Norsk informatikkonferanse NIK 2011

Universitetet i Tromsø 21. – 23. November 2011

NIK-styret og redaksjonskomité

Norsk informatikkonferanse

NIK 2011

Universitetet i Tromsø
21. – 23. November 2011

NIK-styret og redaksjonskomité

John Markus Bjørndalen	Universitetet i Tromsø
Lars Ailo Bongo	Universitetet i Tromsø (redaktør og arrangør)
Dag Haugland	Universitetet i Bergen
Erik Hjelmås	Høgskolen i Gjøvik (styreleder)
Arne Løkketangen	Høgskolen i Molde
Birger Møller-Pedersen	Universitetet i Oslo
Andreas Prinz	Universitetet i Agder
Ragnhild Kobro Runde	Universitetet i Oslo
Frode Eika Sandnes	Høgskolen i Oslo og Akershus
Trond Aalberg	NTNU

ISSN 1892-0713
ISBN 978-82-519-2843-4

Redaktør: Lars Ailo Bongo, Universitetet i Tromsø
Digital trykk og innbinding: AIT Oslo AS

Forlagsredaktør for denne utgivelsen:
Lasse.Postmyr@tapirforlag.no

Tapir Akademisk Forlag
7005 TRONDHEIM
Tlf.: 73 59 32 10
Faks: 73 59 32 04
E-post: post@tapirforlag.no
www.tapirforlag.no

Forord

Norsk informatikkonferanse har vært arrangert hvert år siden 1988 ved de forskjellige universitetene og høyskolene i Norge. I år er turen igjen kommet til Tromsø. Formålet har hele tiden vært det samme:

- å være et møtested for datafaglig ansatte og studenter i høyere undervisnings-institusjoner, forskningsinstitutter og næringsliv,
- å gi et representativt bilde av hva som foregår av forsknings- og avansert utviklingsarbeid i hele det norske fagmiljøet, dvs forsøke å vise bredden i forskningsmiljøet, både faglig, organisatorisk og geografisk, samt
- å holde et høyt faglig nivå og åpne for debatt om emner som opptar det data-faglige miljøet.

Årets innsendte bidrag holder et normalt nivå. 20 av 40 innsendte bidrag ble akseptert som ordinære bidrag, mens ytterligere 8 bidrag ble akseptert til en egen «short paper session». Det er flere studentarbeider blant de aksepterte bidragene og disse får økonomisk støtte fra NIK for å komme til konferansen og presentere sine bidrag.

I vurderingsarbeidet har programkomiteen også fått uvurderlig hjelp av frivillige faglige konsulenter, nemlig *Anders Andersen, Joanna Bauer, Marc Bezem, Valentin David, Bård Fjukstad, Martin Giese, Tor-Magne Stien Hagen, Helwig Hauser, Randi Karlsen, Anders Kofod-Petersen, Åge Kvalnes, Ove Daae Lampe, Fredrik Manne, Arne Maus, Aida Omerovic, Julius Parulek, Mohammad Ravanbakhsh, Romain Rouvoy, Daniel Stødle, Giacomo Tartari, Hallvard Trætteberg, Weiqing Zhang,* og *Antti Ylä-Jääski*. På vegne av redaksjonskomiteen vil jeg gjerne få takke disse.

I år inviterer vi *Deborah Estrin* fra University of California, Los Angeles (UCLA), som keynote til å fortelle oss om arbeidet og utfordringene knyttet til «participatory sensing».

Artiklene finnes også på nettet på NIKs adresse http://www.nik.no der man også finner annen informasjon om NIK.

Programkomiteen for NIK 2011.

Erik Hjelmås

Gjøvik, oktober 2011.

Innhold

Controlled Sharing of Personal Information in Android

Solvår Bø, Stian Pedersen
Dep. of Telematics, NTNU, Norway

Åsmund Nyre, Karin Bernsmed
SINTEF ICT, Norway

Abstract

Smartphones with third-party applications have become very popular. Recently, they have received attention for transferring personal information without the users' knowledge. The objective of this work is to help users to protect their privacy by increasing their consciousness on how personal information is collected and distributed. We propose a design that provides a higher degree of control by allowing users to set preferences that determine what personal information to share. We implement selected parts from our design, in order to evaluate whether this solution serves as a utility or not.

