A Fine-Grained Data Access Permission Authorization Framework for Mobile Systems

by

WENYUN DAI

DISSERTATION

Presented to the Faculty of

Pace University

in Partial Fulfillment of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY IN

COMPUTER SCIENCE

PACE UNIVERSITY

September 2017

Abstract

Smartphones play irreplaceable role in our daily life. When users enjoy the functions, convenience, and efficiency offered by mobile apps, their sensitive and private data are facing potential privacy hazard to be leaked. These data could be shared be among various apps installed on the same smartphone, or even be sent to third-parties without any notification. Furthermore, the permission authorization provided by current mobile operating systems is just coarse-grained. First, the permission authority process is an one-time operation. After authorized of accessing to some data, one app can hold this permission forever theoretically. Then, this process only works on the whole data, not any specific portion of data. In other words, users only can authorize one app to access all data or none.

This study focuses on a new framework to provide fine-grained data access permission authorization for mobile systems. The study consist of three main layers: a) Separating mobile data with different privacy level and applying a finer-grained data request in local mobile system; b) Mobile data request and access management framework as the middle layer; c) high performance and efficient cloud storage strategy. We also design and implement a benchmark with some candidate apps to prove the feasibility and performance of our proposed framework.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Smartphones play irreplaceable role in our daily life. They can be used as not only communication devices, but also portable computers. It is convenient for users to store data into their smartphones and use them anywhere anytime afterward. However, sensitive and private data are facing potential privacy hazard to be leaked, such as bank account number and password, photos with commercial confidential, business cooperations' contact information. Researcher have been studying mobile phone security for several years [1, 2, 3], especially malware [4, 5]. Security problems are happened more frequently in non-iOS systems than in iOS systems. The locked development environment of XCode makes iOS apps no access to other permissions, and the diligent scrutiny of App Store makes iOS apps cannot or extremely hard to handle personal information and company data. Meanwhile, during these years, large amounts of security and antivirus apps are developed, such as 360, Avast, ESET and Avira.

Furthermore, some researchers presented a few excellent solutions to deal with malwares in Android, such as SmartSiren [6]. With their help, the chance of malware to survive and attack smartphones is remarkable reduced. As mentioned in [7], although malware created to compromise device security or data is portrayed as the principal villain in the mobile application security narrative, malware is not the primary threat to user privacy or enterprise security. Furthermore, they

showed that mobile malware infects only 0.4% of mobile apps in the enterprise and 0% found in the Top 400 [8]. As a result, malware on smartphones is a problem, but it is not the most important issue security and privacy issue on smartphones.

At present, the most serious potential privacy hazard is that apps collect data more than enough on its original function while in permission scope, which we call it data over-collection[9]. We survey current solutions to solve data over-collection problem and find that almost all these approaches are passive defense measures, which are a remedy approach after being hurt. Furthermore, these solutions need to run tools or other apps to monitor and detect the data over-collection behaviors, which consumes more energy of smartphones [10].

The common data over-collection behaviors include tracking location, accessing photos, address book, and calendar, and tracking IMEI/UDID (International Mobile Station Equipment Identity/Unique Device Identifier). It will do much greater harm if these data over-collection behaviors work together [11]. For example, pictures with location information collected by different apps could be classified with the same IMEI/UDID, which indicates that these data are from the same device, even the same user. This makes leaked data with more practical significance and closer to the real life.

The original sin of data over-collection behaviors is the permission authorization provided by current mobile operating systems, which is just coarse-grained. First, the permission authority process is an one-time operation. After authorized of accessing to some data, one app can hold this permission forever theoretically. Then, this process only works on the whole data, not any specific portion of data. In other words, users only can authorize one app to access all data or none. For example, user wants to post one of his/her photos via Facebook app. He/She has to authorize Facebook app the full permission to access all his/her photos including the private ones. It is obvious that this kind of permission is too excessive and should be detailed. However, from the view of apps, it is too complicated to figure out which piece of data user allows an app to access he/she really selects the one.

The next section describes some common data over-collection behaviors and the main motivations.

