DroidTrack: Tracking Information Diffusion and Preventing Information Leakage on Android

Syunya Sakamoto, Kenji Okuda, Ryo Nakatsuka, Toshihiro Yamauchi

Graduate School of Natural Science and Technology, Okayama University, Japan
sakamoto@swlab.cs.okayama-u.ac.jp,yamauchi@cs.okayama-u.ac.jp

Abstract. An app in Android can collaborate with other apps and control personal information by using the Intent or user's allowing of permission. However, users cannot detect when they communicate. Therefore, users might not be aware information leakage if app is malware. This paper proposes DroidTrack, a method for tracking the diffusion of personal information and preventing its leakage on an Android device. DroidTrack alerts the user of the possibility of information leakage when an app uses APIs to communicate with outside. These alerts are triggered only if the app has already called APIs to collect personal information. Users are given the option to refuse the execution of the API if it is not appropriate. Further, by illustrating how their personal data is diffused, users can have the necessary information to help them decide whether the API use is appropriate.

Keywords: Android, Malware, Preventing Information Leakage, API Control,

1 Introduction

In recent years, adoption of the smartphone has been rapidly spreading, and Android [1] is one of the popular operating systems (OS) for smartphones. An app developer can make the app available through a Web site, such as Google Play Store [2]. However, an app [3] can hijack administrative privileges in order to exploit vulnerability in the Android OS and send out illegally collected personal information.

Malware that target the Android OS are usually intended to illegally collect personal information. A mobile device contains a large amount of personal information, such as name, address, phone number, etc. and their information can be easily obtained by apps using the Android API. In addition, many users are unaware that mobile phones are not secure and usually do not come with any anti-malware software. For this reason, there is a possibility of information leakage while user did not notice the infection of malware.

An Android app is executed in sandbox, and communication with other apps is severely restricted, except using Intent [4]. Key features such as external communications and the acquisition of personal information require permissions from the user. However, the user cannot detect when the personal information is obtained by the app and whether that personal information was leaked.

In this paper, we propose DroidTrack: a method for tracking information leakage diffusion and preventing information leakage on Android, tracks information diffusion after the app has obtained personal information. DroidTrack alerts the user if there is a possibility of information leakage, and allows the user to limit the use of the API. DroidTrack monitors any app that uses the information-gathering API and displays a warning when the app also uses the API that sends information outside of the device. Personal information can also be leaked when one app obtains personal information and then sends it to another app, which sends the information out of the device. For this reason, DroidTrack manages both apps using the Intent. In addition, the user is allowed to decide whether to limit the use of the API, which sends information out of the device, thereby preventing information leakage.

2 Android component and security issues

2.1 Android component

In the Android OS, all apps operate on the Applications layer. If an app requires resources, it must use the API provided by the Application Framework. Android apps have individual user IDs (UID), and communication with an app with a different UID is highly restricted, except when using the Intent.

Apps cannot use the Android API to access a protected resource or to gather personal information without user's permission. It is necessary to obtain the user's permission [5] for app to use these APIs. An app can request specific permissions, e.g., permission to connect to the Internet (INTERNET) or permission to read the status of the unit (READ_PHONE_STATE). The safety of the resources is preserved by granting only minimum permissions required by the app.

Each Android app runs in sandbox. By default, apps cannot communicate with another app because they are strictly separated from each other. However, it is possible to enable communication between apps by using the Intent, which allows an app to communicate with another app and receive the results of process. In addition, an app can pass data both as a string or as an object.

2.2 Security issues in Android

The following are the problems associated with malware and malware infections in the main security areas in the Android OS:

(1) Problems obtaining administrator authority
(2) Problems with development tools (Android Debug Bridge (ADB) [6])
(3) Permission abuse
(4) Difficult detection of information leakage
(5) WebKit abuse

Problems (1), (2), and (3) are cause of the infection to malware. Problems (4) and (5) are related to malware behavior. Problem (4) makes it very difficult to inform the

user when the app gathers information and what kind of personal information it gathers. Therefore, if a user installs malware by mistake, user cannot detect the leakage of personal information. In this work, we deal with problem (4) by preventing the transmission of information out of a smartphone.

3 Design principles of the proposed method

3.1 Requirements and challenges

In order to deal with the problem, we propose the following requirements:

(1) Detect all APIs with the possibility of information leakage.
(2) User can judge the risk of information leakage.
(3) Information leakage can be prevented by disallowing the execution of API.

