RADBOUD UNIVERSITY NIJMEGEN

MASTER THESIS
COMPUTING SCIENCE

Fair Privacy: Improving Usability of the Android Permission System

Author:
Maarten DERKS

Supervisors:
dr. Jaap-Henk HOEPMAN
drs. Fabian VAN DEN BROEK
dr. Veelasha MOONSAMY (reader)

External supervisors:
Kees JONGENBURGER
Dirk VOGT

August 2015

FAIRPHONE

Radboud University

Abstract

Research has shown that users have a hard time comprehending and assessing the Android permission system. Users tend to ignore the permissions dialog during application install. The users that do pay attention have difficulties comprehending the permission descriptions and assessing what impact applications will have on their privacy. Another issue of concern is intransparency: once the user grants the permissions, an application has unlimited access to them, but the user has no insight into how applications are actually using the requested permissions. These problems lead to badly motivated security decisions that impact the privacy of the user in a negative way.

This research aims to help users understand what is happening with their data. The goal is to present this information to the user so he can be informed about what applications are doing and help him better understand the implications, and provide him with rationales to investigate alternatives.

The research was broken down into three projects. The first project analyzes the permissions an application requests and translates them into an easy to comprehend graphic element: the Privacy Impact. Usability tests showed that the Privacy Impact did trigger users to think about application privacy, as opposed to the permissions dialog in Google Play. The second project addresses the issue of intransparency. I modified the Android framework so that it records permission usage and present it to the user for review. This could lead to the discovery of overprivileged applications or permission usage while the phone is on standby. The third and final project explores various ways of encouraging users to discuss the privacy impact of the applications that they use, and stimulate them to explore more privacy-friendly alternatives.

The research described in this thesis was conducted at Fairphone, a social enterprise that aims to develop a sustainable smartphone.

Acknowledgements

This thesis forms the final step in obtaining a Master's degree in Computing Science from the Radboud University in Nijmegen. The work described here would not have been possible without the advice and support of many people, so I would like to take this opportunity to thank all of them.
First and foremost, I would like to thank Jaap-Henk Hoepman for suggesting me to write my thesis at Fairphone, and Olivier Hebert and Kees Jongenburger for giving me the opportunity to actually do so. During the course of this research Kees's supervision proved to be invaluable in guiding me in the right direction. The bi-weekly meetings with Fabian van den Broek at Radboud University helped me to secure the academic aspects of this research, for which I owe him many thanks as well.
I would also like to thank the rest of the software team at Fairphone for their cooperation, especially Rick for working together on usability testing and Dirk for taking over supervision when it was needed most. I also appreciate Kwamecorp's contributions, especially from João on the graphic design.
Last, but certainly not least, I would like to thank those who did not directly contribute to this thesis, but supported me in various ways during the last few months. Most notably they include my friends and my family. Thank you for enabling me to pursue my ambitions and providing me with encouragement when I needed it most.

Contents

Chapter 1

Introduction

Smartphones and other mobile devices have become explosively popular for personal and business use in recent years. Global smartphone shipments are expected to reach around 1.4 billion in 2015 [1], an increase of 16% over the last year. The market is currently dominated by the Android operating system, accounting for 78% of all smartphone shipments globally in the first quarter of 2015 [2].

Android supports a booming third-party application market. At the time of writing Google Play (the official Android application repository) included over 1.5 million applications [3], which have been downloaded over 150 billion times [4].

Unfortunately, the growth in the Android platform has triggered the interest of dubious application developers. Android malware harvests data or sends premium SMS messages for profit, and other unwanted applications (grayware) collect excessive amounts of personal information (e.g. for aggressive marketing campaigns). Malware and grayware have both been found in Google Play, and the amount of new malware is increasing over time [5]. Google has made efforts in the past to limit malware [6] - with limited success [7] - but Google Play still contains vast amounts of unreliable code [5].

Android also relies on the user to detect privacy- or security-invasive applications by using a *permission* system. When a user initiates the process of installing an application, he is shown the list of permissions that the application requests. This list identifies all of the phone resources that the application will have access to if it is installed. For example, an application with the `SEND_SMS` permission can send text messages, but an application without that permission cannot. If the user is not comfortable with the application's permission requests, then he can cancel the installation. Privacy violations can occur even when a user grants access to protected data (e.g. contact list) to a benign application. Applications often come bundled with malicious advertising libraries, and benign Android applications tend to request more permissions than needed for their intended functionality [8]. Users are not shown permissions at any time other than installation.

1.1 Problem statement

The smartphone platforms' security models delegate users to make security decisions while downloading applications from official application repositories. This delegation ranges from simply authorizing access to some protected resources, to letting users decide whether an application may impair his security and privacy. The survey results of Mylonas et al. [9] are not in line with the expectation of smartphones' security models for a reasonable security aware user. In contrast, the survey results suggest that users are not adequately prepared to make appropriate security decisions. Further research by Mylonas et al. [10] confirms that smartphone users require awareness training specifically tailored for smartphone security.
Kelley et al. [11] studied the Android permission system in particular, and state that users do not understand Android permissions in their current form. Their conclusion is that users are currently not well prepared to make informed privacy and security decisions around installing applications from Google Play. Studies by Felt et al. [12] indicate that Android permissions fail to inform the majority of users. They discovered that users are not able to connect resource-based warnings to risks, cannot reason about the absence of permissions, and some of them experience warning fatigue. Benton et al. [13] confirm that as users are unaware of the implications of the requested permissions, permission requests appear to be ineffective. Low rates of user attention and comprehension suggest that significant work is needed to make the Android permission system widely usable.

1.2 Research question

I created a main research question, which is broken down into several sub-questions:

Main research question

> *How to improve the usability of the Android permission system in order to increase user awareness and comprehension with respect to privacy?*

Sub-questions

- *Can Android permissions be translated into something more comprehensible?*
- *How to implement this simplification to raise awareness on privacy?*
- *Do these changes increase awareness and comprehension among users?*
- *Is it possible to visualize permission usage by applications after installation?*
- *How to start a discussion among users on the privacy impact of applications?*

1.3 Methodology

In order to answer the sub-questions, three sequential projects have been set up.

Privacy Impact

The first project (Chapter 3) addresses the issue of users not being aware of permissions and having difficulty comprehending them. Research has shown [14–18] that conclusions can be drawn from the permissions that applications request. For this research I am interested in those permissions that control access to personal data, for example to text messages or contacts. I design an algorithm that translates these permissions into a meaningful graphic representation: the Privacy Impact. This makes it easier for users to assess the privacy impact of the applications that they use. I conduct usability tests to establish the effectiveness of the Privacy Impact, in terms of both awareness and comprehensibility.

Permission Usage

The second project (Chapter 4) addresses the issue of intransparency: Android permissions are only displayed at install-time. After the user grants access to the requested permissions, the application has unlimited access to them, but the user has no insight into how the application is actually using those permissions. I added functionality to the Android framework that allows for recording of permission usage to a database. This database is used by an application to present to the user a timeline on permission usage per application. This could lead to the discovery of, for example, overpriviliged applications or permission usage while the phone is on standby.

Engaging Users

The third and final project (Chapter 5) explores various ways of encouraging users to discuss the privacy impact of the applications that they use, possibly incorporating the analyses from the first two projects. Users might question if the requested permissions by a particular application are justified, or if the price you pay (in terms of privacy) is too high. This discussion also stimulates users to explore more privacy-friendly alternatives to common applications, such as the browser or the e-mail client.

1.4 Scope

I aim to enhance the privacy of the average user while keeping things simple. The end goal is to protect and educate the user. This research will not go into more traditional security (e.g. detecting malware), but instead I focus on privacy through improving the usability of permissions. I will gather information on applications and their behavior and interaction with other applications without interfering with the default system behavior. I focus on Android 5.1, the most recent version available during the course of this research. I do not consider developer-defined permissions (that are not part of the operating system) at any stage of my analyses.