1 Introduction

Mobile phones have become a central part of our lives and we bring them wherever we go. Today, the smartphone is a fully-edged computer and it carries a lot of personal information. This makes the smartphone even more vulnerable to privacy invasions than traditional computers [1]. Lately, third-party applications for smartphones, often referred to as "apps", have become very popular. To personalize the service, apps often make use of the phone's resources, such as GPS, Internet access, contact list or calendar. However, this information is often shared without the user's consent. An extensive survey done by the Wall Street Journal [2] recently revealed that approximately 50% of the top 100 applications for Iphone and Android collected information about the users and their habits, without the users' consent. Even though an app needs permission to access such information, it only needs to ask at installation or update. Once the door has been opened, it is very difficult for an ordinary user to control the outgoing flow of personal information.

In an effort to meet the discovered privacy risk in relation to third party applications, GSMA recently published Mobile Privacy Principles [3]. These principles describe how the user's privacy should be respected and protected by applications that have access to personal information. Amongst others, three of the principles state that:

- Users shall be given opportunities to exercise meaningful choice, and control over their personal information.
- Users should be provided with information about, and an easy means to exercise, their rights over the use of their personal information.
- Users should be provided with information about privacy and security issues and ways to manage and protect their privacy.

This paper was presented at the NIK-2011 conference; see http://www.nik.no/.

The main objective of the work described in this paper is to help users protect their privacy by increasing their consciousness on how their personal information is collected and distributed, as stated by GSMA. We describe a software tool designed for mobile devices that allows the user to control and modify access permissions to the phone's resources after an application has been installed. In this paper we have chosen the mobile phone operating system Android to implement and evaluate a prototype of our proposed design.

The paper is organized as follows. Section 2 gives an overview of the Android platform, along with a brief description of how privacy and security is maintained in the operating system. The design of our software is described in Section 3. Section 4 and 5 describe the implementation and evaluation of selected parts of the proposed design. Section 6 discusses the limitations of our approach and compares it with related work. Our conclusions are provided in Section 7.

2 Android

The mobile operating system Android [4] was established in 2005, based on a vision of a free and open platform that allows for any coder to contribute. Android has gained a solid foothold in the market; a recent study reported that Android was the best-selling smartphone operating system in the fourth quarter of 2010 [5].

Android Architecture

Android is a software stack for mobile devices that consists of an operating system, middleware and some key applications. It currently offers a variety of connectivity options such as WiFi, Bluetooth and wireless data over cellular networks (GPRS, EDGE and 3G), utilization of location-based services (such as GPS), accelerometers and camera. Since Android is open-source, new technologies can easily be incorporated as they emerge [4].

In Android, applications that extend the functionality of the mobile device, commonly known as "apps", are created using the Java programming language. Apps are distributed through Android Market[1], or can be downloaded directly from third-party sites. The Android operating system is based on a customized Linux kernel, which does not differentiate between third party applications created by third-party developers, and the devices' core applications. This allows application developers to fully utilize the hardware and allows for users to tailor the phone to suit their interests and needs.

In Android, applications are built up from multiple *components*, where each component provides different functionality. Components interact by sending messages, called *intents*, to each other. Components are classified as being either activities, services, content providers or broadcast receivers. In this paper we use *activities*, which are graphical user interface (GUI) windows that contain one or more views with information that is presented to the user. We also use a *service*, which is an Android component that can run long-time processes silently in the background without any user interaction. Finally we utilize a *content provider*, which is a lightweight database commonly used by developers to share data between applications. To query data from the content provider we use *content resolver objects*.

All Android application must have a *manifest file* in its root directory. The manifest file is used to register the components the application is composed of, to declare what

[1] `https://market.android.com/`

access permissions the application needs and to declare what access permissions other applications must have in order to interact with the application's components [6]. The manifest file is frequently used by third party application developers to request permission to the user's personal information, such as his current position [7].

