Detectors for Intent ICC Security Vulnerability with Android IDE

Xianyong Meng, Kai Qian, Dan Lo
Department of Computer Science
Kennesaw State University
Marietta, GA, USA
{xmeng5, kqian, dlo2}@kennesaw.edu

Prabir Bhattachrya
Department of Computer Science
line 2-name of organization, acronyms acceptable
Morgan State University
prabir.bhattacharya@morgan.edu

Abstract— With the time-to-market pressures for mobile app development is increasing, its development cycle is getting shorter and developers have little to no time for security remediation. Many mobile app developers overlook the security quality of the software in the development cycle. Mobile app flaws and security defects could open doors for hackers to easily attack mobile apps. Early elimination of possible security vulnerability will help to secure our software, and mitigate the security risk threats from potential malicious attacking. However, many developers lack awareness of the importance of security vulnerability and the necessary secure software development knowledge and skills. An effective security vulnerability detecting tools integrated with IDE would be very beneficial for software developers. In this paper we explore the Android common ICC vulnerability and present ICC intent flaw detectors with open source FindSecurityBugs integrated in Android Studio IDE.

Keywords— Android vulnerabilit; secure software development; static analysis; FindSecurityBugs

I. Introduction

The mobile security threats are growing explosively with the wide spreads of mobile applications in our daily life. Most of malicious mobile attacks take advantage of vulnerabilities in mobile applications, which should be eliminated in the mobile software development phase. However, most development teams often have little or no time for security remediation due to the project deadlines. Even worse, many development professionals lack awareness of the importance of security vulnerability and the necessary secure software development knowledge and skills at the development stage. Security vulnerabilities open the doors to security threats and attacks, which may be prevented at early stage. The secure mobile software development is an important and integral part of mobile application development instead of an add-on component only.

The collaboration and Communication is needed between these components. In addition, most Android apps need to reuse components within same app or components in external apps for collaboration. Nobody is capable to develop any app from scratch without any component collaboration. The Inter-Component Communication (ICC) supports communication between different types of android components and passes data between components. ICC provides such capacity to facilitate the component collaboration but such collaboration may have security vulnerability, which may compromise user security privacy.

Intent eavesdropping is a common intent attacks where a component sends an implicit intent to another component within same app or in an external app but somehow the intent is intercepted by an unauthorized malicious receipt because any component with matching Intent Filter can intercept the Intent. It is caused by either using unnecessary implicit intent for intra-app or intent exposure without the signature permission protection for inter-app. A common Intent eavesdropping can attack a vulnerable broadcast intent and intercept all sensitive data or lead victim to a malicious target for exploitation. Vulnerable activity and service Intents may also suffer from eavesdropping attacks where the malicious component intercepts an activity launching or service binding request and starts its own activity or service instead. This attack will result in user data breaching and application exploitation. An intent eavesdropping case is described in Fig. 1 where the malicious app may tamp the data in turn and then transfer the data to a victim.

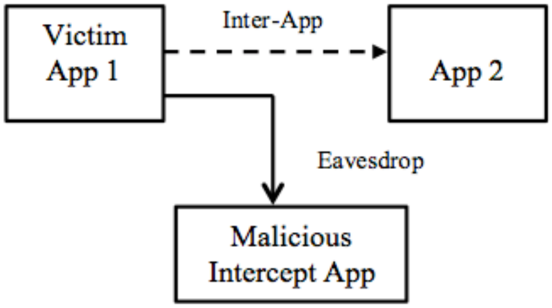


Fig. 1. Implicit Intent Interception

Intent Spoofing is another form of intent flaws in Android ICC. A malicious app can inject malicious messages and lead to sensitive data leakage and cause security threats. If a developer carelessly leaves an internal component exposed to external applications with implicit intent, an installed malicious app can spoof this vulnerable component in that app. The intent spoofing may attack all kinds of components such as activity,

978-1-5386-4646-5/18/$31.00 ©2018 IEEE

service, and broadcast receiver to exploit victim's system. A spoofing injected vulnerable activity may start an unintended activity or bind a malicious service. The most common intent spoofing case is that a vulnerable broadcast receiver suffers from broadcast injection with malicious data or code from a malicious spoofed broadcast. An intent spoofing case is shown in Fig. 2.