1.5 Context

The research described in this thesis was conducted at Fairphone. Fairphone is a social enterprise with the aim to develop a smartphone in the most sustainable way possible. To achieve their ambitions they focus on activities and interventions within four core action areas: mining, design, manufacturing and life cycle.

Fairphone started in 2010 as a project of Waag Society, Action Aid and Schrijf-Schrijf to raise awareness about conflict minerals in consumer electronics. The most common conflict minerals are tin, tantalum, tungsten and gold, and the mining of these minerals fuel conflict in the Democratic Republic of the Congo. Fairphone used the mobile phone, a deeply personal device that people carry with them at all times, as a storytelling device. The campaign and related research ran for three years.

In 2013 Fairphone was officially established as a social enterprise, in order to start producing actual phones. The money for the initial batch of phones was raised through pre-orders, reaching the required 5,000 in June 2013. The entire production run of 25,000 phones sold out a month before the actual release date in December 2013.

The first Fairphone was a reference model licensed from its manufacturer. Fairphone adapted it to include recycled plastics and minerals from conflict-free suppliers. They also improved working conditions and established employee representation in the China-based factory. Fairphone made it easy to repair the phone yourself by selling spare parts through the web shop and publishing repair guides. The phone shipped with the open source Android operating system, with root access enabled by default. A second batch of 35,000 phones was released for sale in May 2014, and in the beginning of 2015 all 60,000 phones were sold.

During the course of this research the development of the second Fairphone reached its final stages. In contrast to the first Fairphone, the Fairphone 2 was designed from scratch. This design approach allowed Fairphone to take their ambitions for fairness even further by improving the ability to select suppliers and increasing transparency. It also allowed them to focus on making a phone that lasts longer and gives users a stronger sense of ownership. At the time of writing pre-orders have just started, and the

phone is scheduled for release later this year.
Fairphone's ambition is to bring more fairness to software as well. To them that means focusing on three key principles: longevity, transparency and ownership. At the software level, they aim to enhance longevity by supporting their product for as long as possible. Their goals for transparency include making software as open as possible, making users more aware of the data they are sharing, and giving them more control over their applications. This because Fairphone believes that the phone is yours, and your data belongs to you.
My research project falls under on the transparency principle, and it was started with the intention of adding the results to Fairphone 2's code base.

1.6 Thesis outline

The remainder of this thesis is organized as follows. I start off by describing the background and related work in Chapter 2. This includes defining (smartphone) privacy, as well as a detailed description of Android's permission system. The related work section focuses on user awareness and comprehension of permissions, as well as deriving conclusions from application permission requests. I continue by detailing the Privacy Impact project in Chapter 3, following the development process from user stories to the final implementation. The second project, providing transparency on permission usage, is discussed in Chapter 4, including the proof of concept I developed. Chapter 5 addresses the third and final project: how to start a discussion among users on the privacy impact of the applications that they use and stimulate them to search for alternatives. In Chapter 6 I discuss the results of this research and the announcement of the next Android version. Finally, in Chapter 7 I draw conclusions and discuss directions for further research.

Chapter 2

Background and Related Work

In this chapter I provide an overview of (smartphone) privacy. I also detail the Android permission, both from a technical and a user's point of view, as well as ways to control permissions. I also present relevant literature on the effectiveness of Android permissions and on analyzing permission requests.

2.1 Privacy

Privacy is a broad concept that captures various aspects of ordinary life. Therefore several conceptualizations of privacy exist. Westin [19] already made a classical distinction in 1967, differentiating between various spheres of privacy: solitude, intimacy, reserve and anonymity. A more recent definition of privacy presented by Solove [20] captures the most relevant dimensions that are usually found:

1. the right to be let alone (Warren and Brandeis's famous formulation for the right to privacy [21]),
2. limited access to the self (the ability to shield oneself from unwanted access by others),
3. secrecy (the concealment of certain matters from others),
4. control over personal information (the ability to exercise control over information about oneself),
5. personhood (the protection of one's personality, individuality and dignity), and
6. intimacy (control over, or limited access to, one's intimate relationships or aspects of life).

Information privacy

Hoepman and van Lieshout [22] look at privacy from an information and communication technology-perspective. The focus is then primarily on informational dimension of privacy: the relationship between collection and dissemination of data, technology, the public expectation of privacy, and the legal and political issues surrounding them. For example, web users may be concerned to discover that many of the websites which they visit collect, store, and possibly share personally identifiable information about them. In this context privacy can be defined as the right to control the release of personal information about oneself, even when that data is collected and stored by a third party. It is important to understand that privacy is not the same as data security, although the latter can help in protecting personal information from unauthorized access and use. The right to privacy also states that personal data is only collected when necessary, that no more personal data is collected than needed, and that this data should only be used for the purpose for which it was originally collected. Also, people have the right to view and update personal information held by others about themselves.

Smartphone privacy

Smartphones have features of both a mobile phone and a computer, allowing people to talk, text, access personal and work e-mail, browse the Internet, take pictures, make purchases and manage bank accounts. They are becoming capable of doing more and more every day, with new functionalities such as fingerprint identification and contactless payments. Unlike many computers, smartphones are always with us and many people rarely turn them off. For these reasons users need to be aware of the kind of information that can be collected by various entities from their smartphone.
Mobile network operators (MNO's) such as T-Mobile and Vodafone collect data, for instance for billing and performance measurement purposes. But data collection by MNO's also takes place because of legislation (referred to as *data retention*), so that governments can access the data for traffic analysis and mass surveillance. Such legislation is currently on hold in Europe. The details of what MNO's are collecting are usually not clear, but they may include the following:

- Date and time for every communication:
  - Starting or receiving phone calls;
  - Sending or receiving text messages;
  - Internet access.
- The initiator or recipient of the communication;
- Location.

But data collection by MNO's is not the primary target in this thesis. In addition to the data collected by MNO's, users should also be aware of the possible privacy issues surrounding the collection or disclosure of:

- Photos or video taken on their phone;
- Text messages and e-mails sent and received on the device;
- Who called them, who they called, when it was placed and how long it lasted;
- The contacts stored in their phone;
- Passwords;
- Financial data;
- What is stored in their phone's calendar;
- Their location, age, and gender.

Companies, advertisers, cybercriminals and even federal agencies would have an interest in the data stored in a smartphone. Applications can collect all sorts of data and transmit it to the application developer or sell it to third-party advertisers. Advertisements from advertising networks running on some applications may change smartphone settings and take contact and other information without the user's explicit permission.
Some applications might track your location: location-based services like Google Maps, Foursquare or Tinder need your location in order to function properly. However, there are applications that do not need location to function but may still be tracking it. Applications may also be infected with malware. Many mobile applications do not have privacy policies, and when they do, they are often long and difficult to understand.

2.2 Android permissions

To protect Android users from unwanted access to their personal data by applications, access to phone resources is restricted by using *permissions*. An application must obtain permissions in order to use sensitive resources like the camera, microphone, or call log. For example, an application must have the `READ_CONTACTS` permission in order to read entries in a user's phonebook. Obtaining permissions is a two-step process. First, an application developer declares that his application requires certain permissions in a file that is packaged with the application. Second, the user must approve the permissions requested before installation. Each application has its own set of permissions that reflects its functionality and requirements. Users can weigh the permissions against their trust of the application and personal privacy concerns.
Permissions are part of Android's overall security model, which is largely based on the

underlying Linux kernel. I will not address the Android security model here, but it is important to mention that each application runs under its own user ID (UID). This is the fundamental construct for application separation. Interprocess communication (IPC) between applications is accomplished using the Binder interface and *Intents*, message objects containing data and a destination component address. Application permissions allow components to control which other applications are allowed to interact with them.

2.2.1 Overview

Android 5.1 defines 152 permissions, categorized into 31 permission groups (see Appendix A). The permissions fall into four protection levels:

- **Normal** - These permissions cannot inflict real harm to the user (e.g. change the wallpaper) and, while applications need to request them, they are automatically granted.
- **Dangerous** - These can inflict real harm (e.g. open Internet connections, call phone numbers, etc.) and applications need to request them with user confirmation.
- **Signature** - These are automatically granted to a requesting application if that application is signed by the same certificate (so, developed by the same entity) as that which created the permission. This level is designed to allow applications that are related (e.g. part of a suite) to share data.
- **Signature/System** - Same as Signature, except that the system image gets the permissions automatically as well. Designed for use by device manufacturers only.

