Advances in Computational Sciences and Technology
ISSN 0973-6107 Volume 10, Number 5 (2017) pp. 1449-1455

http://www.ripublication.com

Different Android Vulnerabilities

Prateek Bhiwani

Student M.tech.(Cyber Security), Department of Information Technology
Raksha Shakti University, Ahmedabad, India.

Chandresh Parekh

Assistant Professor, Department of Telecommunication
Raksha Shakti University, Ahmedabad, India.

Abstract

Android is the most popular OS in the world. A few weeks ago it took the number one OS spot from windows. But Ever since android came into the market it has become the best Smartphone OS which put it in lime light. Android became so because it is open source and is flexible. But to gain that spot it trade with security of data. Android is most famous and at the same time the most vulnerable OS in the world. In this paper I will discuss about android, its components and its vulnerability. In this i have discussed some of the major attacks and vulnerability in android.

Keywords: Intent, shared preference, ICC Leak, attacks

I. INTRODUCTION

Android OS in most widely used Operating System right now in the world. It is because of user friendliness for user and because it is open source for different vendors to customize so they can add addition functionality or software which is specific to their brand device.

Android provides an ICC model i.e. Inter-component communication which provides a way to send data not only between two different applications but also between two components in the same application. However this mechanism for message and data passing can be misused by malicious application to gain access and send sensitive information outside. Research work is done to find a measure for this leak however

not much is done because this loosely couple method for communication provides flexibility for which android is famous among developer. Malicious application take permission from user in lieu of functionality and then send data out without even user know it.

II. DATA SHARING IN ANDROID

Data in android is shared by two methods : 1)Intent 2)shared preferences

A. ***Intent***

Intent is of two type *implicit* and *explicit* intent. Intent object is used to switch between activities and also for data passing. In Implicit intent android provide a mechanism in which it gives options to users to choose between different application which provide same feature.

In explicit intent an application or target is specified beforehand so it does not provide any option to user.

B. ***Shared Preference***

Shared preference is used by application for data sharing also. It is a shared space where data is stored by an application to use it afterward or by some other application.

III. DATA LEAK

Data can be leaked easily through either explicit intent or shared preference in android. As explicit intent ask for no choice so malicious application take all permission from user beforehand in lieu of functionality and features and then keep sending users personal information without user knowledge.

A. ***Permission seeked by malicious application***

There are different permission which malicious application ask so they can get access to user sensitive data.

- INTERNET
- WRITE_EXTERNAL_STORAGE
- WAKE_LOCK
- ACCESS_COARSE_LOCATION
- READ_PHONE_STATE
- SEND_SMS
- RECEIVE_SMS
- READ SMS

- ACCESS_FINE_LOCATION
- WRITE_SMS

Above permission are listed as top permission that malicious application ask in lieu of functionality. Even though application doesn't need to access many of things. Google in its latest OS has given users the power that if user want to give the permission or not and have asked developer to mention their policies and allow user to at least download application without granting any permission so that user can decide whether user should give permission at the time of use. However even with these many steps taken by Google malicious application still get access to user data because of users trust many application blindly and give away the permission.

B. ***Intent used in infected application***

Malicious application uses explicit intent to access data and send it to write it on disk. Following are some intent that are mostly used:

- BOOT_COMPLETED
- SENDTO
- DIAL
- SCREEN_OFF
- TEXT
- SEND
- USER_PRESENT
- PACKAGE_ADDED
- SCREEN_ON
- CALL

IV. DIFFERNET TYPE OF ICC A TTACK

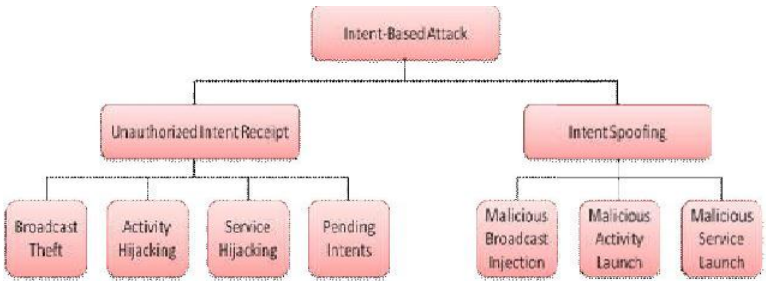

Figure 1: ICC Attack

Figure 1 show different types of ICC attacks. ICC is a powerful mechanism that promotes collaboration and code reuse among apps. ICC is also an application attack surface.

A. ***Unauthorized Intent Receipt***

There are no way that an implicit Intent will be received by the intended recipient. A malicious component can intercept the Intent by declaring an Intent filt er. The malicious component can then gain access to all the data in the intercepted Intent.

1. Broadcast theft: Broadcasts are vulnerable to passive eavesdropping and active denial of service attack. A malicious broadcast receiver intercepts an Intent or prevents it from reaching other broadcast receivers.12% of apps are vulnerable to this attack.
    a. Broadcast Theft-passive Eavesdropping: When a component sends an unprotected broadcast in tent, the system sends the intent to all the registered receivers. An eavesdropper receiver silently receives the broadcast Intent.
    b. Broadcast Theft – DOS Attack: An active attacker could launch a DOS attack or data injection attacks on a broadcast receiver. Non-ordered broadcast Intents are sent to all registered receivers at the same time.
2. Activity Hijacking: Malicious activity registers to receive another component's implicit Intent. Malicious activity is launched in place of the intended Activity. The malicious activity could read the data sent to another activity. The hijacker could spoof the expected activity user interface to steal user-supplied data.57% of apps are vulnerable to this attack.
3. Service Hijacking: Service hijacking occurs when a malicious Service receives an implicit Intent intended for a legitimate Service. When numerous Services can handle an Intent, then there is a random selection. Contrasting from Activity hijacking, Service hijacking did not deals with users because there is no user interface. If the calling app binds to a malicious Service, then it can return incorrect data (i.e., false response attack).
4. Pending Intent Delegation Attack: Pending Intent is a type of Intent with a target action that can be performed at a later time by the holder of that Intent. One App can send a Pending Intent to another app to perform the action on behalf of the sender at a later time. By giving a Pending Intent to other application, you are giving it the right to execute the process you have specified as if the other application was yourself (with the same permissions and identity).

