current and emerging threat scenarios on smart phones both for private and corporate targets.

5.1 Manifest File

On installation, the user is presented with a dialog listing all permissions requested by the app to be installed. These permission requests are defined in the Manifest File AndroidManifest.xml

This system has a few flaws:
• All or none policy: A user cannot decide to grant single permissions, while denying others. Many users, although an app might request a suspicious permission among much seemingly legitimate permission, will still confirm the installation.

• Cannot judge the appropriateness of permissions: Often, users cannot judge the appropriateness of permissions for the application in question. In some cases it may be obvious, for example when a game
app requests the privilege to reboot the smartphone or to send text messages. In many cases, however, users will simply be incapable of assessing permission appropriateness.

• Circumvention: Functionality, which is supposed to be executable only given the appropriate permissions, can still be accessed with less permission or even with none at all.

5.2 Security Holes

Multiple security holes have been found in different components of the Android operating system. Up until now, they have primarily served to grant device owners administrative privileges on their devices. Only recently malware authors have begun utilizing such holes and publicly available exploits for malicious code.

5.4 Broadcast scenarios

For mobile malware, current broadcast scenarios significantly differ from those of desktop malware. Direct self-spreading mechanisms over primary communication networks known from desktop environments are very unlikely.
However, different approaches exist, which utilize existing infrastructure such as the Android Market and websites. Threat Broadcast scenarios may vary like

- ✓ Seepage through Logging Service.
- ✓ Seepage through Online Banking.

- ✓ Seepage through Contact Information, Location Data, Credentials and Private Details.
- ✓ Seepage through GSM based Pivot Attacks.

5.5 Types of infections

1.Wicked applications; are the most common infection channel and are comparable to Trojan programs on desktop platforms .Wicked code can be packaged and redistributed with popular applications. Furthermore, users can choose to allow installation from websites, which can also be exploited by hackers
2.Infection via Personal Computers: Technically, desktop computer malware have to implement the Android Debug Bridge's protocol to install arbitrary software on any device with USB debugging activated.
3.Rooting; one's smartphone may introduce higher risks of successful wicked infection Application markets preconfigured for rooted or modified operating systems are not well monitored and contain many Trojan programs. Thus, rooting a smartphone may pose a high security risk.

3.1 Device to Device Infection; with Android versions 3.1 and 4.0, two major changes have been introduced which may serve for device to device propagation:

- USB host mode (Android 3.1)-an Android smartphone may use the Android Debug Bridge to push and install malicious apps to other devices with USB debugging enabled. This may happen both intentionally or unintentionally.
- Android Beam (Android 4.0)-it requires user interaction for installation. For example, a web like to a malicious app can be sent to another Android 4 device via Android Beam, but the user still has to click the link and confirm it. The limited physical distance reduces malware infection risks even further.

4. Infection via Rogue: Wireless Networks users logging into the rogue wireless network may be presented with a fake website displaying a "critical update" to an app installed on nearly all devices such as Google Search.

5.6 Tools to avoid Seepage

1. ComDroid: To detect potential vulnerabilities in Android applications. ComDroid can be used by developers to analyze their own applications before release, by application reviewers to analyze applications in the Android Market, and by end users.

2. Wood Pecker: To identify leaked permissions or capabilities in Android applications. Woodpecker employs inter procedural data flow analysis techniques to systematically expose possible capability leaks where an untrusted app can obtain unauthorized access to sensitive data or privileged actions data.

5.7 Recommendation to avoid Seepage

→ Android based smart phones should prohibit the execution of any native code added after shipping completely. Prohibiting non vendor native code is the only way to contain exploits against system security flaws.
→ The privilege of setting the executable bit for files may be limited to the root user and to the Android Market app. Any file added to the file system later on, i.e. not during the installation process, cannot be declared as executable. SEAndroid has proven very effective for this objective.
→ Code signing of native code may also be an option. Prohibiting execution of non-signed binaries could also prevent usage of exploits in usual apps.
→ Customers should get Awareness on security a risk on mobile platforms seems significantly lower than on desktop platforms. Identical or even more caution should be applied for mobile devices
→Finally, Google. Of course, device manufacturers should be responsible to supply their customers with security patches

6. CONCLUSION

While the Android message passing system promotes the creation of rich, collaborative applications, it also introduces the potential for attack if developers do not take precautions. Outgoing communication can put an application at risk of Broadcast theft (including eavesdropping and denial of service), data theft, result modification, and Activity and Service hijacking. Incoming communication can put an application at risk of malicious Activity and Service launches and Broadcast injection.

REFERENCES

[1] IDC. Android Rises, Symbian 3 and Windows Phone 7 Launch as Worldwide Smartphone Shipments Increase 87.2% Year Over Year. http://www.idc.com/about/viewpressrelease.jsp?
[2] J. Hildenbrand. 150,000 apps in Android Market, tripled in 9 months. http://www.androidcentral.com/150k-apps-android-market-tripled-9-months.
[3] M. Egele, C. Kruegel, E. Kirda, and G. Vigna. PiOS: Detecting Privacy Leaks in iOS Applications. In
[4] Proceedings of the 18th Annual Network and Distributed System Security Symposium, NDSS '11, February 2011.
[5] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. Mc-Daniel, and A. N. Sheth. TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones. In Proceedings of the 9th USENIX

Symposium on Operating Systems Design and Implementation, OSDI '10, February 2010.
[6] K. Mahaffey and J. Hering. App Attack-Surviving the Explosive Growth of Mobile Apps. https://media. blackhat.com/bh-us-10/presentations/Mahaffey_Hering/Blackhat-USA-2010-Mahaffey-Hering-Lookout-App-Genomeslides. Pdf
[7] Y. Zhou, X. Zhang, X. Jiang, and V. Freeh. Taming Information-Stealing Smartphone Applications (on Android).In Proceedings of the 4th International Conference on Trust and Trustworthy Computing, TRUST '11, June 2011.
[8] Android Team, "Platform Versions," February 2012. http://developer.android.com/resources/dashboard/platformversions.html.
[9] Android Team, "What is the NDK?," January 2012.http://developer.android.com/sdk/ndk/overview.html.
[10] Android Team, "What is Android?," Feburary 2012. http://developer.android.com/guide/basics/whatisandroid.html.
[11] "Analyzing Inter-Application Communication in Android " Erika Chin Adrienne Porter Felt Kate Greenwood David Wagner University of California, Berkeley , USA

Janaki Sivakumar received M.C.A degree in 2002 from Bharathidasan University and M.Phil (C.Sc) from Mother Teresa University Kodaikanal in 2004.Currently she is a research scholar in Computer Science, PRIST University, Tanjore, India. Her main area of interest is in Image and Signal Processing. She has published papers in 3 International Journals & attended 2 International Conferences.

Ammar Yassir received the B.Sc. degree with Honors in Computer Science in the year 2002 from Future University, Sudan & Master in Business Administration and IT from SMU, India in 2006 & currently a Ph.D. candidate in IT, CMJ University, Shillong, India. He has published International papers in several Journals.

P.Saravanan has received M.Tech (Computer Science & Data Processing) from IIT, India in 1995. He graduated M.Sc (Applied Mathematics) from Anna University, India in 1987 and M.phil from Madras University 1992. Currently he is working as a senior lecturer in Muscat College, Oman. He is doing research in Image processing. He published papers in two international journals.