DiffUser: Differentiated User Access Control on Smartphones

Xudong Ni, Zhimin Yang, Xiaole Bai, Adam C. Champion, and Dong Xuan

Department of Computer Science and Engineering
The Ohio State University
Columbus,Ohio,USA
{*nxd,yangz,baixia,champion,xuan*}*@cse.ohio-state.edu*

Abstract

Smartphones have been widely used in recent years due to their capabilities of supporting many applications from simple Short Message Service messages to complicated Location-based services. It is challenging for smartphones to enable their end users to manage all applications in all possible use cases to protect privacy or sensitive data. However, the security model for smartphone users is still a two-state model in which they can do anything or absolutely nothing, and it is no longer suitable. In this paper, we propose DiffUser*, a* diff*erentiated* user *access control model to enhance smartphone security and user privacy. DiffUser classifies smartphone users based on certain sets of user access privileges. We implement a prototype of DiffUser on real-world T-Mobile G1 smartphones. The evaluation results show that our system is lightweight and flexible.*

1. Introduction

Recently, smartphones are becoming increasingly popular and the number thereof has increased greatly in the past few years. A report from Canalys [1] shows that, despite the worldwide economic downturn, global shipments of smartphones hit a new peak of just under 40 million units in the third quarter of 2008 and smartphones represent around 13% of the total mobile phone market. Several companies such as Nokia, Microsoft, and Apple, and Google develop various smartphone operating systems (OSes) such as Symbian OS, Windows Mobile, iPhone OS, and Android, respectively. It is likely that smartphones will have a strong presence in the future mobile phone market.

1.1. Motivation

Smartphones' processing capability exceeds that of "regular" mobile phones and it continues to grow. For example, today's iPhone has much greater processing capability than an IBM PC did in 1981 [2]. In addition, smartphones are often equipped with additional functionality such as GPS systems, cameras, Wi-Fi, FM radios, Bluetooth, and various sensors. They can support many new applications such as Internet services, photography applications, and location-based services. Consider the Android operating system. Although it was released less than two years ago, there are thousands of applications on the Android Market [3]. Smartphones are used pervasively in our daily lives.

This poses a challenge for smartphone systems. How can they enable smartphone owners to manage all applications installed thereon in all possible use cases, thereby protecting privacy and sensitive data? The following examples illustrate this challenge and existing limitations in such state-of-the-art systems.

– *Example 1*. Many enterprises distribute company-owned smartphones to their employees for business use, on which critical applications are installed to facilitate corporate operations and workflows [4]. Usually, only a device administrator should be able to install or uninstall applications. He would like to prohibit an employee from (1) uninstalling these critical applications, (2) installing malicious applications (either intentionally or unintentionally), or (3) installing non-work-related applications. This use case demands that smartphone OSes differentiate between administrative and normal users. However, current smartphone OSes do not provide such functionality, as they do not differentiate among users.

– *Example 2*. Since smartphones are becoming popular, children commonly have access to them. Consider parents who want to control their children's smartphone use. They do not want their children to spend too much time playing games or accessing the Internet and they want to block inappropriate content. Thus they need means to control their children's smartphone access aand determine what applications their children can run and how long they can they run them, among other things.

– *Example 3*. Many people lend their mobile phones to someone else—a friend, a coworker, or even a stranger. In most cases, the owner wants to limit phone usage to specific functions or applications such as making calls. The mobile phone borrower should not be able to access the owner's private information (e.g., contacts, pictures, multimedia, etc.) stored in the phone.

978-1-4244-5113-5/09/$25.00

There are other situations in which a smartphone owner needs to customize access rights for a specific user. In summary, although there are many demands for smartphone security models, current smartphone OSes have no mechanisms to support these demands, not to mention user access control. To our knowledge, all existing smartphones are designed for a *single* user.

1.2. Our Contributions

In this paper, we propose DiffUser, a *diff*erentiated *user* access control model for smartphone systems. DiffUser classifies smartphone users based on certain sets of user access privileges. Our specific contributions are as follows:

– We present DiffUser on smartphones and investigate the potential demands for differentiated users on them. Then, we classify smartphone users based on particular user access privileges. In addition, we propose an access control mechanism designed for user management of smartphones.

– We implement a prototype of DiffUser on a real smartphone system, Android, which is an open source platform. Although Android is based on the Linux kernel, it is a new OS specifically designed for mobile devices. We take the advantage of its built-in permission-based access control to implement our prototype. However our system could also be applied to other popular phone sytem, such as Symbian.