Making use of a system API that requires a permission requires a developer to specify that permissions in their application's manifest (AndroidManifest.xml). For example, if an application needs access to the Internet, specify the `INTERNET` permission:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.example.app.test">
        ...
        <uses-permission android:name="android.permission.INTERNET" />
        ...
</manifest>
```

Application developers can also define custom permissions for their applications. Developers can restrict access to various components and services for the purpose of self-protection. Any application that interacts with another application's component would need to possess the required permission for the call to succeed. A custom permission must be declared in an application's manifest.

2.2.2 Enforcement

Applications can dynamically check to see if calling applications have the appropriate permissions to interact with certain components. Permissions can also be enforced by specifying them within the containing application's manifest. This method is generally preferred over dynamic checks, as permissions are more easily managed by including them all in a central place and decoupled from the application's source code. Each component type can specify required permissions using attributes in their manifest entries [23, 24]. These components will be used in Chapter 4 to build an application that monitors permission usage by applications.

Activities. An *Activity* is a component that defines an application's user interface. Typically a developer creates one activity per 'screen'. Permissions can be used to restrict what applications can cause an Activity to start by including a `permission` attribute in the manifest entry for the Activity in question:

```
<activity android:name=".Activity1"
                  android.permission="com.example.perm.START_ACTIVITY">
          <intent-filter>
                  ...
          </intent-filter>
</activity>
```

Services. A *Service* is a component that runs in the background and provides services to other components. Permissions can be used to restrict what applications can interact with a Service (creating an instance of it, binding to it, or calling methods on it) by including a `permission` attribute in the manifest entry for the Service in question:

```
<service android:name=".MailListenerService"
         android:permission="com.example.perm.BIND_TO_MAIL_LISTENER"
         android:enabled="true"
         android:exported="true"
         <intent-filter></intent-filter>
</service>
```

An example of a Service living inside the Android framework is AppOpsService, which controls the granting of permissions to application operations.

Content Providers. A *Content Provider* is the standard way for Android applications to make data available to other applications. It is normally used as the front end for a database or similar store. Content Providers are not accessed via Intent-based IPC, but via connections to uniform resource identifiers (URI's) that point into a Content Provider namespace (e.g., `content://com.example.app.provider/`). Content

Providers can specify permissions required to either read or write them (so write does not imply read):

```
<provider android:name="com.example.app.provider"
          android:authorities="com.example.app.provider"
          android:readPermission="com.example.perm.DATABASE_READ"
          android:writePermission="com.example.perm.DATABASE_WRITE">
</provider>
```

This basic level of permissions would require read/write permissions to the entire Content Provider to be granted. As Content Providers typically expose access to an underlying database, this is not ideal, as these levels do not allow for fine-grained access control to possibly sensitive data in the Content Provider. This is why *URI permissions* are also available, that allow permissions to be granted to specific URI's within the provider (e.g. `content://com.example.app.provider/table1`).
System Content Providers (such as the one for contacts) are installed as standalone applications, separate from the system process and API library. They are protected with both static and dynamic permission checks, using the same mechanisms that are available to applications to protect their own Content Providers [8].

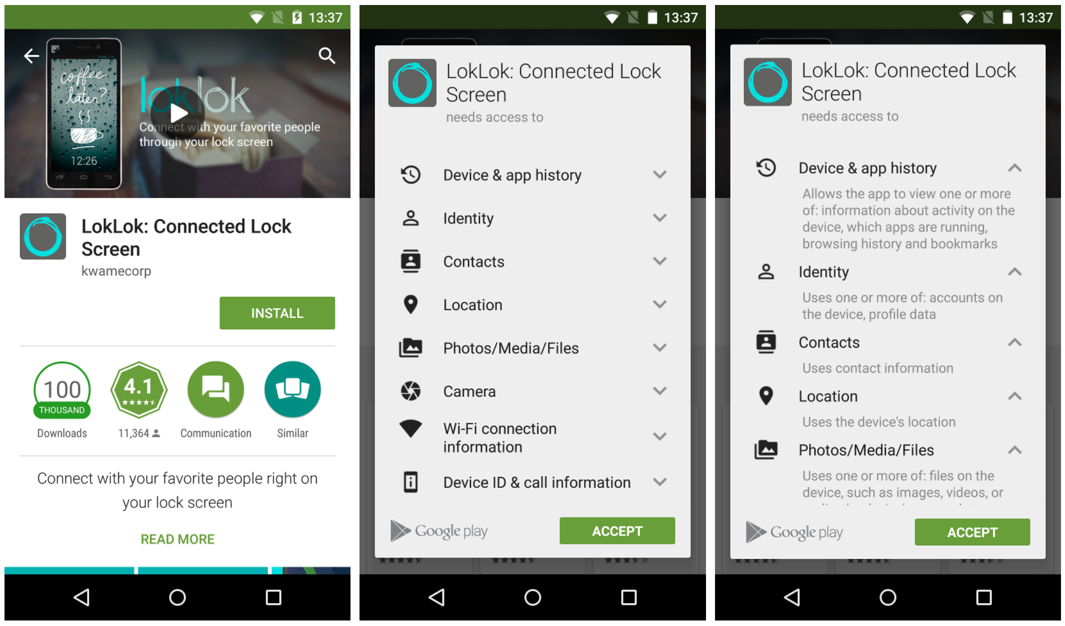
Broadcast Intents. *Broadcast Intents* are messages sent out across the system, to all applications that would like to receive them. A developer can restrict which applications are allowed to receive a broadcast by requiring a permission to do so:

```
Intent bcastIntent = new Intent(MESSAGE_RECEIVED);
context.sendBroadcast(bcastIntent, "com.example.perm.MSG_NOTIFY_RCV");
```

Developers can also configure a broadcast receiver to only receive broadcasts from applications that have a permission:

```
IntentFilter intentFilter = new IntentFilter(MESSAGE_RECEIVED);
UIMailBroadcastReceiver rcv = new UIMailBroadcastReceiver();
context.registerReceiver(rcv, intentFilter,
                             "com.example.perm.MSG_NOTIFY_SEND", null);
```

The two-sided nature of Broadcast Intents, broadcasting them out and listening for them, allow developers to properly secure both ends. By requiring a permission to receive specific Broadcast Intents, an originator can specify which applications are allowed to be notified when specific messages are broadcast. By requiring a permission from Broadcast Intent senders, applications that are listening for broadcasts can choose to only accept such messages from specific applications.



LokLok: Connected Lock Screen
kwamecorp
INSTALL
100 THOUSAND Downloads
4.1 11,364
Communication
Similar
Connect with your favorite people right on your lock screen
READ MORE

LokLok: Connected Lock Screen
needs access to
Device & app history
Identity
Contacts
Location
Photos/Media/Files
Camera
Wi-Fi connection information
Device ID & call information
Google play
ACCEPT

LokLok: Connected Lock Screen
needs access to
Device & app history
Allows the app to view one or more of: information about activity on the device, which apps are running, browsing history and bookmarks
Identity
Uses one or more of: accounts on the device, profile data
Contacts
Uses contact information
Location
Uses the device's location
Photos/Media/Files
Uses one or more of: files on the device, such as images, videos, or
Google play
ACCEPT

FIGURE 2.1: This figure shows the workflow for installing applications and viewing application permissions. Screen 1 shows the LokLok application as displayed in Google Play. If a user clicks on "Install" they are shown Screen 2, which allows him to accept the permissions and install the application. The user can also expand the permissions to reveal more information, as displayed in Screen 3.

Not all system resources (such as the camera, microphone and network) are accessed through components [24]. Instead, Android provides direct API access. The services that provide indirect access to hardware often use APIs that are also available to third-party applications. Android protects these sensitive APIs with additional permission label checks: an application must declare a corresponding permission label in its manifest file to use them. The most commonly encountered protected API is for network connections: if an application requires Internet access, it must declare the `INTERNET` permission label.

2.2.3 User interface

Google Play provides every application with two installation pages. The first installation page includes the amount of downloads, user reviews, application category and similar applications. There is also a description (with the option to "Read more") and some screenshots. After pressing the "Install" button the user arrives at a final installation page that includes the application's requested permissions (Figure 2.1). Permissions are displayed as a two-layer warning: a large heading that states each permission's general category, and an expandable label that describes the specific permission. If an application requests multiple permissions in the same category, their labels will be grouped together under that category heading. The user is not forced to review all requested permissions, he can click "Accept" without scrolling through the entire list. Users can either decide to accept the requested permissions and proceed, or they can cancel the installation.

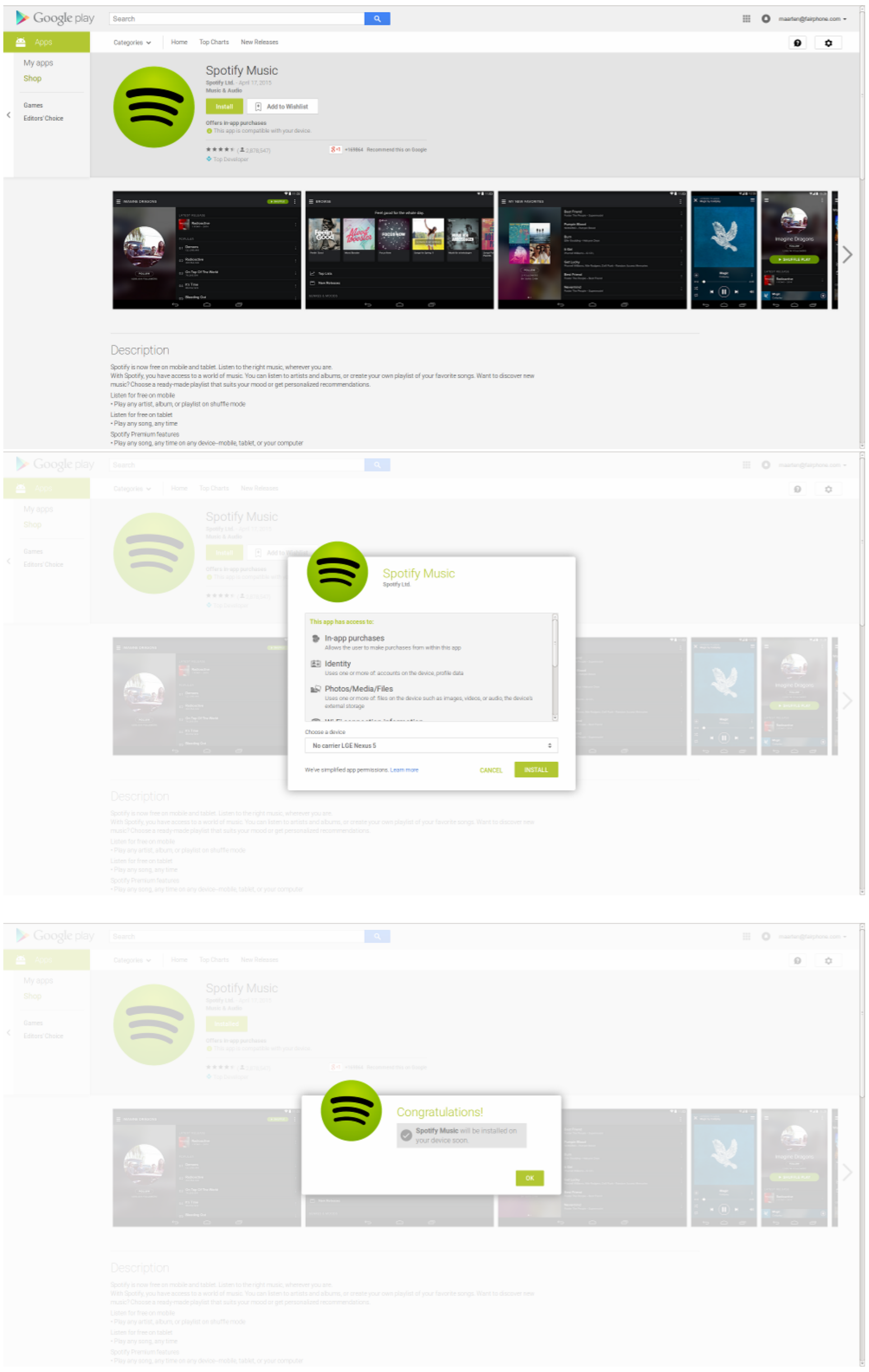
Applications can also be installed remotely through the Google Play website. Its application installation page (Figure 2.2) contains the same elements as the Google Play application. After pressing the “Install” button a dialog appears that includes the application’s requested permissions and the device you want to install the application to. After accepting the permissions the user is notified of the fact that the installation will start shortly. No further action on the phone side is required, it will download and install the application without any interaction from the user [25].

Users can also download applications through non-Google stores such as F-Droid,[1] a Free and Open Source (FOSS) Android application repository. When a user installs an application through an unofficial store, that store might not present permission information. However, Android’s installation system will always present the user with a permission page before the application is installed on the phone (Figure 2.3). Like the final installation page in Google Play, the installer displays permissions as a multi-layer warning: a heading that states each permission’s category, a label that describes the specific permission and a hidden details dialog. If an application requests multiple permissions in the same category, their labels will be grouped together under that category heading. If a user clicks on a permission, the details dialog opens. In contrast to application installations through Google Play, the user has to