Privacy and Security

To ensure privacy and security, Android uses a combination of application separation and access control; applications are run i sandboxes and are given individual permissions to the mobile device's resources. More specifically, Android uses *mandatory access control*, which means that the application's manifest file must include a list of requested access permissions to the resources the application intend to use. The requested permissions will then be presented to the user upon installation, which must either accept or reject the entire set of permissions. If the list is rejected, the application will not be installed [8].

From the user's point of view, there are several problems related to the access control model used in Android. First, the user has to grant permissions already upon installation of the application, often without any information on when and why the application needs this access and how the application intends to use the collected data. Once installed, the application can access the requested resources at any time, without the users knowledge. Second, as indicated above, the user have no means to reject some of the requested permissions and accept others [8]. In addition, in our opinion, the requested permissions are often difficult to understand. Even though many applications have a privacy policy, which explains the intended use of collected data, this is not mandatory. The users often have to search the web on their own initiative in order to find it.

3 Design

In this section we propose a design that aims to solve the identified problems associated with the mandatory access control in Android. The main objectives of our design are to:

- Allow users to continuous control the sharing of personal information
- Extend the granularity of the access permissions to include a limited amount of the requested permissions rather than "all or nothing".
- Give the user an explanation of the possible consequences of granting a third party application access permission to personal information.

Our proposal is a middleware that consists of a *Privacy Service* and a *Privacy Application*. Using Privacy Application users can set their preferences for sharing personal information (i.e. access to resources) for each individual application installed on the device. Privacy Service will then enforce the preferences by overriding the requested permissions in the applications' individual manifest files. The main reason for choosing a middleware solution (that has the obvious drawback that it requires modifying the applications to use the middleware) was to make it as simple as possible to test the proposed concepts in a few prototype applications.

To demonstrate the intended use of our design we use a simple scenario.

> *Bob uses an Android smartphone in both business-related as well as private matters. Lately, Bob has become increasingly concerned about privacy and*

feels he has lost control over the third-party applications on his phone. Bob therefore downloads and installs the Android middleware "Privacy Service", which includes the application "Privacy Application". When Bob opens the Privacy Application it displays a list of all third party applications installed on his phone and their corresponding access permissions.

The purpose of Privacy Application it to help the user control what application can access what personal data, to control the granuality of the sharing of personal information and to understand the risk of sharing too much information with untrusted third parties.

Bob has previously installed an app called MyDailyNews, which feeds him with fresh news every day. Using Privacy Application, Bob sets the location granularity for MyDailyNews to "No location". However, when he opens MyDailyNews the next time, the expected list of news is empty and a message tells him to check if his GPS is turned on. Bob realizes that he has to share his location to be able to use MyDailyNews, but he does not want to share his exact location. A compromise has to be made. Using Privacy Application, he clicks on MyDailyNews and then on "fine (GPS) location" to get further options. A new window reveals itself, and present Bob with detailed information about the "fine (GPS) location" permission. By reading the text Bob understands the possible risks involved in sharing his exact position. Since he rarely read the local news he concludes that MyDailyNews only needs to know his current state and country. He clicks on the "settings" button and marks the checkbox "state". When he opens the MyDailyNews application the next time, he receives regional and national news.

From the user's perspective, there are two main advantages of using Privacy Application. First, the information window gives the user a better understanding of the requested access permissions. Secondly, by navigating to the settings menu, the user can choose to share "some" information, avoiding the "all or nothing" pitfall.

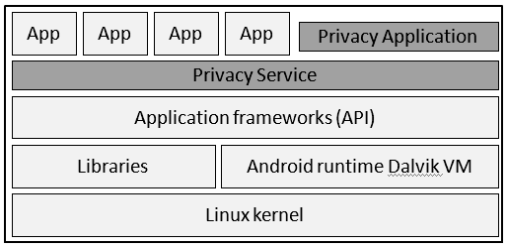


Figure 1: Android software stack including the Privacy Service middleware.

