A Static Detection of Inter-Component Communication Vulnerability in Android Application

Yutong Chen[1, a]

[1]Department of computer Guangdong University of Technology Guangzhou, China
[a]magnum163@163.com

Abstract. Security issues were found between inter-component communication in Android application: the usage of explicit or implicit intent might give rise to data attack, component hijacking and privilege escalation, etc. in order to detect these problems, a detect method of inter-component communication vulnerability on Android application was presented. Firstly, an APK was analyzed reversely, and ICC methods in its component were detected. Afterwards, its component call pair, component call link and component call graph were constructed. Next, the detect framework began to analyze data flow of the suspicious components. Experimental results show that the proposed method is able to detect common inter-component communication vulnerability comprehensively and effectively not only between different components but also different applications.

Keywords: Android application, static analysis, inter-component communication, vulnerability detect.

1. Introduction

With the rapid development of mobile network and communication technology, Mobile Devices, such as smartphone, become one of an indispensable communication tool in daily life. Based on the 40th Static Report on Internet Development in China, up to June 30th 2017, the number of broadband net citizens has reached 751 million, at the same time. The number of mobile net citizens in china has reached 724 million. The number of newly added mobile net citizens in the year of 2016 was 28,300,000, with an increase of 1.2 %. The proportions of people using mobile phone to surf continue to increase. The report also pointed out, in all smart terminal devices, the operating systems of the devices used by domestic mobile phone users are mainly Android, iOS, Symbian and WindowsPhone. Among them, the Android system running on smartphones has an absolute advantage, with a total of 1.93 billion, accounting for 83.02% of all smartphones. Suffice to illustrate the popularity of the Android platform for smart terminals. As the current smartphone operating system with the highest market share, the android application, like its operating system, is experiencing rapid development and also exposes a large number of security issues. For example, the leakage and theft of user data no longer only occur in a single component of the application, but the use of ICC mechanisms between components within applications and even between applications is implemented in the process of component invocation and data transfer. The main research work on ICC for Android applications is also based on static analysis or dynamic detection methods around privacy issues. The typical representative of detective ICC vulnerabilities in static analysis, epic[2], focuses on ICC, which attempts to translate the problem into an IDE problem, but there is a certain false positive because there is no data flow analysis. At present, the tools designed for the privacy leaks caused by the communication mechanism between components mainly include ICCTA [1] and amandroid [3]. ICCTA is based on the java static analysis framework soot, which first converts the dalvik bytecode into a kind called jimple. The intermediate code, then ICC chain extraction, and modify the jimple to directly connect the components, so that the tool can carry out data flow analysis between components, through the use of improved high-precision component stain analysis tool flowdroid, establish the entire android application complete Control flow graph.

2. ICC Overview

Components are the basic unit that constitutes an android application, including Activity, Service, Broadcast receiver and content provider. Activity is used to display the visual interface for user operations; Service can perform operations in the background without displaying user interface; content provider is used to save and retrieve data to achieve data sharing between programs; Broadcast receiver is for applications or operating systems. The sent broadcast is filtered to accept and respond. There are two ways for the android system and its application to send intent objects: (1) if the name of the target component is specified in the intent so that it can be sent directly to the receiver component, which is explicit call; (2) The implicit call is that the intent sent does not specify the target component that receives the intent. Only the action, data, and category attributes of the intent object are set. The system looks up the intent filter of each component and filters out the application that meets these conditions. Using this implicit intent is one of the powerful features of the android system, but it also brings many security risks.

3. ICC Security Analysis

Before you begin to format your paper, first write and save the content as a separate text file. Complete all content and organizational editing before formatting. Please note sections A-D below for more information on proofreading, spelling and grammar.