review all permissions before the Install button appears. It is currently unknown how Android determines from which source an application comes, and thus which installation process it has to invoke. Application updates might require new permissions. If an update requests permissions in a category that has no permissions granted yet, Google Play will ask the user to accept these new permissions. But new permissions in a category that already has permissions granted to it will not be displayed [26]. This allows developers to silently add permissions to application updates, a feature available since last year’s Google Play overhaul. Reddit user iamtubeman demonstrated this by conducting an experiment [27]. He developed an application with a particular set of permissions and published it through Google Play. He then developed a new version with additional permissions, all out of the same groups as the ones in the first version (Figure 2.4). Installing the update through an unofficial application repository does display the new permissions, similar to Figure 2.3.

2.2.4 Control

The Android permission system offers users a binary choice: either they accept the requested permissions and obtain the application, or the installation is canceled. Out of the box Android does not allow for fine-grained permission control. The reason why such control is needed is because many applications have a poorly defined set of permissions. There are cases where an application requests a particular permission but never actually uses it (so the application is *overpriviliged*), which raises security issues. But the user does not know this, and there is no way to tell if an application is actually

[1] https://f-droid.org/



Description

Spotify is now free on mobile and tablet. Listen to the right music, wherever you are.
With Spotify, you have access to a world of music. You can listen to artists and albums, or create your own playlist of your favorite songs. Want to discover new music? Choose a ready-made playlist that suits your mood or get personalized recommendations.
Listen for free on mobile
• Play any artist, album, or playlist on shuffle mode
Listen for free on tablet
• Play any song, any time
Spotify Premium features
• Play any song, any time on any device–mobile, tablet, or your computer

Spotify Music
Spotify Ltd.

This app has access to:

In-app purchases
Allows the user to make purchases from within this app

Identity
Uses one or more of: accounts on the device, profile data

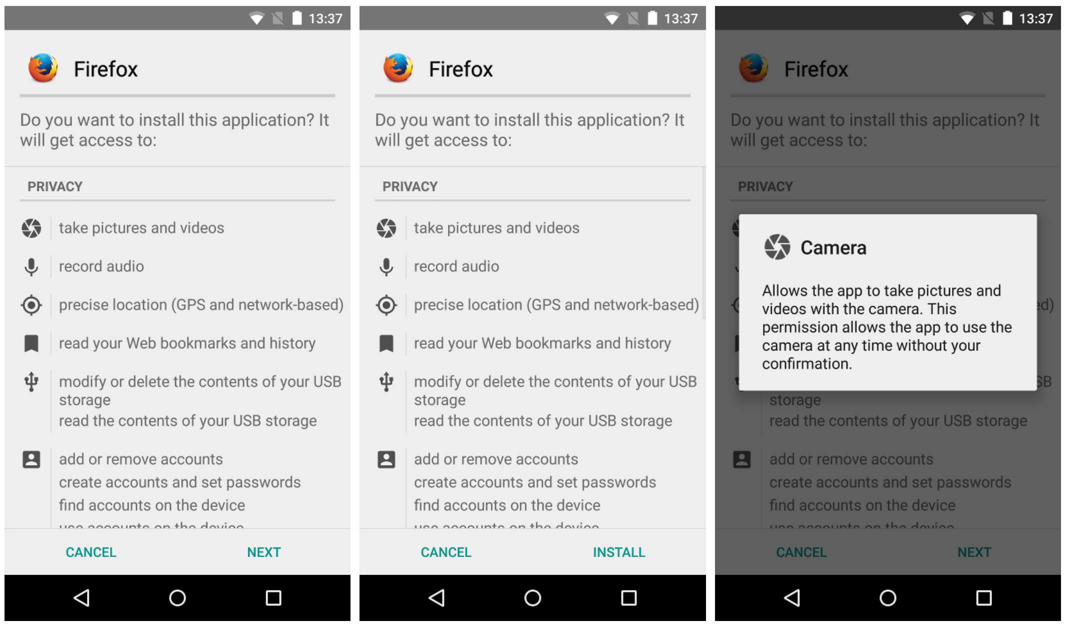
Photos/Media/Files
Uses one or more of: files on the device such as images, videos, or audio, the device's external storage

We've simplified app permissions. Learn more

Congratulations!

Spotify Music will be installed on your device soon.

FIGURE 2.2: This figure shows the workflow for installing applications and viewing application permissions remotely through the Google Play website. Screen 1 shows the Spotify application as displayed in Google Play. If a user clicks on "Install" they are shown Screen 2, which allows them to review permissions, select a device and install the application. Screen 3 notifies the user that the application will be installed soon.



13:37

Firefox

Do you want to install this application? It will get access to:

PRIVACY

take pictures and videos

record audio

precise location (GPS and network-based)

read your Web bookmarks and history

modify or delete the contents of your USB storage
read the contents of your USB storage

add or remove accounts
create accounts and set passwords
find accounts on the device

CANCEL NEXT

13:37

Firefox

Do you want to install this application? It will get access to:

PRIVACY

take pictures and videos

record audio

precise location (GPS and network-based)

read your Web bookmarks and history

modify or delete the contents of your USB storage
read the contents of your USB storage

add or remove accounts
create accounts and set passwords
find accounts on the device

CANCEL INSTALL

13:37

Firefox

Do you want to install this application? It will get access to:

PRIVACY

Camera

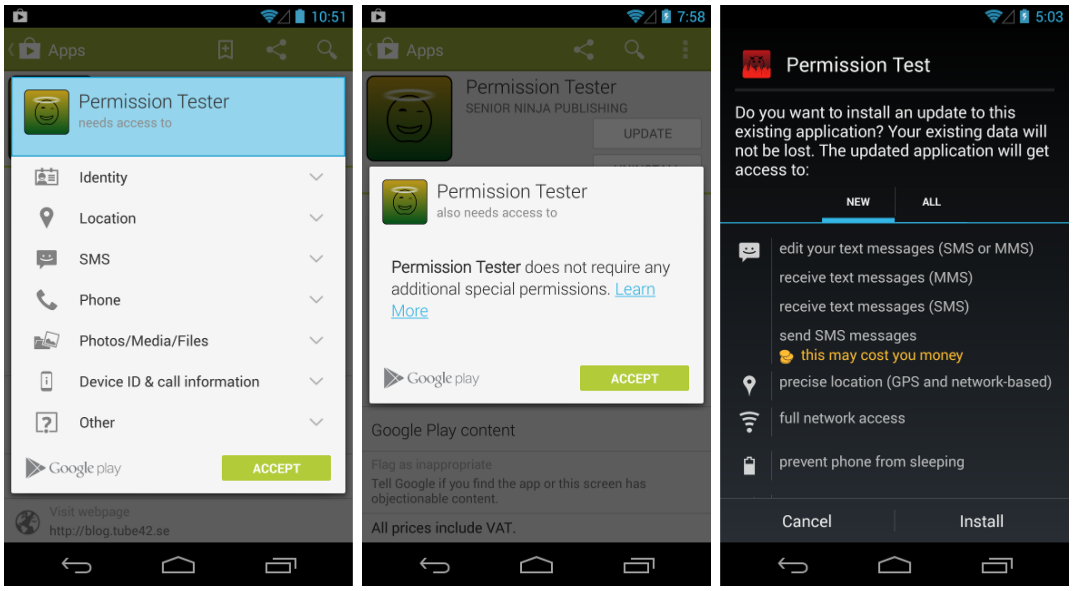
Allows the app to take pictures and videos with the camera. This permission allows the app to use the camera at any time without your confirmation.

storage
read the contents of your USB storage

add or remove accounts
create accounts and set passwords
find accounts on the device

CANCEL NEXT

FIGURE 2.3: This figure shows the workflow for viewing and accepting application permissions when installing applications from a non-Google store. Screen 1 allows the user to review permissions, tapping "Next" to scroll through all of them. Now the "Install" button appears, allowing the user to install the application. Screen 3 shows the details dialog that appears when a user clicks on a permission warning.



10:51

Apps

Permission Tester
needs access to

Identity

Location

SMS

Phone

Photos/Media/Files

Device ID & call information

Other

Google play ACCEPT

Visit webpage
http://blog.tube42.se

7:58

Apps

Permission Tester
SENIOR NINJA PUBLISHING

UPDATE