Our design is illustrated in Figure 1. The Privacy Application is used to interact with the user, while the Privacy Service runs silently in the background. The middleware is placed between the applications and the API in the software stack. By introducing this scheme, calls either from or to the API will be handled by the middleware. When Privacy Application is started it displays a list of all applications installed on the phone. The user will then have the possibility of overriding the access permissions granted to the applications upon installation.

The key idea in our design is the use of *data separation*. For some applications, as an alternative to the "all or nothing" approach, it might be useful to only provide access

to a subset of data, or to limit the accuracy of the provided data. The ability to provide "some" data means that the user does not have to trade functionality for a good privacy (an example is the location-based service MyDailyNews described in the scenario above). To start, we have proposed data separation for six different resources, listed in Table 1. The table briefy describes the access permissions and data separation options related to each resource.

Resource	Access permission description	Data separation options
Location	Allows an application to access the device's location using e.g., its WiFi or GPS.	Give the user the opportunity to choose the accuracy of the location.
Internet	Allows an application to open network sockets.	Open a limited Internet access through a filter, or prevent the application from connecting to the Internet when the application is not used.
Calendar	Allows an application to read the user's calendar data.	The user can choose what events to share (e.g. only public events).
Contacts	Allows an application to read the user's contact data.	The user can mark what contacts to share and what to hide.
Accounts	Allows access to the list of accounts in the Accounts Service	The ability to choose what accounts that are visible to the application.
Storage	Allows an application to write to external storage.	Defines a new folder inside the external storage with specific read, write and delete options

Table 1: The proposed data separation options in our design.

Fig. 2 shows the information flow between three of the different Privacy Application user interfaces[2] and a database, after the user has clicked on the access permission "fine (GPS) location" (GUI 1). The hyper-link opens a new window with a short explanation of the chosen access permission (GUI 2). The reason for having this interface is to give the user a better understanding of the risk associated with granting the requested access permission. The "settings" button leads the user to a menu (GUI 3), which presents a list of possible location granularities. The settings will be saved in a database.

By querying the database, the Privacy Service can keep track of what data to return upon a request from an application. Fig. 3 illustrates the information flow between the Privacy Service and an application that requests access to the GPS. When the application sends a request to the middleware, the Privacy Service checks with its corresponding record in the database, and return the coordinates in accordance to the specified granularity.

4 Implementation

To demonstrate the functionality of our proposed design we implemented a prototype version of Privacy Service and Privacy Application on a LG-P500 touch-screen

[2]We have designed user interfaces for all six resources in Table 1. Due to space limitations we do not provide the the complete set of figures in this paper; the reader is referred to [9] for more details.

