POSTER: Android IME Privacy Leakage Analyzer

Peng Lo*, Jia-Chi Huo*, Hsu-Chun Hsiao*, Bo Sun†, Tao Ban†, Takeshi Takahashi†
*National Taiwan university, Taipei, Taiwan
r07944039@csie.ntu.edu.tw, b04502138@ntu.edu.tw, hchsiao@csie.ntu.edu.tw
†National Institute of Information and Communications Technology, Tokyo, Japan
{bo_sun, bantao, takeshi_takahashi}@nict.go.jp

I. Introduction

An Android input method editor (IME) is a special Android application that provides a soft keyboard service. Android also provides extensible methods that enable customization of the outlook, language or font style of the keyboard. Since users use IMEs to type everything on Android devices, including sensitive information such as credit card numbers and passwords, the privacy of user input becomes a huge concern. To further investigate whether these IMEs collect user information stealthily, we build a tool called IMEAnalyzer to test and analyze the privacy leakage problem of IMEs.

Previous work [2] [1] on privacy leakage of IMEs analyzed a small number of IME samples and tested them manually. Their methods need to be improved because manual testing is slow and not scalable. Besides, we have observed several IMEs that update frequently during our experiment. It would be exhausting if manual testing is needed after every single update.

However, there are three challenges for automatically analyzing this privacy leakage problem on IMEs. First, the IME service does not have its own `ContentView`, so Android testing frameworks cannot fetch it by View API like `findViewById`. Second, we cannot simulate user input actions simply by using the keycode API provided by Android [3]. The keycode API would bypass IMEs and directly communicate with the Android operating system thus not triggering the corresponding functions registered by IMEs. Third, many IMEs encrypt their network traffic. It is hard to decrypt the packets content to check if the IMEs send user input back to the server.

In this work, we present IMEAnalyzer, a system that analyzes privacy leakage risks of IMEs. Our contributions can be summarized as follows. First, we build a system to conduct various test cases and record the reactions of the Android third-party IMEs, in a fully automated fashion. Second, we propose our analysis methods to identify the existence of suspicious or data collection activity by observing the logs and requests sent by the IMEs. After analyzing the logs, IMEAnalyzer will generate a list of suspicious IMEs for further research.

II. Background and Threat Model

A. Android Input Method Editor

An IME is a soft keyboard used in Android devices. Android provides an extensible input method framework for apps to offer their users with alternative IMEs, which are often called third-party keyboard apps.

Third-party IMEs, as opposed to default system keyboards, are thriving with a reason. They often have features that default keyboards fail to offer, such as support for cool emojis or various themes. Thus, third-party IMEs tend to grant more permission. However, once a certain permission is granted, a bundle of functionalities required that permission would be available to the IME. It turns out to be more disastrous of an IME to abuse its privilege than of other Android apps, since the IME could record everything of the user input.

B. Threat Model

In this paper, we assume that third-party IME developers could be malicious. The attacker's goal is to get personal information of users. The attack model can be divided into two kinds: keylogging and other privacy leakage. Keylogging is to get all the characters user type by an IME. Other privacy leakage consists of collecting the information which is authorized by the permissions required by the IMEs, such as contact information, GPS, or SMS. Since a long line of work has discussed about the privacy leakage problem of Android apps, we mainly focus on the keylogging problem.

According to what kind of information that a malicious IME sends back to its server, we classify keystroke keylogging into two scenarios: Sending all user input, and sending sensitive data only. Besides, some IMEs only send back the statistical data, for instance, the frequency of a specific words appear during user typing. Since many apps claim that they are collecting statistical information for improving the quality of user experience and all of the collected information are differentially private, we consider it out of scope.

1) Sending all user input: The attacker sends all user input back to the server.

2) Sending sensitive data only: The attacker first analyzes the contents, removes useless words, and then only sends sensitive data back. There are two methods the attacker can recognize sensitive data. First, the attacker uses customize keywords to filter the content, such as bank's name, address or diseases. Whenever the victim types in a keyword, the attacker sends back the whole paragraph. Second, the attacker inspects the `inputType` of `EditText` field in Android app to identify and send back the user input, such as password, credit card number or phone number.

III. System Design and Implementation

IMEAnalyzer includes three components, a testing server, a client application on the mobile, and two loggers. The analyzer automatically executes the steps below, and the user should provide package names of IMEs which are subjected to test.