Permission Tester
also needs access to

Permission Tester does not require any additional special permissions. Learn More

Google play ACCEPT

Google Play content

Flag as inappropriate
Tell Google if you find the app or this screen has objectionable content.

All prices include VAT.

5:03

Permission Test

Do you want to install an update to this existing application? Your existing data will not be lost. The updated application will get access to:

NEW ALL

edit your text messages (SMS or MMS)
receive text messages (MMS)
receive text messages (SMS)
send SMS messages
this may cost you money

precise location (GPS and network-based)

full network access

prevent phone from sleeping

Cancel Install

FIGURE 2.4: Reddit user iamtubeman demonstrates the ability to silently add permissions to application updates. The first screen shows the initial application requesting a particular set of permissions. A second version requires additional permissions, all out of the same groups as the ones in the first version. Screen 2 shows that Google Play does not require the user to accept these new permissions, as opposed to the alternative installation process (Screen 3).

utilizing all the permissions it asked for. Secondly, some of Android's permissions are too course. For example the `READ_PHONE_STATE` permission is needed to detect when a call is being received, so that the application can mute its own sound. But the same permission also allows the application to harvest the phone's International Mobile Equipment Identity (IMEI). Thirdly, permissions are also requested because of secondary functionality that has nothing to do with the main function of the application. Mobile advertising platforms rely on using data like the user's location and the ability to access the Internet to serve advertisements. However, advertisers are usually not transparent about what they collect and how they store and process that data. Mobile advertising companies often have millions of unique records about users, which can track users from their phone to their desktop.

App Ops

App Ops is a framework inside Android that enables tweaking of individual application permissions. App Ops was originally introduced by Google in Android 4.3 as a hidden feature. With the release of Android 4.4 Google made it more difficult to access App Ops, but continued to introduce new improvements. Ultimately in Android 4.4.2, Google removed access to App Ops. However, there are still various ways to install it, although root access is required. App Ops' code is open source, so developers can integrate it with their own Android build as well.

What App Ops (Figure 2.5) does is allow a user to revoke individual permissions on a per-application basis. When such an application tries to make a system call to access something that the user has now forbidden (e.g., the camera) then Android will return an error and will not grant access to that data or functionality. In the worst case applications will crash, because they do not expect permissions te be denied.

When I started my research Fairphone was developing their own implementation of App Ops to ship with Fairphone 2. The implementation featured a reworked user interface, as displayed in Figure 2.6. Users can browse their installed applications sorted by application name or by permission. After selecting an application users can turn off individual permissions, or, if they selected a permission, they can turn off that particular permission for every applications that uses it. To bring App Ops to the user, the implementation notifies users when they run an application for the first time, as shown in Figure 2.7. Again, users can review the permissions (just like in Google Play) but now they also have the option to adjust permissions before they start the application. This requires the user to go into Advanced Mode.

Eventually Fairphone decided to scrap their implementation of App Ops, as discussed in Chapter 3.

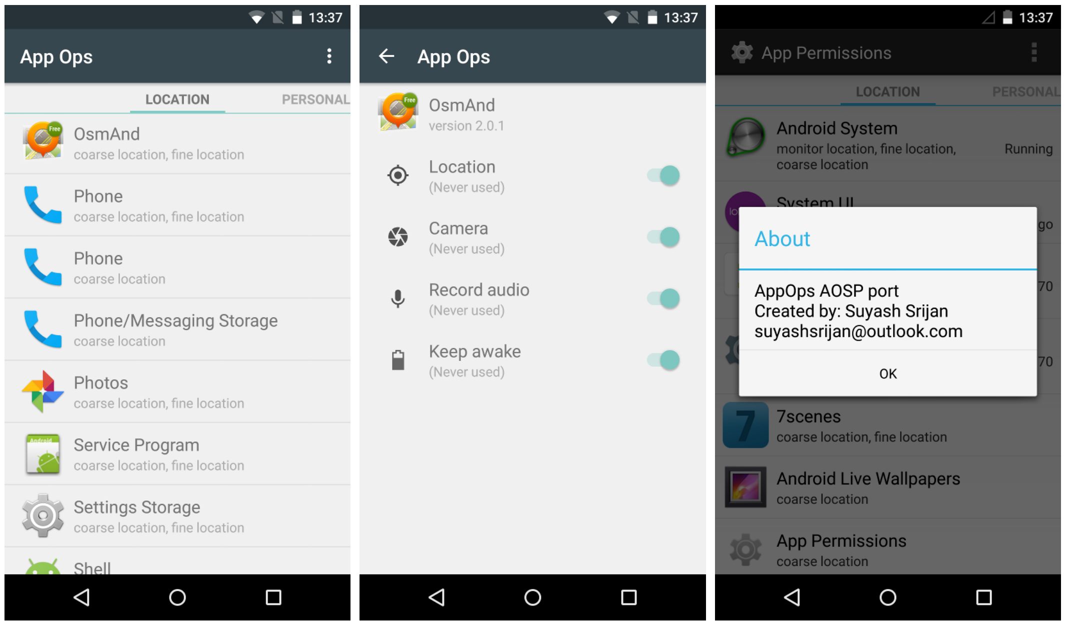


FIGURE 2.5: This figure shows the workflow for App Ops. Screen 1 shows your installed applications, indexed by the categories in which they use permissions. Screen 2 shows the OsmAnd application with its respective permission controls.

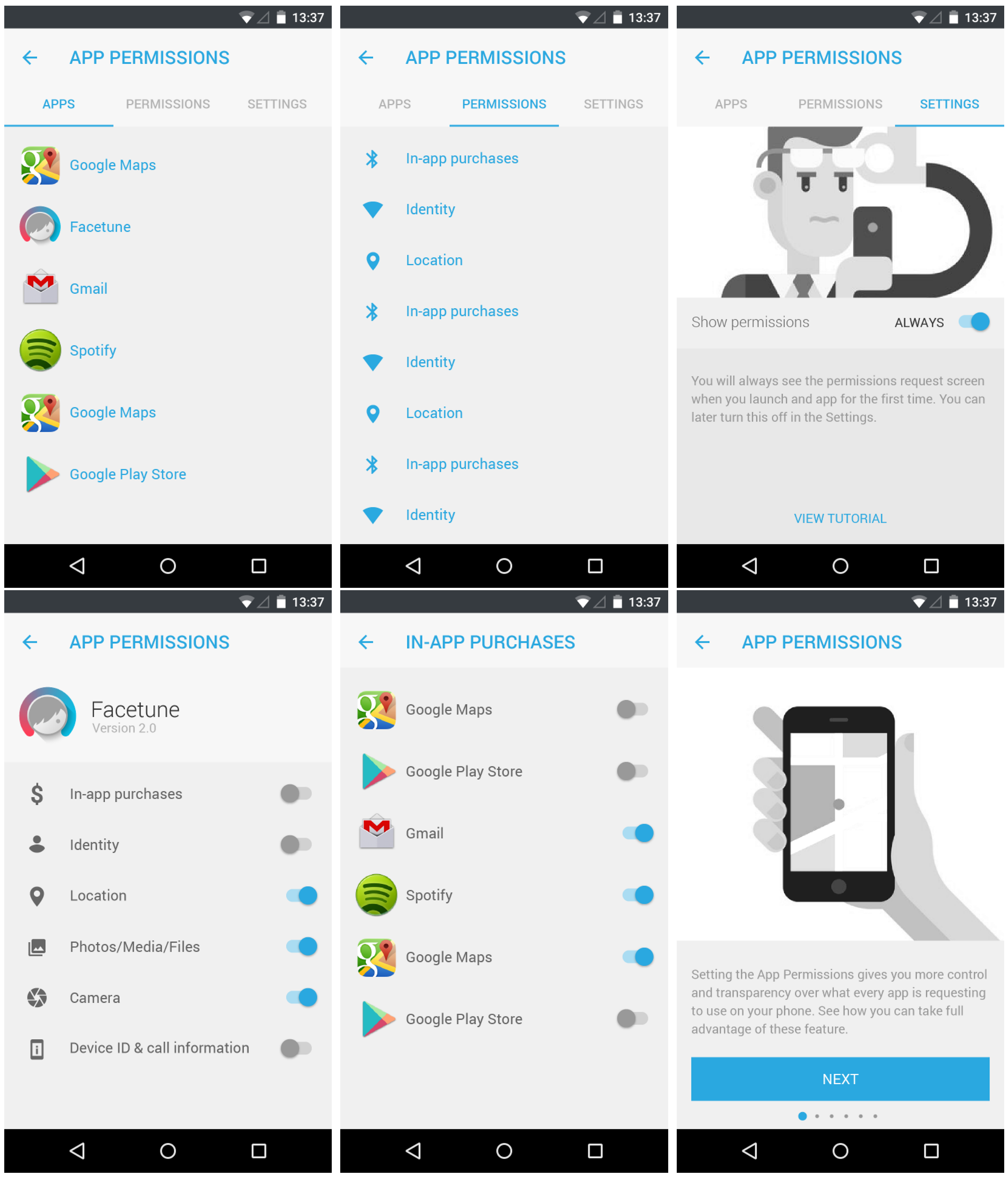
Privacy Guard

The Cyanogenmod project[2] also made their own implementation of App Ops, named Privacy Guard (Figure 2.8). This feature is accessible through the Android settings menu. Users can browse their installed applications and decide to enable the Privacy Guard per application. This means that the user will be prompted at the moment an application requires a permission. Users can also browse their applications and control permission from there. A permission can be set to Allowed, Ignored or Always ask. The screen also displays statistics on allowed versus denied count.

XPrivacy