1.2 Data Over-Collection Behaviors Analysis and Motivations

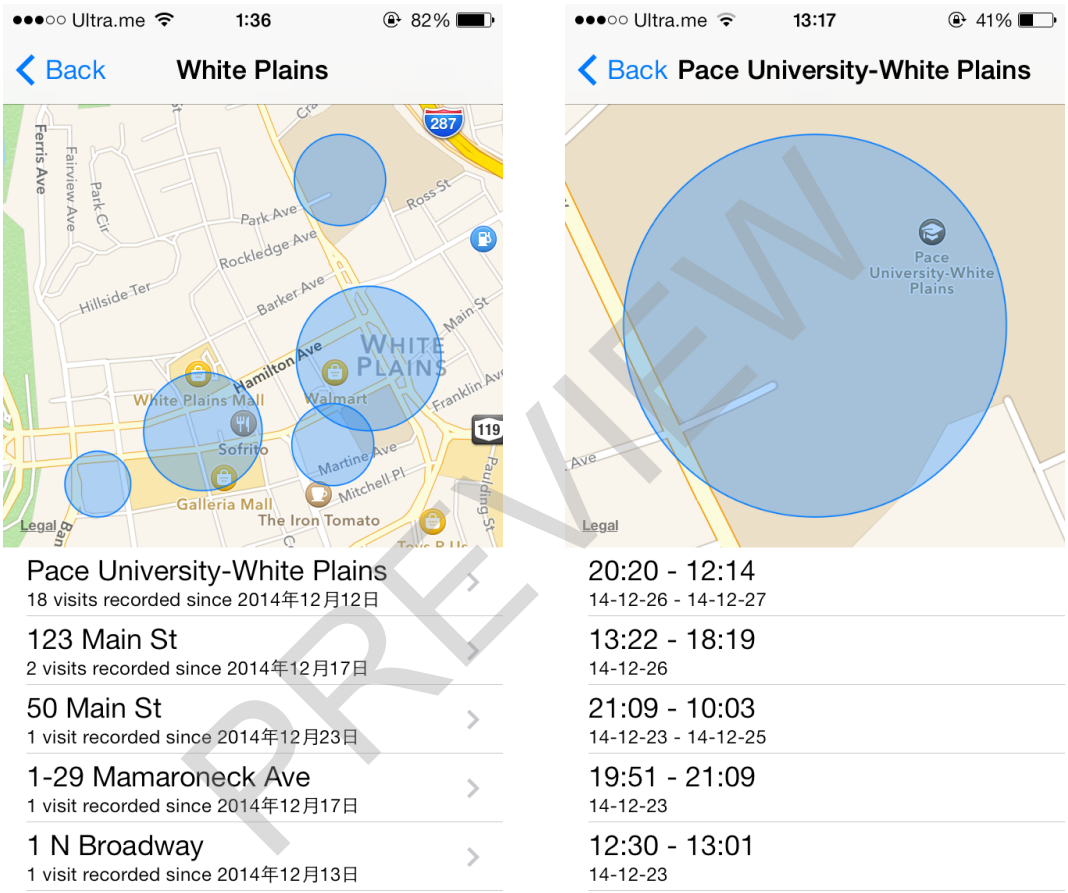
In this section, we analyze the data over-collection behaviors first, and then address the motivations of our study. Current mobile phone operating systems only provide coarse-grained permissions for regulating whether an app can access private information, while provide little insight into how much private information is actually used. Meanwhile only few users actually notice and understand permission information during installation. Furthermore, as shown in [12], small part of users choose to stop installing or to uninstall an app when system warns them and asks for permission, even though they know it may bring some hidden security troubles. We choose some most often data over-collected behaviors, analyze their current status and discuss their risks.

1.2.1 Tracking Location

Location data is the most frequently used data in smartphones. It can be used in apps whose main function are maps, photo organization, *social networking service* (SNS), shopping and restaurant recommending, and weather [13]. From the report of Appthority [7], 50% of the top iOS free apps and 24% of the top iOS paid apps to track users' location. Although, users are warned whenever an app intends to capture user's location, they usually choose to allow the permission for the function offered by the app.

Ironically, iOS system itself offers a system service, named Frequent Locations [14]. It allows the system to learn places user frequently visit. From the view of users, it is totally useless because it is impossible that someone has no idea about the accuracy address where he/she visits often. From the view of Apple, it says that Apple uses the frequent location information to improve Maps and other Apple location-based products and services. For the old users, who update their iOS from older version to iOS 7 or higher, this service was default "on".