Due to the characteristics of the communication mechanism between android components, the communication between components within the application and even between applications is prone to security problems caused by other unexpected applications intentionally or accidentally intercepting, forging or even tampering with the intent in the communication process to raise communication security problems. Including the following categories: (1) Privilege escalation: The application does not have permission to perform certain operations, but it can achieve the purpose of the operation by calling other applications that have obtained the privilege and are not using access restrictions. (2) Data leakage: After the component acquires the data stored in the intent, it directly forwards the data or stores it in a new intent and sends it to another component to perform the operation of sending data outward, such as outputting to the log and writing. Incoming and sending text messages, transmitting over the network, etc. (3) broadcasting Eavesdropping: The component sends a broadcast to the broadcast receiver inside the application, but is also accidentally captured by other receivers outside the application; (4) Data attack: Data attacks generally occur in the communication process between applications. The attacker constructs and sends some targeted intents to external components. Usually these external components will parse the extras value and directly in the component. Use them, or indirectly forward to other internal components to operate, if the component does not perform security checks or exception handling after obtaining the data, it will lead to denial of service attacks or data tampering attacks;

The following is an example of a component hijacking security problem. The example consists of three applications. Each application has only one activity component. App1 is the application responsible for obtaining the IMEI number of the device. It does not implement the ability to send text messages by itself, so it needs to call app2 that does this. Then app1 sends the device number to app2 by implicit intent call. Since the target component is not explicitly specified, the app3 can also receive the intent send by the app1 and be selected by the user. After receive the intent, the app get the carried data and tampers it and then forwards it to the app2 using the explicit intent. App2 unwittingly displays the modified data in the textview, at which point component hijacking occurs.

4. Detection Method Implementation

Aiming at the vulnerabilities in the communication process between android application components, a detection method is implemented based on static analysis. The overall framework of

the method is shown in the figure 1, including preprocessing module, detection module and data flow analysis module.

4.1 Pretreatment

1）Reverse analysis. This phase includes two steps: mainfest file parsing and code decompilation analysis. The first step is to decompile the apk and get the manifest.xml file and extract the relevant content by using AXMLprint2.jar. In addition to obtaining the package name and its basic component information in the manifest file, it also includes application security-related attributes, including the application's permission and use-permission attributes, the component's exported attribute, and the intent-filter attribute. The second step of the work is to decompile the APK file, get the dex file in the application, use the dare[3] tool to decompile the android application into java source code, and extract the data related to intent, ICC methods of the application component based on the code. All of these are created and stored in the corresponding tables.

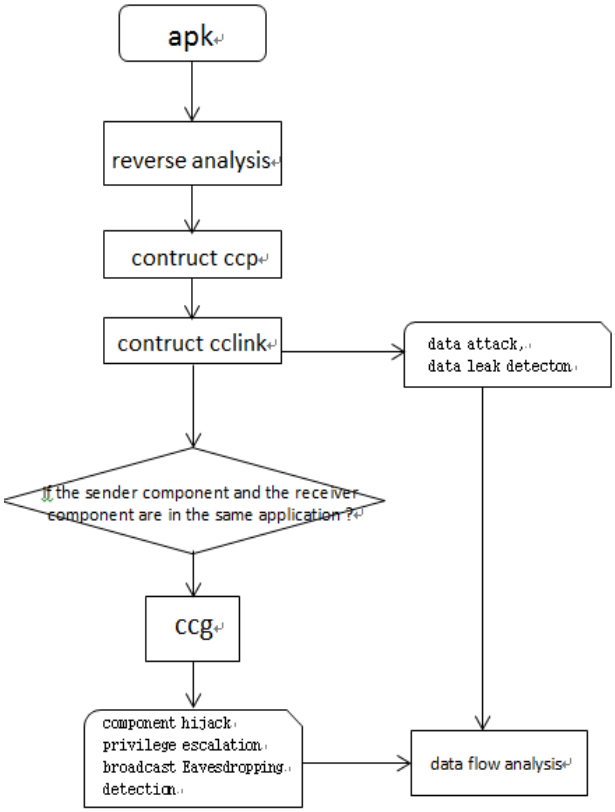

Fig. 1 Detection framework

2）CCP construction. The CCP means the component call pair, which represents the connection between a pair of components. A set of CCP is behalf of the call relation of one sender component to one receiver component. The specific process of building a CCP is as follows: First, use the IC3 tool to search for the applied icc method. If there is, the CCP constructor will be input according to the search result, so that the result of matching the target component is automatically generated; For dynamic registration broadcasts, the CCP constructor does not effectively match the results, it need to find the source code and parse the first parameter in the registerreciver() method to get the target component to add it to the CCP table.