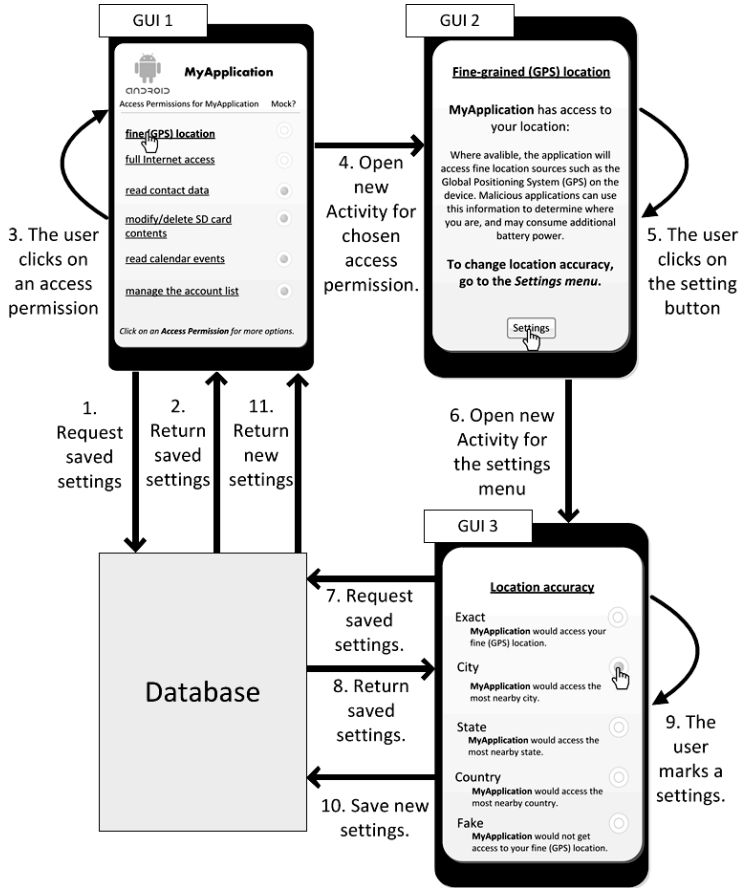


Figure 2: Communication between the Android activities and the database.

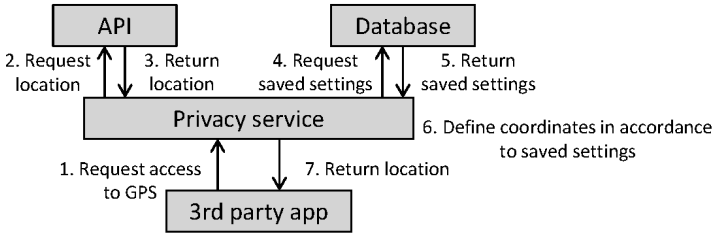


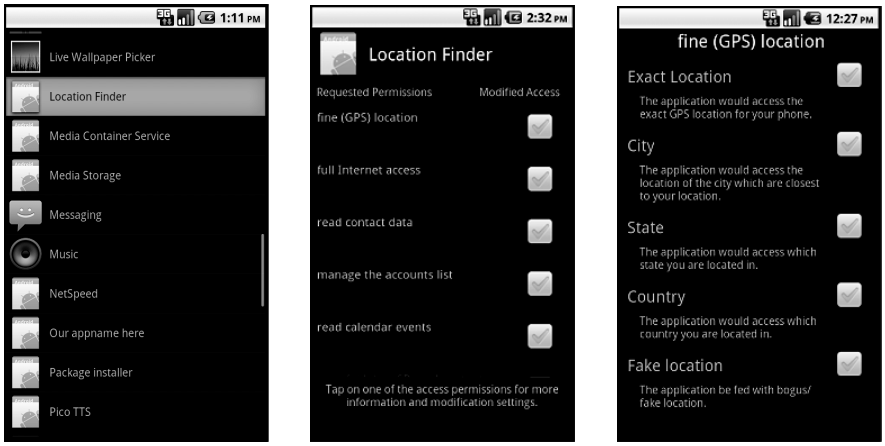
Figure 3: Communication between Privacy Service and a third party application.

smartphone, running an original Android 2.2 Froyo software stack. The implementation is described below.

Privacy Application

The prototype version of the Privacy Application consists of four activities, which are used to display GUIs. Three of these are illustrated in Figure 4. The first activity displays

a list of all installed apps on the phone (Fig. 4a). When the user clicks on an item in the list, a listener is activated, an intent is created and another activity is pushed to the top of the activity-stack.



(a) List of all installed applications (b) List of permissions requested in the application's manifest file (c) Interface showing coarseness of GPS location

Figure 4: Three activities (GUIs) in the Privacy Application

The second activity (Fig. 4b) shows the requested permissions for the chosen application. Each row in the GUI consists of the textual meta data description of the requested permissions for the application, obtained from the manifest file. In addition, we implemented a checkbox associated with each permission. To check the current state of the checkbox, we make a call to a content provider (described further down in this section) to check whether the modified access-tag have been set previously or not.

There are two listeners associated with each row in the activity shown in Fig. 4b; one for the requested permission and another one for the checkbox. The listener connected to the textual description is starting a third activity that simply gives the user a more thorough description of that permission. In addition to an explanation, the activity also have a "settings" button who starts the fourth activity (Fig. 4c), which lets the user change these permissions. The state of each checkbox in this activity is also fetched from our content provider.