A. Environment Setup

Before the test starts, IMEAnalyzer sets up two loggers to collect the information during experiments. One is a system logger, which contains all the information from Android Debug Bridge (ADB) logcat. The other is a network traffic logger, which records network packets by a Mitmproxy server set up in the same network.

B. Keyboard Layout Reconstruction

As mentioned in Section 1, IME service lacks user interface to interact with, and this is also one of the major problems that make it difficult to automate the process of analyzing IMEs. We solve this problem by getting a key map before the analysis starts. IMEAnalyzer obtains the keyboard layout of an IME by running a script on the testing server with the client app installed. The client app provides an input field and records the input characters. The script simulates the key touch function by ADB and finally outputs the coordinate of each character on the keyboard.

C. Testing Process

With the keyboard layout we gathered in Section 3.2, IMEAnalyzer can easily find the coordinates of characters and trigger them. This helps overcome the second challenge – cannot simulate the actions simply by the keycode API [3]. To interact with IMEs, IMEAnalyzer does not send keycode to Android OS directly, but instead taps the IME service interface to trigger the IME to generate keycodes. After the testing process, it will output the pcap file generated by our Mitmproxy module which contains the full packet records of the requests during user typing, such as timestamp, request URL and other detailed fields.

D. Analysis

In the analysis phase, the system takes the datasets from the testing process and analyzes them with corresponding user input data. We provide following functions and separate them into parts by the different scenarios in our threat model:

1) Sending all user input: In this scenario, we design several test modes to distinguish whether and when the attacker sends packets under different input behaviors.

- Normal typing: We execute typing test described in Section 3.3 to examine the IME service solely.
- Not typing: We first open the client app and make the keyboard visible. Then we wait for 15 seconds without typing anything.
- Typing in fixed frequency: We type words in fixed frequency, e.g. type one word, wait for 15 seconds and repeat. Therefore, by looking into the timestamp of packets and typing, we can observe the relation between typing word and packet traffic.

2) Sending sensitive data only: In the scenario, we assume that the attacker only sends data back to server while sensitive words appear. There are two kinds of methods to detect sensitive data, which can be mapped to distinct test modes.

- Typing keywords: we hypothesize that some attackers may use keywords to detect whether users are typing sensitive information. In this test mode, we will execute the typing test using given sensitive words.
- Typing in specific `inputType`: `inputType` is an attribute of Android `EditText`, which is used to indicate different types of user input such as password, email, or phone number. We implement an app that contains 10 kinds of `inputType` (`none`, `text`, `textEmailAddress`, `textPassword`, `textVisiblePassword`, `textPostalAddress`, `textUri`, `textPersonName`, `phone`, and `number`) to observe the result of typing in different `inputType` of text input fields.

E. Result

We compared the results of "Normal typing" and "Not typing" modes to reveal the difference between whether or not the IME service is used, in terms of number of packets transmitted. We classified the results into three categories:

1) Probably innocent: Nothing was sent to the Internet during our experiment.
2) Same Behavior in both situations: The IME apps do send packets to the Internet during our testing process, but the number and content of the packets, as well as the quantities of POST and GET requests are all identical whether or not the IME service is used.
3) Suspicious: IME apps in this category sent more packets to the Internet while the IME service is used to input text.

The purpose of this process is to distinguish suspicious IMEs from innocent ones and generate a list for further examination.

IV. Conclusion and Future Work

In this work, we build the first automated analyzer tailored for Android third-party IMEs, thereby accelerating the testing process. Future work includes (1) in-depth evaluation of suspicious MEs, (2) digging into packet contents for further investigation, and (3) extending to system log analysis.

References

[1] J. Chen, H. Chen, E. Bauman, Z. Lin, B. Zang, and H. Guan, "You shouldn't collect my secrets: Thwarting sensitive keystroke leakage in mobile ime apps," in *24th USENIX Security Symposium (USENIX Security 15)*, 2015, pp. 657–690.

[2] J. Cho, G. Cho, and H. Kim, "Keyboard or keylogger?: A security analysis of third-party keyboards on android," in *2015 13th Annual Conference on Privacy, Security and Trust (PST)*. IEEE, 2015, pp. 173–176.

[3] Keyevent — android developers. [Online]. Available: https://developer.android.com/reference/android/view/KeyEvent