XPrivacy (Figure 2.9) is a module for Xposed,[3] an open source framework that allows you to change the behavior of the system and applications. Xposed does not touch the original Android application packages (APKs), but instead changes are done in memory. This makes it easy to undo changes and get the original system back. In case of XPrivacy, execution of original functions is either skipped, or the result of the function is altered (for example, to return an empty contact list). The framework requires root access to run.
XPrivacy[4] was developed by the Dutch Marcel Boekhorst, because he was concerned

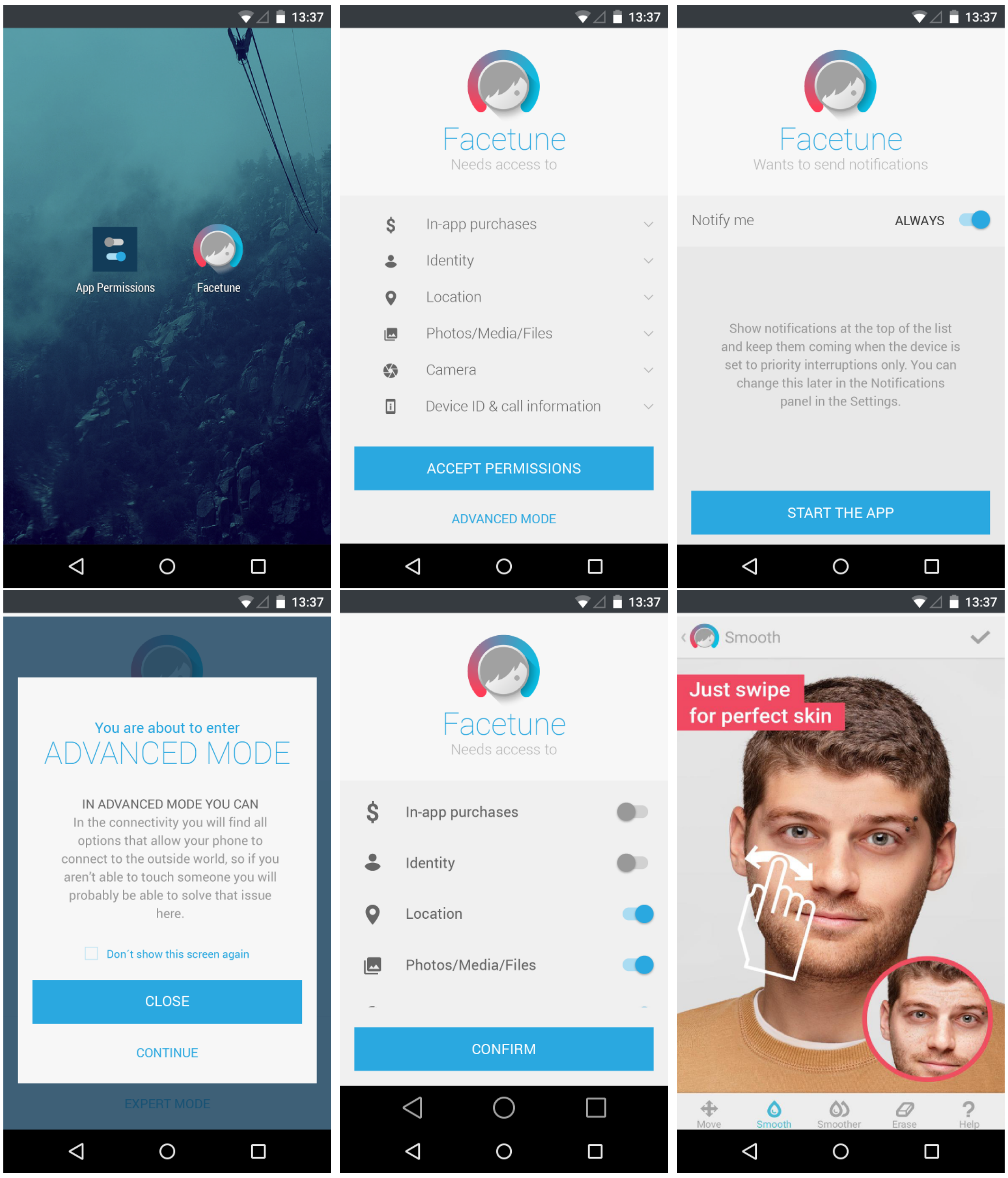
[2]http://www.cyanogenmod.org/
[3]http://repo.xposed.info/
[4]http://repo.xposed.info/module/biz.bokhorst.xprivacy



You will always see the permissions request screen when you launch and app for the first time. You can later turn this off in the Settings.

Setting the App Permissions gives you more control and transparency over what every app is requesting to use on your phone. See how you can take full advantage of these feature.

FIGURE 2.6: This figure shows the design for Fairphone's implementation of App Ops. You can browse your installed applications sorted by application (Screen 1) or by permission (Screen 2). Screen 3 shows the settings tab. After you click on an application in Screen 1, Screen 4 is displayed. Clicking on a permission in Screen 2 results in Screen 5. Screen 6 shows the tutorial.



Notify me ALWAYS ●

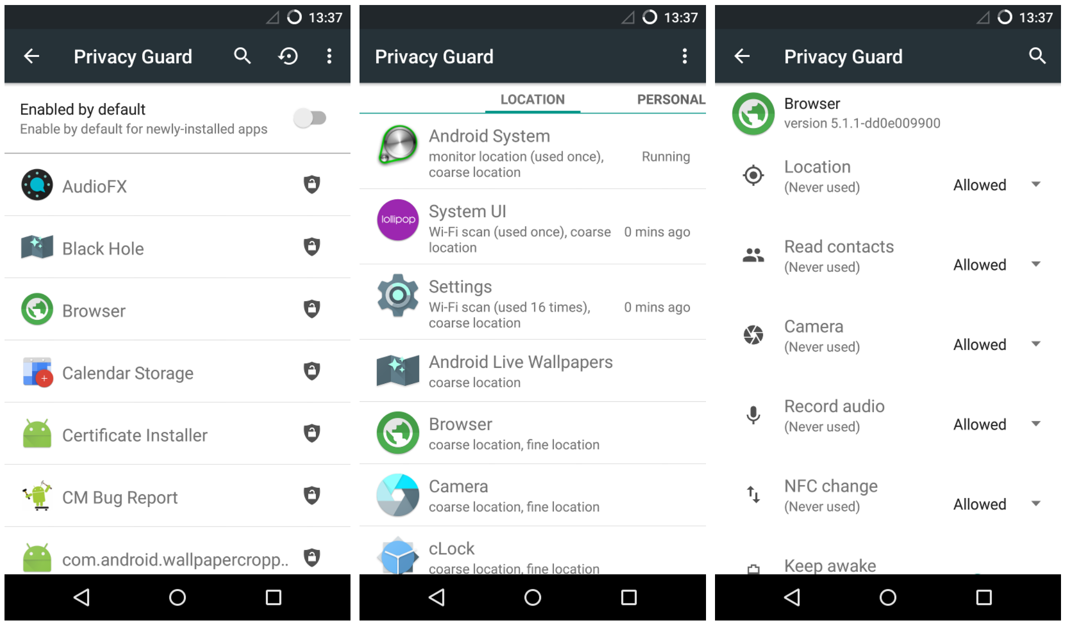
Show notifications at the top of the list and keep them coming when the device is set to priority interruptions only. You can change this later in the Notifications panel in the Settings.

IN ADVANCED MODE YOU CAN
In the connectivity you will find all options that allow your phone to connect to the outside world, so if you aren't able to touch someone you will probably be able to solve that issue here.

☐ Don't show this screen again

In-app purchases	○
Identity	○
Location	●
Photos/Media/Files	●

Figure 2.7: This figure shows a way to bring App Ops to the user. When running an application (Screen 1) for the first time, the user is shown the list of permissions the application requests (Screen 2). He can either accept these and go to the next screen (Screen 3), or he decides to adjust the permissions. After a warning (Screen 4) the user can adjust the permissions in Screen 5. After confirmation the user can start the application (Screen 6).


13:37

Privacy Guard

Enabled by default
Enable by default for newly-installed apps

AudioFX
Black Hole
Browser
Calendar Storage
Certificate Installer
CM Bug Report
com.android.wallpapercropp..

13:37

Privacy Guard

LOCATION PERSONAL

Android System
monitor location (used once), coarse location — Running

System UI
Wi-Fi scan (used once), coarse location — 0 mins ago

Settings
Wi-Fi scan (used 16 times), coarse location — 0 mins ago

Android Live Wallpapers
coarse location

Browser
coarse location, fine location

Camera
coarse location, fine location

cLock
coarse location, fine location

13:37

Privacy Guard

Browser
version 5.1.1-dd0e009900

Location
(Never used) — Allowed

Read contacts
(Never used) — Allowed

Camera
(Never used) — Allowed

Record audio
(Never used) — Allowed

NFC change
(Never used) — Allowed

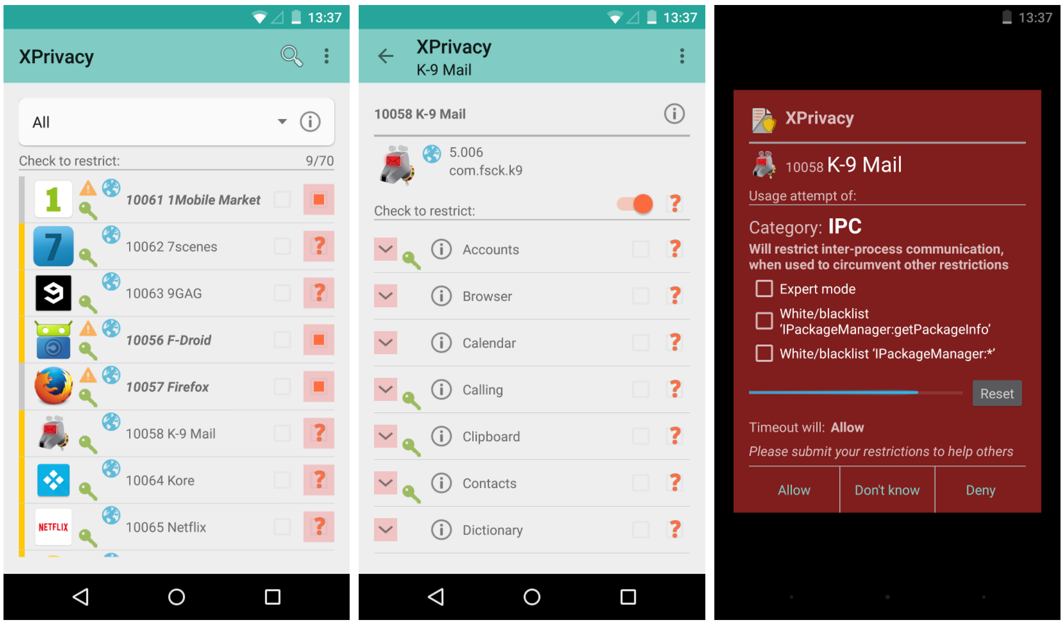
Keep awake

FIGURE 2.8: This figure shows Cyanogenmod's Privacy Guard. Screen 1 allows users to enable the Privacy Guard for individual applications. You can also modify permissions from here: select your application from Screen 2, then control its permissions in Screen 3.

about personal data being shared through applications. XPrivacy can prevent applications from leaking privacy sensitive data by restricting the categories of data an application can access. Instead of revoking or blocking permissions from an application (which could cause it to crash), XPrivacy feeds applications with fake or no data. An example is location access: restricting an application's access to your location will result in a fake location being sent to the application. There are two exceptions: access to external storage (typically an SD card) and to the Internet. These do have to be restricted by revoking permissions, because Android delegates handling these permissions to the underlying Linux file system/network. If restricting a category of data for an application causes functional limitations, access can once again be allowed to resolve the issue. Each application has on/off switches per restriction (Figure 2.9). By default, all newly installed applications cannot access any data category, which prevents an application from leaking sensitive data right after installing it. After starting the application XPrivacy prompts the user when the application requests access to a data category.

2.3 Related work

The smartphone platforms' security models delegate users to make security decisions while downloading applications from official application repositories. This delegation ranges from simply authorizing access to some protected resources, to authorizing the user to deduce whether an application may impair his security and privacy. The survey results of Mylonas et al. [9] are not in line with the expectation of smartphones' security



13:37

XPrivacy

All

Check to restrict: 9/70