To communicate the user preferences between the Privacy Application and the Privacy Service it is necessary to save activity state. To mitigate the possible risks related to sharing of private data we utilized a content provider for this purpose. A content provider exposes read and/or write access to any private data of an application, dependent on whatever restrictions one want to impose for it. None of the content providers already shipped with Android were suitable for our middleware so we decided to design and implement our own (an SQLite database), whose structure is illustrated in Table 2. In our implementation this content provider is created and broadcasted by the Privacy Service middleware.

As can be seen in Table 2, the database in our content provider currently contains access permissions for the phone's GPS location, contact data and Internet access. A "1" in the FAKE column indicates that the Modified Access checkbox have been marked (see Figure 4b). The number in the FAKE-SETTING column indicates the coarseness of

ID	Access Permission	FAKE	FAKE-SETTING
1	F-GPS (fine GPS location)	1	4
2	RCA (read contact data)	0	0
3	FIS (full Internet access)	0	0
...			

Table 2: The structure of our content provider.

the returned value given from our middleware. For example, for the GPS location "0" indicates that none of the options is chosen, "1" indicates the exact location, "2" indicates city, and so on (see Figure 4c).

Privacy Service

We implemented Privacy Service as an Android service component in order to handle the interaction between the Privacy Application and other applications. Using a service (rather than an activity) lets our middleware constantly listen for requests. The Privacy Service is triggered by an intent message. An intent filter in the manifest file defines what intents the Privacy Service should listen for. The service listens for both intents sent internally in the middleware and catch intents sent from other applications. In our current implementation Privacy Service can be triggered by two actions; one that starts the service and another that triggers the service to fetch and return the current GPS location to the requesting application.

To obtain the user preferences defined using Privacy Application, Privacy Service use a content resolver, which fetches the data from the content provider database (Table 2). For example; to obtain the user's current location in accordance to his stated preferences Privacy Service does the following. First, it queries the database to obtain the ID of the access permission "fine (GPS) location". This ID is then used to get the corresponding FAKE SETTING. If the FAKE SETTING is 0 or 1, the coordinates will be set to the exact location. If the setting is 5, the coordinates will be set to [0.00,0.00]. If the setting is 2, 3 or 4, the Privacy Service will make a call to Google Maps[3] to find the address, city, state and country that correspond to the provided coordinates. Dependent on the accuracy defined in the Privacy Application, the Privacy Service would run a new method named getFromLocationName(String name, int maxResults). The results are broadcasted system-wide by an intent that will be caught by the application that requested it.

Test Application

To be able to test the prototype we implemented a simple Android app that we named "Location Finder" that could interact with the Privacy Service middleware. Location Finder uses the mobile device's current location to provide the user with a list of URLs to online newspapers ordered according to their geographical vicinity. When Location Finder is started, an intent with the action "LOCFINDER GPS REQ" is sent to the Privacy Service. Privacy Service can then fetch and return the coordinates as explained in the example above.

In order for the Location Finder to obtain the returned coordinates from the Privacy

[3]http://maps.google.com

Service, a *broadcast receiver*[4] is needed. In addition, a reciever tag have to be included in its manifest file. When the Privacy Service broadcasts the intent including the action "LOCFINDER GPS LOC", the broadcast receiver in Location Finder is able to obtain the coordinates. The manifest file and the GUI for Location Finder can be found in [9].

5 Evaluation

To verify our proposed design we tested the prototype in two steps. First, we performed a descriptive test that verified that the implementation behaved as expected. The details of this test are described in [9]. Second, we performed an experimental test where we studied the proposed solution in a controlled environment in order to evaluate its usability and usefulness.

Preparing the Experimental Test

The purpose of usability testing is, according to Bevan and Macleod, "to ensure that the delivered product reaches a minimum required level of usability, to provide feedback during the design on the extent to which the objectives are being met, and to identify potential usability defects in the product" [10]. The goal of our test was to improve our design and to discover, at an early stage, if there were some parts of the design the users did not understand. We also aimed to find out whether the users found our design useful.