3）CCLink construction. Since the CCP can only construct a call relationship between a pair of sending components and the target component, the security issue caused by ICC may involve multiple inter-component calls. This requires further construction of CCLink using CCP to more intuitively represent calls between components relationship. The specific strategies for building CCLink are as follows: Traversing the CCP table, (1) if both sets of CCPs are found to be from the same application, and the target components of the two sets of CCPs are the same component as the sending component, they are connected by their common components; (2) if the target component of the CCP The sending component is not from the same application, but its target component is the same component as the

sending component of another group of CCPs, and is also connected by their common components; (3) directly generating the CCP that is not in the same application as the sending component CCLink;

4）CCG construction. Component communication hijacking, privilege escalation, broadcast eavesdropping and other types of communication errors between components may involve component calls between multiple applications. It is necessary to prepare for the detection of these vulnerabilities by constructing a CCG (component call graph). The CCG is represented by a matrix, and each element in the matrix represents the case where the column (sender component) calls the row (receiver component). The specific strategies for constructing CCG are as follows: (1) traversing the CCLink table, finding out the components that send the broadcast, and all the components that can receive their broadcast messages (especially the receiving components that are not in the same application as the sending broadcast component) to establish the CCG; (2) traversing the CCLink table, and constructing the CCG directly by CCLink involving three applications;

4.2 Detection Method

1）Component hijacking detection. Firstly, the CCG is traversed to obtain the calling relationship between all components in the graph, and the following rules are used to determine whether there is a potential component hijacking problem: if there are two or more call paths between a sender component and a receiver component, that is, the sender component indirectly calls the receiver component by calling another component in addition to directly calling the receiving component, then determines that there is a potential component hijacking problem with media component. The control flow graph construction in the component is performed on the relevant components detecting the suspected component hijacking problem, and the ICC method of each transmitting component is connected with the entrance method of the target component according to the generated CCG to form a complete control flow graph.

2）Privilege escalation detection. Firstly, according to the CCG analysis, the calling relation between the sending component and the receiving component determines whether the call is directly or indirectly; if it is an indirect call, find the <permission> and <uses-permission> attributes set by the receiving component, if Without <permission>, there is no privilege escalation vulnerability, it is no need to detect further; if there is <permission>, check if the sender component has the corresponding <uses-permission>, and if so, it is determined that there is no privilege escalation vulnerability; If the component is not set that, it need to further determine whether the permission of the media component <uses-permission> is related to the <permission> corresponding to the receiving component, and then check if it has the <permission> attribute. If there is no permission promotion problem, vice versa. There is a privilege escalation issue.

3）Broadcast Eavesdropping (Denial of Service) Detection. The occurrence of the broadcast eavesdropping or denial of service indicates that the broadcast receiver of the application other than the application sending the broadcast intercepts the broadcast message, and the sent broadcast message does not have the receiving permission, so it is necessary to analyze that the broadcast receiver from other apps which can be received broadcast without the access restriction. For the sender component with data transmission, it is necessary to further detect whether the receiving component has obtained data through the getxxxextra() method, and compare the key values of the two, if equal, it is considered that there is data leakage caused by broadcast eavesdropping; For the detection of denial of service, it is necessary to check whether the sending component has called the sendorderedbroadcast() method and the priority values of all broadcast receivers of the broadcast receiver, and whether the termination broadcast method is called.

4）Data Attack and Data Leak Detection. Data leakage can occur between different components of the same application, or between different applications, data interaction is passed through the intent; data attacks generally occur between the public and private components of the application. The detection method is as follows: first traverse CClink, find the first component that has API which can get private data and the ICC method, and judge whether the CClink starts from the component and whether all the components involved are in the same application. If it is, the control flow graph is

constructed; the api of the privacy sending and the api of the private data acquisition are searched on the control flow graph, and the taint mark and propogation are performed; if they are not in the same application, it is necessary to check whether the receiver component has the access permission, and the corresponding use-permission of the sender component, and if the sender component can call receiver component, the above steps are performed.