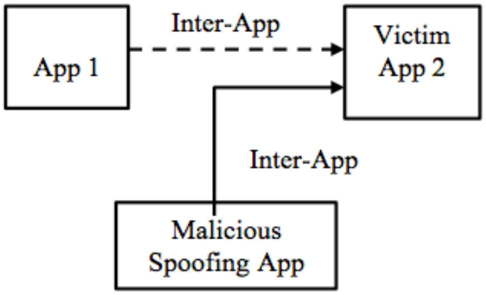


Fig. 2. Implicit Intent Spoofing

The mobile application flaws and security defects could open doors for hackers to break into the app, access sensitive information, and conduct all kinds of malicious attacks. Most vulnerability should be addressed and fixed in the mobile software development phase. If all the mobile apps are secure or have less security flaws and vulnerabilities, the security threat risks will be greatly reduced. Computer users, managers, and developers agree that we need software and systems that are "more secure". Such efforts require support from both the education and training communities to improve software assurance, particularly in writing secure code. There are many open source static Java code analysis tool that helps developers to maintain and clean up the code through the analysis performed without actually executing the code such as Eclipse IDE, IntelliJ IDE, Find Bugs Plugin. These tools focus on finding probable bugs such as inconsistencies, helping improve the code structure, conform your code to guidelines, and provide quick-fix. In general, SCA tools are used to ensure code quality from the very beginning and to make software development more productive. The security vulnerability checking is not their major task. Source code analysis tools, also referred to as Static Application Security Testing (SAST) Tools, are designed to analyze source code and to help to find security flaws with a high confidence that what's found is indeed a flaw. However, there is no tool that can automatically find all flaws and also can guarantee all detecting are positive and never miss any potential flaws.

FindSecurityBugs (FSB) is a static code analysis plugin for the FindBugs(a plugin for IntelliJ API)[3]. It specializes in finding security issues in Java code by searching for security. It can be used to scan Java, Android, and Scala applications. Since it analyzes code at the bytecode level to find defects and/or suspicious code, source code is not mandatory for the analysis. It helps to prevent potential security flaws from released software. Moreover, it allows us to design our own custom flaw detectors to find and report the emerging security threats such that many flaws can be detected during the software development phase instead of finding vulnerabilities much later in the development cycle. Moreover as security threats and are changing and vulnerability detections must also be updated. FSB allows developer to design custom security vulnerability detectors.

In this paper we present Android integrated vulnerability detector development based on current OWASP top 10 mobile risks to increase the security vulnerability check coverage [5].

II. Related Works

Many efforts have been made to enhance the secure software development education in recent years. UNCC [1] has designed and developed an Application Security IDE (ASIDE) plug-in for Eclipse that warns programmers of potential vulnerabilities in their code which addresses input validation vulnerabilities, output encoding, authentication and authorization, and several race condition vulnerabilities. ASIDE only works in the Java Eclipse IDE and cannot support the Android IDE. David Kantilla, Erika Chin [2, 4] have identified Android intra-app and inter-app vulnerability and analyzed the security threats. They proposed to secure the Android ICC system. David Foramen and others [3] have analyzed common software vulnerability such as buffer overflow and SQL injection and developed vulnerability detectors with FindSecurityBugs. All these works either mainly focus on the Java secure software application development without addressing secure mobile software development or focus on the mobile vulnerability analysis without platform implementation. We have explored the mobile risks posted by OWASP and developed Android Studio integrated ICC vulnerability SCA detectors with FindSecurityBugs API in Android Studio platform to enhance SMSD. In next section we will introduce our ICC detectors for intent interception and intent spoofing in Android secure mobile software development.

III. Secure Mobile Software Development with INTENT Vulnerability Detectors

A. FindSecurityBugs Detectors for Secure Android Software Development

To meet the ever-increasing demand for high quality information security professionals with expertise in SMSD, we built innovative Android vulnerability detectors with an open source FindSecurityBugs API plugin for the popular Android Studio IDE based on the on most current OWASP 2017 mobile top 10 mobile security risks [6,7] in the category of SQL injection, unintended data leakage, insecure communication, insecure data storage vulnerability detectors. For example, the built detectors can recognize a vulnerability of SQL injection and data leakage in an Android mobile application program, which may face the threat of potential malicious code injection, and then issue a warning on the code line. Following the provided options, students or developers can enforce a new secure statement to replace the unsecure statement.

The built package can be loaded into the Android Studio IDE, parse Android java source code, identify specific API calls, warn the potential vulnerabilities, recommend security solutions, and replace code statements. For example, it can recognize a vulnerability of SQL injection and data leakage in an Android mobile application program, which may face the threat of potential malicious code injection, and then issue a warning on the code line. Once the developer clicks the warning icon, secure coding prevention and protection options will be shown in Android Studio. Following the provided options, students or developers can enforce a new secure statement to replace the unsecure statement.

The intent ICC security vulnerably detectors are implemented with Open source FindSecurityBugs API based the top 10 OWASP mobile risks.