In order to satisfy these three requirements, we propose the following challenges:

(A) Clarifying the condition of information leakage
(B) Detecting all uses of APIs that have a possibility of information leakage
(C) Controlling the operations of apps that have a possibility of information leakage
(D) Allowing the user to decide whether app can receive sensitive information

3.2 Solution

3.2.1 Solution for challenge (A)

Information leakage can occur when an app uses the Android API to send information to out of the device (diffuse information) after obtaining personal information. The app can also obtain personal information without using the information-gathering API by using the Intent instead. In another scenario of information leakage, one app uses the information-gathering API and then communicates with another app that uses the information-diffusing API.

In this work, we address the following scenarios of information leakage:

(1) A single app uses the information-gathering API and the information-diffusing API or the Intent.
(2) One app uses the information-gathering API and then communicates with another app using either the Intent or the information-diffusing API.

3.2.2 Solution for challenges (B) and (C)

As mentioned in Section 3.2.1, information leakage can occur when an app uses the API that obtains personal information, the API that diffuses the information, or the Intent. Therefore, to deal with challenges (B) and (C) (detecting all use of the API and controlling the operations of apps), we propose a method to control the behavior of the app as follows:

- by intercepting calls to the information-gathering API, information-diffusing API, or Intent,
- by determining the user's preference to control the use of API if either scenario described in Solution for challenge (A) is true, and
- by controlling the use of the APIs or the Intent based on the user's preference.

4 Method for tracking diffusion of information and preventing information leakage on Android

4.1 Design principle

To address challenges (B) and (C), we propose the following requirements:

(1) To inform the user if there is potential for information leakage.
(2) To limit the use of APIs and the Intent with accuracy based on user's preference.

4.2 Basic method

We change the framework of the Android as follows:

(1) "Hook" or intercept calls to the information-gathering API and the information-diffusing API and inform the user of both the name of the app using the API and the name of the API used.
(2) "Hook" or intercept calls to the Intent and inform the user of both the name of the app that uses the Intent and the name of the app called by the Intent.
(3) Execute a process based on the user's preference regarding the use of the APIs or the Intent.

The API is used differently depending on the type of personal information that is gathered by the app. Therefore, because of the change described in (2), the user can be informed when and what kind of personal information which the app obtains and attempts to transmit out of the device. Furthermore, the change described in (3) can be used to prevent information leakage according to the user's preferences. In the following section, we describe a method to track and prevent information diffusion by using the modified framework described above.

4.3 Control of API in the framework

Figure 1 shows the flow of control of the API in the framework. DroidTrack consists of two "Control Aps" at the Applications layer and one "Calling Control AP Unit" at the Application Framework layer.

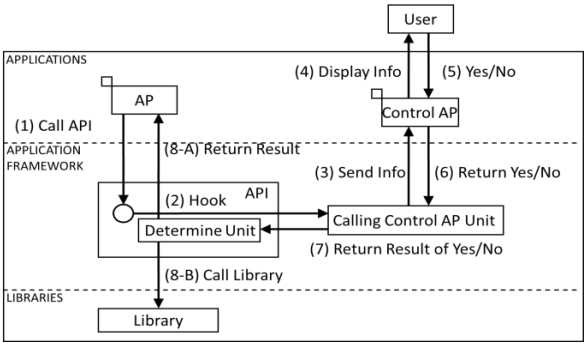

Fig. 1. Control of APIs

In Figure 1, "Control AP" is an app that provides information about the API to the user and prompts the user to choose whether to limit the use of the API. The "Calling Control AP Unit" informs the "Control AP" whenever an app calls the information diffusing APIs and transmits back to the API engine the user's preferences regarding the use of the API.

The following describes details of the flow of the process in the framework:

(1) The app "AP" calls the information-diffusing API.
(2) The "Hook" intercepts the call to the information-diffusing API in the framework.
(3) The "Calling Control AP Unit" passes the information about the intercepted call to the "Control AP."
(4) The "Control AP" displays a warning dialog to the user if it suspects that an information leak is possible.
(5) The user replies to the dialog to indicate whether user allows the use of the API.
(6) The "Control AP" forwards the user's preference to the "Calling Control AP Unit."
(7) The "Calling Control AP Unit" returns the result to the "Determine Unit."
(8) The "Determine Unit" handles API based on the user's preference, as follows:
    (A) Error handling is carried out if the user disallows the API to be called by the app.
    (B) API process returns to normal mode if the user permits the API to be called by the app.

Like the above-mentioned procedure, we satisfy the requirement 4.1-(2) by requiring user's preference before use of APIs and processing according to the preference.