4.3 Data Flow Analysis

1）CFG construction. The component control flow graph CFG is generated for each component that detects potential ICC security problems to be analyzed. It is a directed graph. Each node in the graph represents a code block, and the jump between the code blocks is control. The directed edge of the flow graph point to the execution direction of the program. This requires connecting the calling component to the called component based on the previously generated CClink. Here is actually the ICC method inside the component is connected to the entry method of the target component.

2）taint analysis. A taint propagation is performed on the generated control flow graph. Firstly, the data generated from the source method called by the component in the CClink starts to spread the taint. If it is detected that the taint data can reach the ICC method of the component, the key value storing the taint data is recorded. When performing taint data propagation between components, it is compared whether the key value of intent between the sender component and the receiver component is equal, and the taint propagation is continued if they are equal. If the taint data eventually flows into the parameters of the sink method, then find a path from the source method to the sink method, it can determine the path of privacy leakage or data attack.

5. Experiments and Analysis

In this paper, 60 applications below 5M randomly downloaded from the Android market are detected and recorded. The results of the ICC method call and the intent data acquisition method and other detection systems are shown in the following two tables. The number of explicit calls to the ICC method in all applications under test is 196, and the number of implicit calls is 384; the number of unknown calls in the implicit call is 214, and the number of data transfers is 108. it is mainly to analyze whether the sender component implicitly calls the intent, and does not combine the receiver component to analysis.In general, the scope of this article is more comprehensive.

Table 1 Detect result and comparison of detection tool

type	Detection tool			
	KMDroid[4]	CHEX[5]	[6]	This system
component hijacking	N	Y	N	Y
privilege escalation	N	N	N	Y
Broadcast Eavesdropping	N	Y	N	Y
Data Attack	Y	N	Y	Y
Data Leak	N	Y	Y	Y

6. Conclusion

Aiming at the security problems such as component hijacking, privilege escalation and broadcast eavesdropping that may be caused by inter-component communication, this paper proposes an android application detection method based on static analysis. By connecting the sender component and the receiver component, the method can find the suspicious component and judge whether it has improper behavior through the corresponding detection algorithm. The experimental results show that the method can screen out suspicious components and effectively detect ICC vulnerabilities, which improves the detection accuracy.

References

[1]. L. Li, P. Mcdaniel, A. Bartel, TF. Bissyande, J. Klein “IccTA: Detecting Inter-Component Privacy Leaks in Android Apps.” IEEE/ACM IEEE International Conference on Software Engineering, 2015, pp. 280-291.

[2]. D. Octeau, D. Luchaup, M. Dering, S. Jha, and P. McDaniel “Composite constant propagation: Application to android inter-component communication analysis” In Proceedings of the 37th International Conference on Software Engineering (ICSE), 2015.

[3]. F. Wei, S. Roy, X. Ou. “Amandroid: A Precise and General Inter-component Data Flow Analysis Framework for Security Vetting of Android Apps” Proceedings of 20th International Symposium on the Foundations of Software Engineering，New York：ACM，6．2012.

[4]. K. Wang, Q. Liu and Y. Zhang “Android inter-application communication vulnerability mining technique based on fuzzing.” Journal of University of Chinese Academy of Sciences., 2014, pp. 827-835.

[5]. L. Long, Z. Li and Z. Wu “Chex: statically vetting android apps for component hijacking vulnerabilities.” Proceedings of the 2012 ACM Conference on computer and communications Security. USA: ACM, 2012, pp. 229-240.

[6]. T. Yang, H. Cui, S. Niu and Z. Huang “Risk Analysis and Detection on Commincation with Intents in Android Applications.” Transactions of Beijing Institute of Technology, 2017, pp. 625-636.