B. ***Intent spoofing ICC Atta ck***

A malicious attacker can launch an intent spoofing attack by sending an intent to an exported component that is not waiting for the intent from the sender component. If

the receiver takes an action with the received intent , the attacker can trigger a malicious action .

1. Malicious broad cast injection: Malicious component sends an Intent, either explicit or implicit, to an exported Broadcast Receiver. The Broadcast Receiver blindly trusts an incoming broadcast Intent, without checking the authenticity of the sender. Broadcast Receivers that listen to system broadcast Intents are particularly at risk. These broadcast receivers operate in response to system actions without checking to see if the system has sent the Intent.14% of apps are vulnerable to this attack. packageInstalledReceiver receives a broad cast Intent from the system whenever a new app installed. Once packageInstalledReceiver receives the Intent, it starts newPackage Service to scan that app from viruses. Malicious Activity can start package Installed Receiver using an explicit Intent and provide a fake app's name.
2. Malicious Activity Launch: This attack is analogous to cross-site request forgeries (CSRF).Android App comprises many entry points. Any component can start any exported Activity using explicit or implicit Intent. The system loads the launched Activity's user interface. In addition to disrupting/annoying the user, the launched Activity might:
    - Update a stored data and change the app's state.
    - Return sensitive data to the called component.12% of apps are vulnerable to this attack.
3. Malicious Service Launch: Similar to the previous attack, a malicious component can start or bind to any exported Service and not protected with a permission. The started/bound Service might leak sensitive information or corrupt the app's state.1% of apps are vulnerable to this attack.

V. PERMISSION RE -DELEGATION ATTACK

Application with permission(s) performs a privileged task for an application without permission(s).sendMsg component executes a privileged task on behalf of an unprivileged component, LocLeaker. The attack allows the Locleaker to leak location n information to a premium number.

VI. ATTCKS THROUGH HIDDEN CODE

Malwar attackers usually hide their code to prevent analysis tools from detecting it. There are many ways to hide code like code encryption, reflection and dynamic class loading.

Dynamic class loading: Android allows apps to load code from external sources at runtime. Developer uses this feature as an instrument to collect data during beta testing from users. Multiple apps use the same library which can be installed on device as a separate APK file to reduce space.

1. Code injection attacks against benign apps
    - Android system does not check the integrity of the downloaded class files.
    - Oftentimes developers don't check the integrity of the installed codes.
    - Attackers can inject a malicious code and replace the expected legitimate code.
    - 16% of the top apps are using this feature in an insecure way.
2. Executing another app's code
    - *createPackageContext(pkg, flags)* API is an Android API that returns a lightweight context object for a given app's package name.
    - Apps built from the same developer use this feature.
    - Malicious app can use the class loader from the returned Context object to execute another app's code.
    - For this attack to work you need to disable the security checking by passing

CONTEXT_INCLUDE_CODE and

CONTEXT_IGNORE_SECURITY flags.

3. Hiding malicious behavior
    - Malware authors use this feature to bypass Google Bouncer, an internal Google testing tool for vetting apps
    - *DexClassLoader* API is an Android API that loads classes from JAR or APK file
    - The JAR or APK files should contain a classes.dex file.
    - The JAR or APK files can be downloaded from any source including SD card and internet
    - A malicious app can use this API to dynamically load code at runtime.

VII. CONCLUSION

A lot of research work has to be done in the field of android security and to solve all the issue of data leaks I have mentioned in this paper. However there are some precaution that should be taken by both developers and users like protecting sensitive components by permission, not using application from third-party store as they are not on play store that means Google don't consider them safe. Apart from this Google is also taking certain steps by trying to be strict is term and agreement and permission policies that developers has to follow. However this is a war between security experts and attackers which has methods and code in place of nukes.

REFERENCE

[1] Erika Chin Adrienne Porter Felt Kate Greenwood David Wagner, "Analyzing Inter-Application Communication in Android" University of California, Berkeley Berkeley, CA, USA.

[2] Li Li, Alexandre Bartel, Tegawende F. Bissyand ´ e´, Jacques Klein, Yves Le Traon, Steven Arzt, Siegfried Rasthofer, Eric Bodden, Damien Octeau, Patrick McDaniel, IccTA: Detecting Inter-Component Privacy Leaks in Android Apps, Department of Computer Science University of California, Berkeley.

[3] Zhihui Han, Liang Cheng, Yang Zhang, Shuke Zeng, Yi Deng, Xiaoshan Sun, "Systematic Analysis and Detection of Misconfiguration Vulnerabilities in Android Smartphones" Trusted Computing and Information Assurance Laboratory, Institute of Software, Chinese Academy of Sciences, 978-1-4799-6513-7/14 IEEE.