To our knowledge, we are the first to propose a differentiated user access control system for smartphones. We classify users into three classes: (1) *administrative users*, who have complete control over smartphones; (2) *normal users*, who have many smartphone privileges but cannot install or uninstall critical system applications; and (3) and *guest users*, who have very limited privileges. These three user types can solve the problems in the above examples.

The rest of the paper is organized as follows. Section 2 describes related work. Section 3 discusses the concept of *differentiated users* and our access control model. Section 4 details our prototype of DiffUser on Android. Section 5 presents evaluations and discussions. Finally, Section 6 concludes the papers.

2. Related Work

Increasingly, today's mobile phones are smartphones. Although there are many smartphones OSes, all of them support *only* a single user, who is the smartphone owner. This is easy to understand: mobile phones' primary function is telephony, which is suitable for a *single* user. However, smartphones support many applications and provides programming APIs that far exceed telephony. There is some work related to our. VMware [5] proposes using virtual machines to facilitate employee and personal use of smartphones, but their approach cannot provide fine-grained user access control. To manage children's use of phones, AT&T [6] has a service called Smart Limits for Wireless that can set monthly usage limits, time-of-day restrictions and content filters. However, it only affects AT&T services and it cannot manage other applications such as games or browsing the Internet through Wi-Fi. In addition, users must pay monthly fees and use AT&T phones.

Some work has been done on smartphone access control [8]. But their focus is on the *security* of applications installed on smartphones. The access control on whom and how to run applications is not taken into account. Our method targets a higher level of access control, such as who has privileges to execute programs and how they may be executed.

Some smartphone OSes, such as Symbian OS, are specially designed for *single-user mobile* phones. Others such as Windows Mobile, iPhone OS, and Android are ported from *multi-user* desktop OSes such as MS Windows, OS X, and Linux, respectively, and these smartphone OSes lack their counterparts' multi-user access. There is demand for multi-user access on smartphones, but this multi-user access is *different* from that of "traditional" desktop computers. The single-user *mobile* phone OSes only have basic functionalities such as hardware management, so they did not require multi-user access. Desktop OSes' multi-user support is not applicable to smartphones, as many people can use a traditional multi-user system for various purposes, but only one person at a time can use a smartphone, and usually there is only one *primary* user thereof. Also, unlike these traditional systems, smartphones have no concept of a user's "home" directory for his file storage. Our DiffUser system provides multi-user access for smartphones in which only *one* person uses the phone at a time. We *differentiate* smartphone users in order to provide security and access control for them.

Some work has been done on mobile phone sharing. In [9], the author did field studies reporting that there are a variety of situtation in which people naturally share phones. However, the author just did the survey and provides no multi-user implementation. In [10], the author proposed a shared mode to support impromptu sharing of mobile phones. But this work mainly focuses on sharing media files such as pictures or music and the implementation is very simliar to Windows file sharing. Our work is based on a *permission* security model, which is much more flexible than their virtual file-level access system.

Above all, our DiffUser system differs from traditional computers' multi-user concept. In addition, it is easier to incorporate fine-grained access control to better support execution of smartphone applications in different modes.

3. Differentiated Users on Smartphones

In this section, we define the concept of *differentiated users* on smartphones and classify them.

Differentiated users is a security model that provides smartphone users with a predefined set of access rights for *different* smartphone uses in different contexts. It replaces the existing "all-or-nothing" mobile phone security model.

Desktop OSes' multi-user concept is designed for computer systems to be used by multiple users whose identities are usually known. In such OSes, a group of users has similar access rights. Though *these* users are members of *one* group, *any* system user may be a member of *multiple* groups. User and group management relies on these OSes' support of file system access privileges. Originally, mobile phone OSes supported only a *single* user, but *smartphone* OSes require *multi*-user support due to the number of applications they support and possible use cases. DiffUser *does not* port desktop OSes' multi-user concept to smartphone OSes. Rather, DiffUser defines several *classes* of users and any smartphone user is in *exactly one* class. By default, DiffUser defines a set of user access control privileges that correspond to these classes and each user's privileges are *exactly those* of the class to which he belongs. However, DiffUser also allows device administrators to create new classes and modify existing classes of user access control privileges. This model can satisfy the user demands discussed in the Introduction.