For the new users, who hold a latest Apple device with iOS 8 or higher, they are asked about the permission related to this service only in the first time they use their devices. Furthermore, this service meticulously records the amount of visits, the date, the time, and duration user being somewhere, as shown in Figure 1.1, which is over-collection.



(a) (b)

Figure 1.1: Detailed Location Data Tracked by the Frequent Location Service in an iOS

It is easy to reason that this service must capture and record every information about where and when user goes. Then the top frequent records are sorted out. Thankfully it is easy to disable this service, clear the history, and stop it from logging these details. Unfortunately, Apple will still collect the information, and it just means users won't be able to access it in the from of a map. Although Apple has insisted that none of the data ever leaves the phone, it is easy to image the severity when this information is divulged to others.

Much worse than iOS, from the report of Appthority [7], 82% of the top Android free apps and 49% of the top Android paid apps track users location. W. Enck et al. studied 1100 Android apps, and found that half of these apps exposed location data to third-party advertisement servers without requiring implicit or explicit user consent [2].

Apps which over-collect location data can be separated into two types roughly: location service as main function and location service as auxiliary function. The first type of apps normally ask users for permission to their location information, while some of the second ones can collect users' location information without any conspicuous notice.

1.2.2 Accessing Photos

Album is also widely used in smartphones. Users not only take photos for memory, but also for convenience such as taking photos of the slides instead of writing them down and printscreening the route found by Maps. As a result, smartphones with large storage capacity hold increasing amount of pictures including life photos and information pictures. Meanwhile, due to the popularity of SNS, people form a habit of posting a photo showing what they are doing using SNS apps [15]. Almost all SNS apps successfully get permission to users' Album. There are some other kinds of apps also access users' album, such as cloud storage app, wallpaper app, customized album app, and picture decorating app.

In fact, users use these apps to deal with just several or parts of photos, not all of them. However, current smartphone operating systems just provide coarse-grained permission authorization, all or none. Furthermore, this kind of permission authorization is always one-time operation, and almost appears during the first-time use. If an user authorizes an app the permission to their album

once, this app will have this permission forever. iOS gives users a way of escape, that users can manually disable the permission of an app to their photos in the Privacy Settings. What's worse in Android, users have no ways to disable the permission of some app unless uninstall it.

1.2.3 Accessing Address Book

To contact with others more convenient, users are willing to create new contacts, replenish existing contacts with email address, new phone number, and remarks [16]. The functionality of address book does provide users convenience for communication and work. However, over-collection of users' address books brings serious potential security hazards.

From the report of Appthority [7], 26% of the top iOS free apps and 8% of the top iOS paid apps access users address books, and 30% of the top Android free apps and 14% of the top Android paid apps access users address books. The address book includes user names, physical address, phone numbers, email addresses, and other notes. Similarly to photos, current smartphones operation systems fail to provide fine-grained permission authorization. Once an app get the permission to user's address book, it can collect all information in this address book. The data about address book is usually captured by SNS apps, online game apps, commercial apps and apps which have the function of "sharing with your friend".

1.2.4 Accessing Calendar

Calendar apps are used in aim of organizing users' schedule, tracking events and reminding users of impending events. From the report of Appthority [7], 2% of the top Android free apps and 4% of the top Android paid apps access users calendars, and 8% of the top iOS free apps and 1% of the top iOS paid apps access users calendars. User stores the names and phone numbers for meeting attendees, meeting date, and time and attachments within the remark section [17]. Both iOS and Android have their calendar apps attached in the operating system, and it is impossible to uninstall them unless jailbreak. These calendar apps can easily get permission to users' calendar because their main function is managing users' calendar. As a result, scarcely users deny the permission

of calendar apps to their calendar. Furthermore, there are some other kinds of apps, offered in the market, also use calendar as their auxiliary function, such as lifestyle apps, travel apps, and business apps. Once getting the permission to a user's calendar, all these apps will access every line of information stored within that calendar.