B. *Vulnerable Broadcast Intent Detectors*

In Android, there are four types of app components: activities, services, content providers and broadcast receivers. The primary method for inter-component communication, both within and between applications, is via intents. The intent is the main communication mechanism between Android components such as activity, service, and broadcast receiver in intra-app and inter-app. An intent object is a messaging object (an instance of the android.content.Intent class) which can be used to start activities and services, bind to services, and convey notify and pass data to broadcast receivers. There two types of intent: Explicit and implicit intent.

The security vulnerability exists in intent broadcast ICC between Android components.

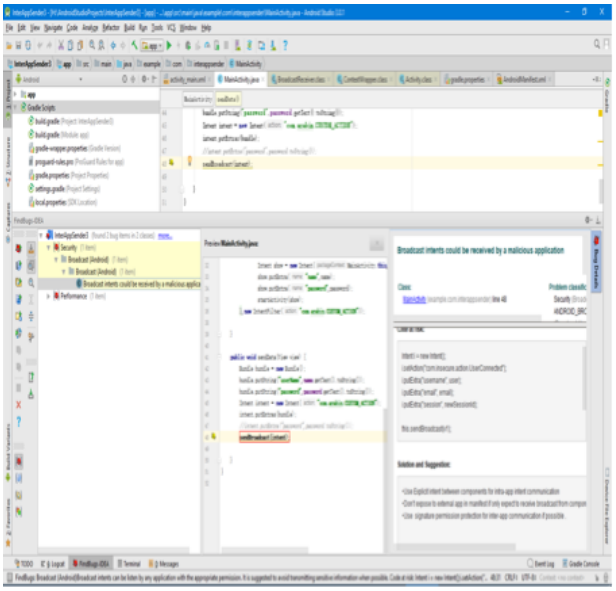
The sendBroadcast method is used to send an intent to other component. A vulnerable code fragment is shown below:

```
public void onClick(View view) {
Intent intent = new Intent();
intent.putExtra("number", 1);
intent.addFlags(Intent.FLAG_INCLUDE_STOPPED_PACKAGE
S);
intent.setComponent(new
ComponentName("example.com broadcastreceiver","exa
mple.com broadcastreceiver.MyBroadCastReceiver")
);
intent.setAction("com example.MyBroadcast");
    sendBroadcast(intent); }
```

Fig. 3 shows a screen shot of diagnosis result by intent vulnerability detector for implicit intent interception. The alert message and suggested solutions are also displayed as follows.

- Use Explicit intent between components for intra-app intent communication
- Don’t expose to external app in manifest if only expect to receive broadcast from components within same app

Use signature permission protection for inter-app communication if possible.



Broadcast intents could be received by a malicious application

Class:

MainActivity (example.com.interappsender) line 48

Problem classific

Security (Broad

ANDROID_BRC

Code at risk:

```
Intent i = new Intent();
i.setAction("com.insecure.action.UserConnected");
i.putExtra("username", user);
i.putExtra("email", email);
i.putExtra("session", newSessionId);

this.sendBroadcast(i);
```

Solution and Suggestion:

- Use Explicit intent between components for intra-app intent communication
- Don't expose to external app in manifest if only expect to receive broadcast from compon
- Use signature permission protection for inter-app communication if possible.

Fig. 3. Detecting sendBroadcast family of methods

IV. Conclusion

We designed and developed mobile ICC vulnerability detectors which are essential in building capacity to secure mobile application development. Our project with Android Studio FindSecurityBugs plugin will help mobile professionals apply custom Android vulnerability detectors in SMSD.

Acknowledgment

The work is partially supported by the National Science Foundation (NSF) under award: 1723578, 1639995, 1623724.

References

[1] M. Whitney, H. Richter Lipford, B. Chu, and T. Thomas. "Embedding Secure Coding Instruction into the IDE: Complementing Early and Intermediate CS Courses with ESIDE" In press, Journal of Educational Computing Research, 2017

[2] B. Fisseha Demissie, D. Ghio, M. Ceccato, A. AvanciniIdentifying Android Inter-app Communication Vulnerabilities Using Static and Dynamic Analysis, IEEE/ACM International Conference on Mobile Software Engineering and Systems (MOBILESoft), 2016

[3] P. Arteau, D. Formánek, FindSecurityBugs, https://github.com/find-sec-bugs/find-sec-bugs

[4] E. Chin, A. Porter Felt, K. Greenwood, D Wagner, Analyzing inter-application communication in Android, MobiSys '11 Proceedings of the 9th international conference on Mobile systems, applications, and services, Pages 239-252

[5] Projects/OWASP Mobile Security Project - Top Ten Mobile Risks, 2016