To illustrate DiffUser in detail, consider the differences between a a desktop OS administrator and a mobile phone user. The administrator has complete authority over the desktop OS, e.g., the `root` user in UNIX/Linux systems. However, smartphone OSes are closely related to the interests of stakeholders such as cellular service providers, handset and OS manufacturers, enterprises, developers, and end users. Network providers may restrict smartphones' functionality in accord with their business interests. DiffUser does not compromise these restrictions. Thus a *smartphone* administrator has control over the device *subject to these restrictions*. For instance, a cellular service provider may disable a smartphone's FM radio although the device's hardware provides this functionality. DiffUser allows the smartphone administrator to manage the device, but he cannot enable the FM radio.

We classify DiffUser users into several classes: *administrators*, *normal users*, *guests*, and *other configured users*. *Administrators* are the smartphone owner. *Normal users* have fewer access rights than the administrator and are the *primary* device users. *Guests* have the fewest privileges. *Other configured users* may be defined by administrators and have different privileges than normal users and guests. For example, we group a number of smartphone features into four categories: *personal information*, *resource access*, *system settings*, and *applications*. *Personal information* includes SMS messages, contact lists, etc. *Resource access* includes sensors and networking functionalities such as Wi-Fi, GPS(location-based service), and Bluetooth.*System settings* include the capability to install and uninstall applications, set the time and date, restore factory settings, etc. *Applications* are self-explanatory. Figure 1 shows an example of different smartphone users' access rights.

Categories of Smartphone Privileges		Administrator	Normal User	Guest
Personal information	SMS	√	√	×
	Contact	√	√	×
Resource Access	Wi-Fi	√	√	Strong Limited
	GPS	√	√	Strong Limited
	Bluetooth	√	√	Strong Limited
System Setting	Software Install/Uninstall	√	Limited	×
Applications	Sensitive Applications	√	Limited	×

Figure 1. Classification of Multiple Users

4. Prototype of Differentiated User Access Control

In this section, we present our prototype of differentiated user access control on T-Mobile G1 smartphones, which run the Android OS [11]. We discuss our prototype's workflow, access control for differentiated users, and our implementation details.

4.1. Prototype Overview

Our prototype's workflow is as follows: When a user powers on the smartphone, it first loads in normal user mode with the normal user profile shown in Figure 2(a). In this mode, the user can configure almost everything except for some system level settings. To switch to the administrator mode, the user must input the correct password as shown in Figure 2(b). If the authentication succeeds, the user will enter administrator mode with full privileges. The administrator can define access rights for different user classes as Figure 2(c) illustrates. The user could enter guest mode instead of administrator mode by pressing the "Quick Switch" button. To enter normal user mode or administrator mode from guest mode, the user must enter the correct password.

Our workflow includes three modules: *authentication*, *user profile configuration*, and *access control*. The *authentication* module switches between different user modes. The *user profile configuration* module allows the administrator to define or modify other user profiles. The *access control*

module checks different user profiles to control their access rights.

We apply a *capability*-based access control model to DiffUser instead of access control *lists* (ACLs) for the following reasons:

– ACLs are unsuitable for mobile phones. While ACLs are suitable for desktop OSes due to their file permission systems, e.g., NTFS, ext3, etc., mobile phone OSes either provide no such permission systems or they use abstraction layers for data and applications that hide file permissions from end users. For examples, Symbian OS and Windows Mobile do not support file access control. In addition, though Android has a Linux-based file system with native file permissions, these have no effect on the end user's experience. Android's built-in application files are encapsulated as `.apk` files that are stored in the `/system/app` directory and owned by `root`. Other applications are saved in the `/data/app` directory and owned by the `system` user. None of these is related to end users and they cannot be mapped to different `uids` as in traditional UNIX/Linux systems. Moreover, some data are saved in SQLite [12] database files and Android's database implementation only differentiates among *applications*.

– Smartphones usually have few users, so it is easier to maintain user *profiles* as opposed to ACLs.

4.2. Prototype Implementation on Android

In this section, we present our Android prototype implementation. First, we describe the end-user interface and corresponding model. Second, we discuss the access control implementation based on Android's built-in permission mechanism. Finally, we analyze the relationship between permissions and differentiated user profiles.

Android is the first free, open source, and fully customizable mobile platform. It offers a full software stack consisting of an operating system, middleware, and key mobile applications as well as a rich set of APIs that allow third-party developers to develop applications. It is based on the Linux kernel and the system is divided into four layers: the kernel, libraries and runtime, application framework, and applications. Different layers are implemented in different languages. For example, Android's system services such as drivers and inter-process communication are written in C or C++, while all platform-related applications and services are written in Java and executed by the Dalvik virtual machine. Android's advantage lies in its open source licensing and extensive programming documentation. The Android SDK provides the tools and APIs necessary to develop applications on the Android platform using Java.