1.2.5 Tracking IMEI/UDID

Both International Mobile Station Equipment Identity (IMEI) and Unique Device Identifier (UDID) are the unique ID of one's phone, and they are like cookies in website that users can not delete. Although Apple has prohibited iOS developers from using UDID as a method to track and identify users, this rule is only enforced on devices with the latest version of iOS. In fact, iOS 8 adoption rate just inched up to around 50 percent [18]. Furthermore, Apple has encouraged app develpoers to use new methods of user identification, to track user behavior on an app-by-app basis.

From the report of Appthority [7], 88% of the top Android free apps and 65% of the top Android paid apps access IMEI/UDID, and 57% of the top iOS free apps and 28% of the top iOS paid apps access IMEI/UDID. As we all known the importance of Primary Key in database, IMEI/UDID is equally significant as Primary Key. It identifies each data in phones and makes all data categorized by device. That makes data closer to the real world and more valuable for data mining.

The main concern of accessing IMEI/UDID is that user behavior can be correlated across multiple apps and matched to a unique user. Even if a user has various usernames and passwords for each app, his/her data can easily be integrated by the unique ID of his/her device. Furthermore, the IMEI/UDID may also used to match with real user data, such as names, passwords, locations, and others. It allows app developers and advertisement networks to create a complete profile of a user across multiples apps and profiles and combine with other over-collected data for an in-depth view of users.

1.2.6 Motivations

The main motivations and purposes of this study consist of three aspects.

1. First, from the view of mobile system users, their privacy could be better protected.

2. Then, from the view of mobile app developers, their efforts on using and maintaining mobile data could be reduced.

3. Finally, from the view of mobile device manufacturers, the burden of deploying large storage space could be released greatly, and they can focus more on the industrial design without considering the space for large storage.

The next section describes the main research problems as well as sub-research questions.

1.3 Research Problems and Solutions

The main research problem of this study addresses the design of fine-grained data access permission authorization for mobile systems. Align with the statement in Section 1.1 and 1.2.1, we propose the main research problem as follows:

Main Research Problem: *How to design a fine-grained data access system for mobile systems while keeping high performance and efficiency?*

The coarse-grained data access permission authorization policy is the sin of data over-collection behaviors from apps. After analyzing their motivations and characteristics, we plan to offer a fine-grained data access system with help of cloud storage. However, there are several issues we need to solve in this topic. Figure 1.2 shows the relations between the main research problem and sub-research problems. First task is distinguishing data based on privacy. There are a few different types of data, and even the same data are with different privacy information. Current mobile operating systems are not as powerful as traditional computer operating systems. They can only manage the same kind of files and data in one policy. This shortage brings about the second issue, which is to separate mobile data based on privacy. Mobile data separation include to separate and access data policies. After separating mobile data, we also need to design the corresponding functions to access them.

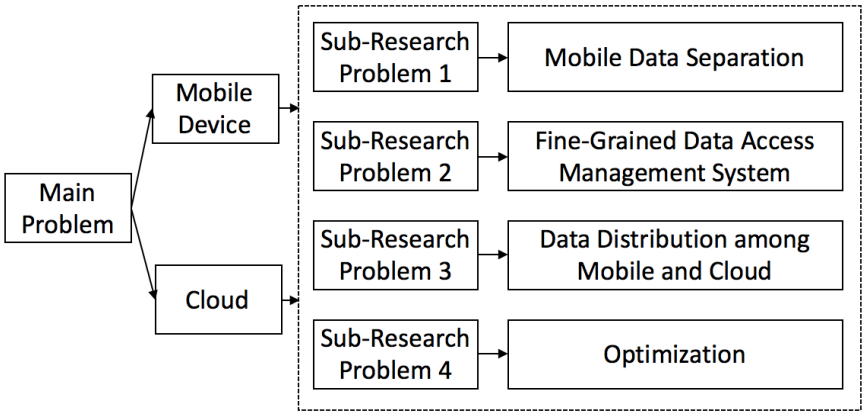


Figure 1.2: Mapping sub-research problems

Moreover, it is necessary to have a complete system to manage separated data and their access policies in the middle of mobile device and cloud storage. The third task is to design a fine-grained mobile data access management system. We need to design the communication protocol between mobile device and the management system, data and policy management, mapping and indexing in cloud storage. The last task is to optimize the whole system for performance and efficiency. The detailed descriptions of four sub-research problems are stated as follows.