The test group consisted of five university students with a good general knowledge of Android, privacy and security, and computer science. The reason for choosing "expert users" was that we intended to test not only usability but also the usefulness of the design. If the participants in our group, who we knew had an over-average interest in privacy and security, did not find the design useful, this would be a strong indication that no one else would. We are fully aware that we have to perform a test with a larger group of regular users, before we can make any general conclusion about the usability and usefulness of our design.

The Test Procedure

The full details from test procedure is described in [9] but we summarize it briefly here. We used the 10-step procedure for usability testing described by Tognazzini [11] as a guide for carrying out the test. All participants were provided with an Android phone with Privacy Application, Privacy Service and the test application (Location Finder) already installed. The test consisted of six practical tasks where the participants experimented with different access permissions for the GPS location used by the test application. The participants then answered questions related to usability and usefulness.

Test Results

To evaluate the usability of the design we analyzed the participants' behaviour when solving the tasks. The test group had no problem understanding how the Location Finder application worked. However, two of the participants found it difficult to navigate from Location Finder application to the "settings menu" in the Privacy Application. In

[4] A broadcast receiver is an Android abstract class created to receive broadcast messages.

addition all the test participants had problems understanding the activity where the access permissions were listed (Figure 4b). Also, some of the participants did not understand the meaning of "fake location". This was a clear indication of that the GUI of our prototype needs some improvements, to become more intuitive and user friendly.

To evaluate the usefulness of the design we analysed the participants' answers to the questions. All the participants stated that they preferred a design like ours rather than the existing "all or nothing" approach to personal information sharing. They especially valued the possibility of faking their location. They all considered the middleware useful, however, not useful enough for them to download and install on their own initiative. In fact, several of them stated that they did not really care about what kind of data the applications on their phones collect. The general opinion amongst the participants was that a major breach in privacy had to occur before they would care to take actions.

6 Discussion

Limitations

The work described in this paper has some limitations. First, our current implementation only supports a location-based service. The reason is that we wanted to evaluate the value of the design before implementing all of it. This would make it easier to abandon the project at an earlier stage, if necessary, and focus on alternative solutions.

Second, our initial plan was to modify the Android operating system, in order to control the applications' access to resources. However, we experienced several practical difficulties (described in [9]), which forced us to abandon this idea and go for the more lightweight implementation described in this paper. An advantage is that it is less likely that a user would have second thoughts on installing a third-party application compared to installing patches that modifies the core components of the OS. Since users in general often seem to prefer a seamless "plug-and-play" experience, the use of a middleware application seems to be a more valid approach. On the other hand, from a technical point of view, forcing changes to applications by modifying the environment where they reside may be a better solution. Both approaches have their pros and cons.

Finally, if the middleware solution turns out to be the preferred solution, the question of how to enforce applications to interact with a middleware arises. Our prototype is based on the assumption that the applications send their access requests to the middleware, and not to the resource directly. We have not attempted to solve this issue in this paper.

The Test Results

The results from the tasks performed in the test revealed that parts of the user interface were not intuitively adequate. We have identified several concrete measures that should be taken in order to improve this and these are further described in [9]. Another aspect we aimed to investigate was whether the solution was useful or not. We were surprised by the results from the test, since we had expected the participants to be more concerned about the privacy risks related to third-party applications than they turned out to be. Their opinion that a major breach in privacy had to occur before they would care to take actions is likely to be shared by other users as well. However, as pointed out in the previous section, since the test group was so small (only five persons) we cannot generalize our findings.

Security

When developing middleware it is very important to maintain security. The middleware is responsible for major amounts of sensitive data. The users should be confident that the personal information accessible to the middleware is secured. Since our prototype only may be regarded as a proof of concept, intended to demonstrate the proposed concepts, we have not taken specific measures to ensure the security of the implementation. While our proposed design does not introduce any obvious vulnerabilities in itself, we see some potential security problems with our implementation. In Section 3 we described the concept of broadcasting intents to exchange messages between the middleware and third-party applications. Since the intents may include personal data, it is important that the broadcasted message is only received by the intended recipient. An possible way to make broadcasting intents more secure, is to require recipients to have a *receiver permission* in their manifest file [4]. If the Privacy Service broadcasts an intent meant for e.g., the Location Finder application, it can make sure that only the Location Finder has the required permission to receive it. Even though using receiver permissions makes broadcasting more secure, it may still be insufficient. As far as we are concerned, there might be other, and better, solutions in order to make communication more secure. Security issues are utterly important to address in future development.