Our prototype is implemented in Java with Android SDK 1.0. The development environment is Eclipse 3.1 with Google's Android Developer Tools plugin. The debug tools are Dalvik Debug Monitor Service (`ddms`), which manages

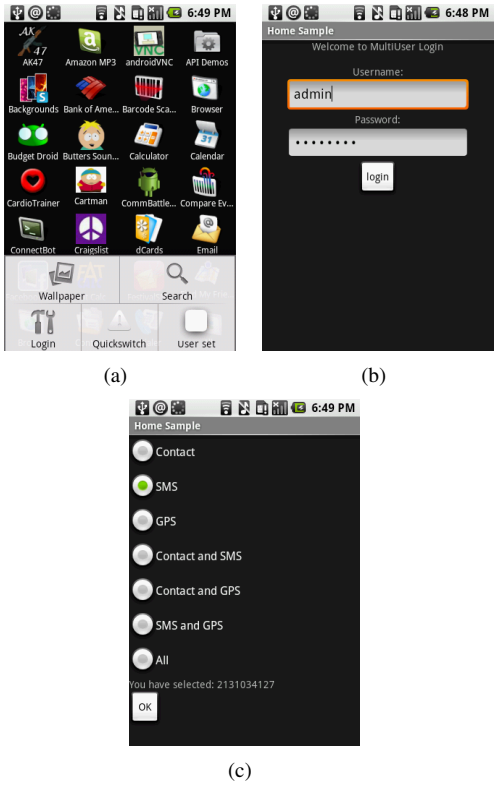

Figure 2. A set of three interfaces: (a) Normal User Home; (b) Login/Authentication ; and (c) User profile Configuration

processes on the Android emulator or smartphone, and Android Debug Bridge (`adb`), which provides a command-line interface to programmers. The source code for our prototype is about 72.3KB. As Android programs require configuration profiles and resources, which are another 225KB, the installed file on the phone is a 216KB `Home.apk` file.

Screenshots of the user interface are given in Figure 2. We load different applications according to users' profiles. Our prototype includes a login Activity, a user profile configuration Activity and a main Home Activity. An Activity is the presentation layer of an Android application screen; its "lifecycle" has four stages—start, pause, resume, and terminate. An Activity provides event handling methods for each of these stages that can be overridden. The user needs to authenticate himself at the login screen to obtain administrative privileges, including modifying system settings. The user configuration Activity provides the interface to determine user access rights to SMS, contacts and GPS. These rights are stored in user profiles. We reprogram Android's native Home Activity. After the system starts up, the Home Activity invokes the method `loadapplication()` to obtain information about all applications installed on the system, including related permissions, that is provided by the system service `PackageManagerService`. We add

End-user Access Rights	Android build-in Permissions
Contact	android.permission.READ_CONTACTS
	android.permission.WRITE_CONTACTS
SMS	android.permission.WRITE_SMS
	android.permission.READ_SMS
	android.permission.RECEIVE_SMS
	android.permission.SEND_SMS
Wi-Fi	android.permission.CHANGE_WIFI_STATE
GPS	android.permission.ACCESS_FINE_LOCATION
Bluetooth	android.permission.BLUETOOTH
	android.permission.BLUETOOTH_ADMIN
Setting	android.permission.WRITE_SECURE_SETTINGS

Figure 3. Permissions Mapping

our permission checking when the Home Activity loads this information from the service so different users in different classes see different available applications in the Home screen. We add a "quick switch" menu option to switch between the administrator or normal user profile to the guest profile. To switch between any two profiles, we invoke `loadapplication()` in the overridden `onRestart()` method provided by the Home Activity.

We use Android's built-in permission-based security model to implement access control. This model provides fine-grained control over what operations a process can perform, e.g., accessing the smartphone camera, location information, etc. Permissions enforce restrictions on these operations and grant access to particular data. As only application developers use permissions for these purposes (end users cannot control them), we need to map these permissions in our prototype to users' access rights, which are stored in user profiles. Android provides an exhaustive set of system permissions (over 100 of them). We only illustrate a subset of those permissions that we map to user access rights in Figure 3. Using the configuration Activity shown in Figure 2(c), we can configure a specific user profile. We use a blacklist method to control access rights, which reduces the overhead associated with providing *all* user privileges.