Sub-Research Problem 1: *How to distinguish data with different privacy levels without too much users' involvement?*

In mobile devices, there are various kinds of data stored on storage level. Some of them are sensitive and private, while some of them are not. Furthermore, even the same kind of data are with different privacy levels. For example, photos with friends are not so private as the copy of SSN(Social Security Number). It is extremely difficult to automatically tell how private some data are. Classifying users' data based on their privacy is the first step to provide fine-grained permission authorization, but it is not reasonable to force users to take care of their own privacy by separating data into different levels, which will burden and annoy users with much higher operation complexity. We need some methods which can classify users' data automatically based on privacy. However, those methods are not easy to be implemented, since users' data in smartphones could be with multiple types and could not be processed using traditional algorithms directly.

Sub-Research Problem 2: *How to implement fine-grained data access request from apps?*

Due to the coarse-grained permission policy in current mobile operating systems, the data access requests from apps are also coarse-grained and imperfect. From Android 4.4, Google introduced Storage Access Framework (SAF) to let users browse and open documents, images, and other files easily [19]. When the app fires the ACTION_OPEN_DOCUMENT intent, all matching documents providers. Then *setType()* method is used to setup the type of requested data. For example, we can use *intent.setType("image/*")* filter to display only image MIME data. Consequently, data type is the data access request granularity of current Android system. From the perspective of apps, they never know which piece of data is the one that user exactly need. As a result, they only can apply the data access policy referring to mobile operating system, that uses data type as the granularity.

Sub-Research Problem 3: *How to ensure the security and privacy on mobile data transmission?*

Mobile data usage mainly consists three kinds of situations based on security and privacy. The first one is that data are only used in mobile device locally. It is the most secure and private usage case, but it only accounts for small proportion. The second one is that data are sent to app developers. For example, online game apps must store users' data on servers. SNS app also need to store users' data for synchronization [20]. In this case, the protection of data storage is controlled by developers. We need to ensure the communication channel secure and reduce the risk of leaking all data. The third one is that data are sent to third-party organizations, such as advertising agencies. This case is the most common one, and does the greatest harm to users' privacy. To protect privacy in this case, we need to know the communication flow of users' data and monitor app's behaviors.

However, it is not wise for any mechanism to monitor app's behaviors and constrain the permission at runtime. The first reason is that this kind of mechanism takes too much resource, since it must be running all the time. The second reason is that itself is one kind of data over-collection behavior. As a result, the third important problem is to ensure the security and privacy of users' data in lowest cost.

Sub-Research Problem 4: *How to improve the performance and efficiency?*

Systems with web based fine-grained data access permission must be with higher latency than coarse-grained one, since there are more executions. These extra executions consist two parts basically. The first one is the communication between cloud storage and local mobile devices. The other one is data separation in mobile devices. Users are more captious and less patient to mobile apps. It is necessary to improve the performance of the whole framework. Meanwhile, more executions indicate that the energy consumption is higher than before, and battery is another important aspects influencing users' experience. As a result, we also need to consider energy consumption as the efficiency.

In line with the main research problems and four sub-research problems, we summarize the main contributions of this study in the next section.

1.4 Contributions

The main contributions of this study include:

1. We propose a phenomenon, data over-collection, which is widely happened and brings about serious privacy leakage risk in mobile systems. We study the current state of data over-collection and some most frequent cases. We quantify the risk of apps' data over-collection behaviors and design a grading benchmark to evaluate privacy protection approaches.

2. This study proposes a set of mobile data separation approaches. These approaches separate mobile data with different privacy levels and apply a finer-grained data access management in local mobile system. We analyze both persistent and temporary mobile data and implement two sets of methods, which are never proposed by others before. Using our own benchmark and experimental apps, the feasibility and advantages of our approach are proved.

3. We provide provide some simple modifications to Android APIs. These APIs are easy to use and require little changes to the code. The function call and usage are the same with the original ones, we modify the parameters and Manifest.xml to provide high privacy protected

methods.

4. We implementa a fine-grained data access system, DASS, to manage mobile data with help of cloud storage. We separate mobile data by types, contents, and privacy levels, and store them into cloud storage. Furthermore, we design multi-dimension data access policies to be applied in different situations.

5. To reduce the extra latency, we design a set of optimization methods for DASS and cloud storage. The optimization includes a mathematic multi-constraints optimization problem and the optimization of storage system. This set of optimization is particularly designed for this study and experimental results show the performance increase of the whole system.