4.4 Control AP

4.4.1 Basic mechanism

Figure 2 illustrates the mechanism of "Control AP." The Search-Leakage function triggers the Info Diffusion Manage Unit to check the possibility of information leak-

age. If there is a possibility, it passes the process to the Control-Write-Out function. If there is no possibility, it allows the API calls to be processed.The Info Diffusion Manage Unit updates the Information Diffusion data structure, examines the possibility of information leakage, and returns the result of the test to the Search-Leakage function. The Control-Write-Out function displays a warning dialog, and then accepts and returns the user's preference regarding the use of the APIs.

4.4.2 Information leakage determination method

Control AP monitors apps that use the information-gathering API by wrapping the app in a managed object. If the managed AP communicates with another app that uses the Intent, the second app is added to the managed object. Control AP also warns the user of the risk of information leakage if the managed app uses the information-diffusing API, thereby satisfying the requirement described in Section 4.1-(1).

5 Experiment of the operation of DroidTrack

Prevention of information leakage by apps was tested using the following procedure:

(1) Obtain the phone number of the mobile device by using "getLine1Number," which serves as the information-gathering API.
(2) Transmit the personal information out of the device by using "sendTextMessage," which serves as the information-diffusing API.

Figure 3 shows the dialog displayed by DroidTrack when the example app runs. The user can detect the use of the API by various information from the dialog. In this case, the user presses "Yes" to allow the use of the API and "No" to disallow the use of the API. DroidTrack could prevent information leakage by pressing "No."

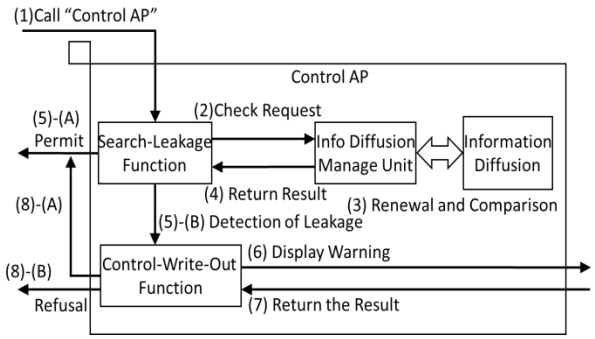

Fig. 2. Basic mechanism of Control AP

Fig. 3. Warning dialog

6 Related Work

MockDroid [7] allows the user to provide fake or “mock” data to prevent a real personal information leakage. This method needs user's set up of permissions for each apps. Therefore, DroidTrack needs no user's set up, but needs user's input only when there is a possibility of information leakage. Furthermore, DroidTrack tracks all transmitted information, including transmissions without leakage, in order to check all API communications in and out of the device. AppFence [8] that using TaintDroid [9] provides a similar approach, but it modifies framework and Dalvik VM, and uses the policy which is made except on Android. On the other hand, DroidTrack modifies only the framework of the Android. Therefore, DroidTrack is easier to implement.

7 Conclusion

We have proposed DroidTrack, a method to warn of the risk of information leakage by monitoring apps that obtain personal information and by keeping track of information diffusion. DroidTrack prevents personal information leakage by controlling the behavior of the API, based on the user's preference, which they indicate when a warning is displayed. DroidTrack can also detect information leakage in scenarios where the Intent is used and where the information-diffusing API is used. In addition, DroidTrack can inform the user of the risk of information leakage and display a list of information-gathering APIs that could have diffused information to other apps.

References

1. “Android”, http://www.android.com/
2. “Google play store”, https://play.google.com/store
3. “Droid dream”, http://blogs.computerworld.com/17929/google android market kills droid dream malware in Trojans
4. “Intent", http://developer.android.com/reference/android/content/Intent.html
5. “Access permissions”, http://developer.android.com/intl/ja/reference/android /Manifest.permission.html
6. “Android debug bridge”, http://developer.android.com/guide/developing/tools/adb.html
7. Beresford, A.R., Rice, A., Skehin, N., Sohan, R.: MockDroid: trading privacy for application functionality on smartphones. Proc. 12th Workshop on Mobile Computing Systems and Applications, pp.49-54, 2011.
8. Hornyack, P., Han, S., Jung, J., Schechter, S., and Wetherall, D.: These aren't the droids you're looking for: retrofitting android to protect data from imperious applications. Proc. 18th ACM Conference on Computer and Communications Security (CCS2011), 2011.
9. Enck, W., Gilbert, P., Chun, B., Cox, L.P., Jung, J., McDaniel, P., Sheth, A.N.: TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones. 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI'10), 2010.