5. Evaluation and Discussion

To evaluate our system, we examine two metrics: battery consumption and switching latency between differentiated user modes. We use `ddms`, which is connected to a T-Mobile

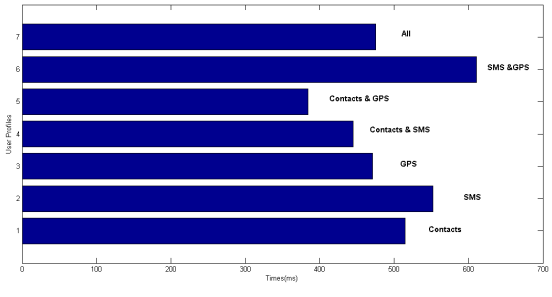

Figure 4. Switching Latency in Different Profiles.

G1 smartphone. We use version 1.1 of the Android SDK and the `ota-radio-1_22_14_11` radio firmware.

To evaluate our prototype's power consumption, we repeatedly execute a sensor-related application from our modified Home Activity in different user profiles compared with executing the application from the original Home Activity. Our results show that there is no measurable difference in power consumption. We attribute this to the fact that our code does not intercept execution of running applications.

To evaluate the switching latency between different user profiles, we design seven of them, including uses of SMS, contacts, and GPS. We install 50 applications from Google's Android Market. We record the latency of these user profiles. The results are shown in Figure 4. The numbers "1" through "7" correspond to revoked access to contacts only, SMS only, GPS only, contacts and SMS only, contacts and GPS only, SMS and GPS only, and all three features. We find that the overall latency is not very high with a minimum of 384 ms and a maximum of 611 ms. Thus it has minimal impact on normal usage of the smartphone. Different user profiles have minor switching latency differences.

In Figure 5, we install 50, 60, 70 and 80 applications and evaluate their latency in seven user profiles. The latency increases with the number of applications, as our program has to load more application information. However, the more permissions we forbid, the faster switching occurs. This is because forbidding more permissions leads to querying and loading fewer applications.

Although our solution can improve end user security, we cannot fix existing security flaws in smartphone systems. For example, if the end user has gained root access on his Android phone, we cannot prevent him from modifying system settings and circumventing system security.

6. Conclusion

In this paper, we present a differentiated user access control for smartphone operating systems. Our goal is to provide end users a finer-grained control instead of an "all-or-nothing" security model. We implemented a prototype sys-

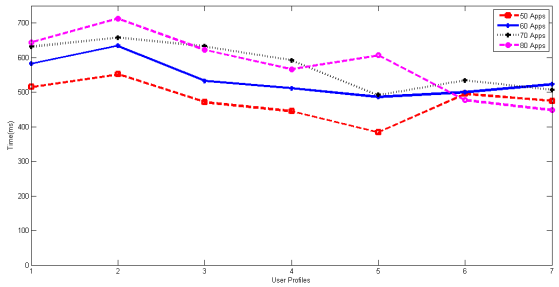

Figure 5. Switching Latency with Different Numbers of Apps.

tem on T-Mobile G1 smartphones based on Android's built-in permission mechanism. Our evaluation results showed that our system is lightweight and flexible.

References

[1] http://www.canalys.com/pr/2008/r2008112.html
[2] http://www.zdnet.com.au/news/hardware/soa/
[3] http://www.android.com/market.html
[4] S. Furnell, *Securing mobile devices:technology and attitude,* Network Security, Volume 2006, Issue 8, August 2006
[5] http://www.vmware.com/technology/mobile.html
[6] http://www.wireless.att.com/learn/articles-resources/
[7] D. Muthukumaran, A. Sawani, J. Schiffman, B. M. Jung, and T. Jaeger, *Measuring integrity on mobile phone systems,* in Proc. of ACM symposium on Access Control models and technologies (SACMAT), 2008
[8] S. M. Lee, S. B. Suh, B. Jeong, and S. Mo, *A Multi-Layer Mandatory Access Control Mechanism for Mobile Devices Based on Virtualization,* IEEE Consumer Communications and Networking Conference (CCNC), 2008
[9] A. K. Karlson, A. J. B. Brush, and S. Schechter, *Can I Borrow Your Phone? Understanding Concerns When Sharing Mobile Phones,* in Proc. of the 26th annual SIGCHI conference on Human factors in computing systems (CHI), 2009
[10] Y. Liu, A. Rahmati, Y. Huang, H. Jang, L. Zhong, Y. Zhang, and S. Zhang, *sShare:Supporting Impromptu Sharing of Mobile Phones,* in Proc. of the 7th Annual International Conference on Mobile Systems, Applications and Services (MobiSys), 2009
[11] http://source.android.com/
[12] http://www.sqlite.org/
[13] W. Enck, M Ongtang, and McDaniel *Understanding Android Security,* IEEE SECURITY & PRIVACY, 1540-7993, 2009