6. We develop some sample apps with our own APIs to example app developers. These apps are with common used functions towards mobile data, such as image, contact, document, message, email, and location. Meanwhile, these apps not only can be used in our system, but also can be easily modified to fit other systems, since we use RESTful protocal to format the in/ouput.

1.5 Organization

The rest parts of this dissertation is organized by the followings. First, Chapter 2 illustrates a comprehensive literature review on the relevant fields of the research. Next, Chapter 3 presents the data over-collection behaviors and the quantification about privacy and security risk. In addition, Chapter 4 designs and implements two sets of approach to separate persistent and temporary mobile data. Furthermore, Chapter 5 addresses the DASS system, which is used to offer fine-grained data access mechanism for smartphones with help of cloud storage. Moreover, Chapter 6 designs a mobile data distribution deployment approach to allocate data to mobile and cloud based on data type, data privacy level, and data amount. Finally, Chapter 7 presents a novel cloud resource management optimization method considering the cost, availiablity, and performance.

Chapter 2

Literature Review

2.1 Mobile Data Leakage Analysis and Approaches

In our previous research [9], we defined Data Over-Collection behaviors, which is collecting data more than enough on original function while within the permission scope. We analyzed some common data over-collection behaviors, the motivating reasons behind them, and the risks and damages they brings about. Furthermore, we quantified the security risk and proposed a framework to improve the privacy of mobile data. However, we only implemented two general function of accessing and requesting data and just did some simulation experiments on local server.

C. Rottermanner et. al. analyzed several popular messaging apps in smartphones [21]. They found that all these apps request over 7 permissions that could leak private information about users. Furthermore, they analyzed the attacking behaviors towards the local database. They only focused on messengering apps, but similar privacy issues are also common in SNS apps, GPS apps, and others. B. Zhang and H. Xu applied two permission interfaces, the frequency nudge and the social nudge, into app settings of smartphone [22]. The main purpose of their research is to alter users' privacy attitudes and then to facilitate their decision-making on information sharing. However, these two interfaces can only work on installed apps. Data over-collection behaviors could happen once users use some app at the very first time.

M. Egele et al. presented PiOS [1] to detect privacy leaks in iOS applications. They used static

analysis to detect sensitive data flow to achieve the aim of detecting privacy leaks in applications in iOS. PiOS checks an iOS application by three steps. First, it reconstructs the control flow graph of the application to find code paths from sensitive sources to sinks. Second, it performs a standard reachability analysis to find the paths in the control flow graph which connect nodes accessing sensitive information to nodes interacting with the network. Third, it performs data flow analysis along the paths to verify whether sensitive information is indeed flowing from the source to the sink.

Sharing a similar goal with PiOS, W. Enck et al. proposed TaintDroid [2], a system-wide dynamic taint tracking multiple sources of sensitive data. The main strategy of TaintDroid is real-time analysis by leveraging Android's virtualized execution environment through three steps. First, they instrument the VM interpreter to provide *variable-level tracking* within untrusted application code. Second, they use *message-level tracking* between applications. Third, they use *file-level tracking* to ensure persistent information conservatively retains its taint markings. R. Stevens et al. propsoed a system that allowed network API to authenticate the mobile app that sended each request so that the API can make an informed access control decision [23]. Meanwhile, they presented the Mobile Trusted-Origin Policy, including an app provenance mechanism annotating outgoing HTTP requests with information about which app generated the network traffic and a code isolation mechanism that separated code within an app which should have different provenance signatures into a "mobile origin".

P. Gilbert et al. [24] present a model of making mobile apps more secure via automated validation. They use commodity cloud infrastructure to emulate smartphones to dynamically track information flows and actions. Then automatically detect malicious behavior and misuse of sensitive data via further analysis of dependency graphs based on the tracked information flows and actions. R. Herbster et al. propsoed privacy capsules, a platform execution model for mobile apps that prevented the flow of private information to untruested parties [25].

J. Ren et al. presented the design, implementation, and evaluation of ReCon: a cross-platform system that reveals personally identifiable information(PII) leaks and gives users control over them without requiring any special privileges or custom OSes [26]. They used machine learning to reveal