Related Work

Our work has been inspired by MockDroid [8], which is a modified version of the Android operating system that allows a user to "mock" (i.e. fake) an application's access to resources. With MockDroid, the user is still able to use third party apps, though with the lack of some functionality that is dependent on mocked data. One of the most important modifications is that the set of access permissions is duplicated so that each permission has both one real and one mocked version.MockDroid shares the same benefits as our design; it allows the user to control the collection and distribution of personal data, which is a great contribution to privacy on mobile devices. However, as the title implies, providing mocked data means that the applications may loose important functionality. For most users, it might be a problematic decision whether to sacrifice functionality for privacy. Compared to MockDroid, our solution gives the users a higher degree of freedom in cases where it may be hard to choose between real and faked data. Other relevant work in the same field is the TaintDroid project [12] and the TightLip project [7].

7 Conclusion

This paper presents a tool that allows users of mobile devices to control what kind of personal information third party applications ("apps") are able to access. The design presented in this paper is based on a middleware solution. The main idea is a shared and centralized application that monitors and controls requests to internal resources, sent on behalf of an application. To demonstrate its functionality and evaluate its usability and usefulness we implement a prototype that lets the user control the sharing of location-based data (GPS).

Our main objective was to help users protect their privacy by increasing their consciousness on how personal information is collected and distributed. By controlling the amount of information to share, users should get an increased consciousness on

how personal information is collected and distributed. The test results indicate that our approach is promising but that user motivation may unfortunately be lacking.

Even though our solution is not complete, we believe that the results obtained from our study serve as a contribution to the field of privacy protection for mobile devices.

References

[1] M. Bonetti, "Mobile privacy: Tor on the iPhone and other unusual devices," *DEFCON 18*, May, 2010.

[2] The Wall Street Journal, "Your Apps Are Watching You," Dec 17 2010. [Online]. Available: http://online.wsj.com/article/SB10001424052748704694004576020083703574602.html

[3] GSMA Mobile Privacy, "Mobile Privacy Principles ," 2011. [Online]. Available: http://www.gsmworld.com/our-work/public-policy/mobile_privacy.htm

[4] The Android Open Source Project, "Android," 2011. [Online]. Available: http://source.android.com/

[5] Canalyse, "Googles Android becomes the worlds leading smart phone platform," Jan 31 2011. [Online]. Available: http://www.canalys.com/pr/2011/r2011013.html

[6] Android Developers, "Application Fundamentals," 2011. [Online]. Available: http://developer.android.com/

[7] A. Yumerefendi, B. Mickle, and L. P. Cox, "Tightlip: Keeping applications from spilling the beans," *NSDI 2007, 4th USENIX Symposium on Networked System Design and Implementation, Cambridge, MA*, April, 2007.

[8] A. R. Beresford, A. Rice, N. Skehin, and R. Sohan, "Mockdroid: trading privacy for application functionality on smartphones," *HotMobile '11, Phoenix, AZ, USA*, March, 2011.

[9] S. Pedersen and S. Bø, "Privacy Services for Mobile Devices. Master Thesis, Norwegian University of Science and Technology (NTNU)," 2011.

[10] N. Bevan and M. macleod, "Usability measurement in context," *Behaviour and Information Technology, Chapter 13, Page 132-145*, 1994.

[11] B. Tognazzini, *Tog on interface*. Addison-Wesley Professional, 1991.

[12] W. Enck, P. Gilbert, B. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, "What you see is what they get: Protecting users from unwanted use of microphones, cameras, and other sensors," *The USENIX Symposium on Operating Systems Design and Implementation (OSDI),Vancouver*, October, 2010.