A Usage-Pattern Perspective for Privacy Ranking of Android Apps

Xiaolei Li\textsuperscript{1}, Xinshu Dong\textsuperscript{2}, and Zhenkai Liang\textsuperscript{1}

\textsuperscript{1} National University of Singapore, Singapore
\{xiaolei,liangzk\}@comp.nus.edu.sg
\textsuperscript{2} Advanced Digital Sciences Center, Singapore
xinshu.dong@adsc.com.sg

\textbf{Abstract.} Android applies a permission-based model to regulate applications (apps). When users grant apps permissions to access their sensitive data, they cannot control how the apps utilize the data. Existing taint-based techniques only detect the presence of exfiltration flow for the sensitive data, but cannot detect how much sensitive data are leaked. Users need more intuitive measures to inform them which apps are going to leak more of their private information. In this paper, we take an alternative approach for identifying apps’ internal logic about how they utilize the sensitive data. We define such logic as a sequence of operations on the sensitive data, named as the data usage pattern. We build a static analysis tool to automatically extract data usage patterns from Android apps. Our evaluation shows that our approach effectively and efficiently identifies the key operations and thus ranks Android apps according to different usage patterns.

\textbf{Keywords:} Android, Privacy, Static analysis, Information flow analysis.

\section{Introduction}

The Android system relies on a permission-based model to protect sensitive resources on mobile devices. However, the existing permission-based model relies heavily on users’ perception of the permissions. A recent study shows that the Android permissions are insufficient for users to make correct security decisions \cite{6}. Users have little idea about how an application (app) would use the granted permissions. For example, to use the advertised features of an app, users may simply grant the dangerous permission to access their locations. In fact, the app may directly leak the location information to an external third-party domain, or carelessly open new interfaces for other apps to escalate their privileges to access it \cite{3}. Although several existing mechanisms have been proposed to analyze the permission usage in Android apps by detecting what and where permissions are used, they do not provide comprehensive information for users to understand how one app utilizes sensitive data after being granted permission to access. Instead, we need a solution which is both technically comprehensive and sufficiently intuitive to end users. Such a solution should help users to make wise choices to protect their privacy when they are installing new apps.

One well-explored direction in understanding the permission usage is to apply data flow analysis on Android apps \cite{2,4,7,8,13,16,17}. However, most of them only determine whether a flow to leak sensitive resources exists or not, but lack precise description...
regarding the internal data processing logic, i.e., whether the data usage leaks a lot of information or only a little. Thus, they are unable to inform users of the difference between an app that sends the raw user location to third parties, and another app that only provides a yes/no answer to whether the user is presently at a certain museum or not. Therefore, a desirable approach should deliver more insight to users regarding how their sensitive data are processed and to what extent they are leaked to other parties.

Quantitative information flow (QIF) is an emerging technique for quantifying the information leakage. Various information-theoretic metrics have been proposed, such as through one particular execution path [14] or publicly observable states [9]. Ideally, QIF could be a suitable tool to evaluate how apps use sensitive resources and how much of such information is leaked. Unfortunately, the performance and scalability of existing QIF algorithms and tools are rather limited in practice. In addition, the Android’s event-driven paradigm heavily involves asynchronous system callbacks and user interaction, which makes it even more difficult to apply existing QIF mechanisms. Considering the huge number of Android apps and their frequent updates, we need a more efficient and scalable approach.

Our Approach. In this paper, we propose a lightweight and efficient approach to ranking apps based on how they use sensitive resources. In particular, we take the location data of the mobile device as a starting point. Meanwhile, the technique is also applicable to other data types, such as the device ID and the phone number. The idea is to summarize the sequence of key operations on the location data into a data usage pattern, which represents the app’s internal logic of the location data usage. By comparing the usage patterns for different apps, we group apps with similar functionality and rank them according to their potential leakage of the location information.

Compared to existing data flow analysis techniques that only detect the presence of sensitive data flows, we focus on identifying the important operations on the sensitive data in such flows, which reflect to what extent the data are leaked. Specifically, we propose PatternRanker, which statically analyzes how an app utilizes the location data by analyzing its Dalvik bytecode, and extracts a general and comprehensive pattern representing the location data usage by identifying key operations on the location data. We collect all the possible operations by leveraging static program slicing and taint-based techniques, and then generate the data usage patterns through pre-defined heuristics. The applicability of the data usage pattern is not limited to app ranking. It can also efficiently assist further analysis, such as accelerating existing QIF solutions by applying their current mechanisms on our extracted patterns instead of on the raw logic of apps.

We evaluate PatternRanker on 100 top location-related apps, and our experiments show that PatternRanker effectively extracts the data usage pattern for ranking apps. We also achieve an average analysis time of 27s per app, which is sufficiently small for analyzing real-world apps.

To sum up, our work has the following contributions:

– We propose a lightweight and scalable approach to ranking apps’ threats to user privacy based on the usage pattern of sensitive information.
– We build a static tool to automatically analyze how Android apps utilize the sensitive data and identify the key operations.