10061 1Mobile Market

10062 7scenes

10063 9GAG

10056 F-Droid

10057 Firefox

10058 K-9 Mail

10064 Kore

10065 Netflix

13:37

XPrivacy
K-9 Mail

10058 K-9 Mail

5.006
com.fsck.k9

Check to restrict:

Accounts

Browser

Calendar

Calling

Clipboard

Contacts

Dictionary

13:37

XPrivacy

10058 K-9 Mail

Usage attempt of:

Category: IPC

Will restrict inter-process communication, when used to circumvent other restrictions

Expert mode

White/blacklist 'IPackageManager:getPackageInfo'

White/blacklist 'IPackageManager:*'

Reset

Timeout will: Allow

Please submit your restrictions to help others

Allow | Don't know | Deny

FIGURE 2.9: This figure shows the workflow for XPrivacy. Screen 1 shows your installed applications, including icons indicating data access. Screen 2 shows the K-9 Mail application with its respective permission controls, and Screen 3 shows the prompt that is displayed when an application requests a permission.

models for a reasonable security aware user. In contrast, they suggest that users are not adequately prepared to make appropriate security decisions. A second research by Mylonas et al. [10] confirms that smartphone users require awareness training specifically tailored for smartphone security. Kelley et al. [11] studied the Android security model in particular, and state that users do not understand Android permissions. Their conclusion is that users are currently not well prepared to make informed privacy and security decisions around installing applications from Google Play. Felt et al. [12] discovered low rates of user attention and comprehension, so significant work is needed to make the Android permission system widely accessible.

Despite researchers' predictions [15, 28], permissions are not wholly ineffective. A minority of users demonstrated awareness and understanding of permissions, and permissions helped some users avoid privacy-invasive applications [12]. This motivates continued effort towards the goal of usable permissions. Suggestions for improvement include connecting reviews to permissions, customizing warnings to users' concerns and investigating new types of warning dialogues.

Benton et al. [13] improved the permissions prompt by introducing a short notice, shown after the standard permissions prompt, which explained the combination of permissions. Initially it did not have a statistically significant effect on the installation count or the post-installation awareness/regret. However, introducing an additional aggregating, non-technical visual warning for risky permissions in a second experiment proved to be much more effective than the text-warning alone. Sarma et al. [29] also proposed the notion of using effective signals to improve Android security. They introduced risk signals combining information about the permissions requested by an application, the function category of an application, as well as what permissions other applications request. The effectiveness of their approach was demonstrated by using two datasets. Yang et al. [30] also tried to enhance user comprehension of Android permissions, by using crowdsourcing. They designed a tool which provides information of permission usage to users using two techniques: record/replay and permission suppression. Their experiments showed that even with a simple visual detection of the differences in output, their approach can help users better understand about 40% of application permissions.

Overall I conclude that in their current state, Android permissions appear to be limited as a tool for only privacy conscious users, although a security background has limited impact on awareness [10]. In light of these results, the Google Play interface may need to be modified to stress permissions or make them easier to understand for users.

Research has shown that valuable conclusions can be drawn from analyzing the permissions applications request. Sato et al. [14] uses the entire manifest file (instead of just the permissions alone) to detect malware. Moonsamy et al. [18] studied the detection of malware as well: they considered not only required permissions, but also the permissions that are actually used. Mylonas et al. proposed a method for Privacy Impact Assessment (PIA) for Android applications based on permissions, whose results I will use in Chapter 3.

Chapter 3

Privacy Impact

The first project I started with addresses the issue of users not being aware of Android permissions or having difficulty comprehending them. The solution discussed in this chapter translates the permissions requested by an application into an easy to comprehend graphic element, with the emphasis on privacy: the Privacy Impact. This chapter details its development process, from user stories and design to the final implementation.

3.1 User stories

The goal of the Privacy Impact is to help users decide if they want to use particular applications. It can serve as an additional property to take into account when selecting applications, together with other properties such as amount of downloads, reviews and word of mouth.

In an application store

The best place to put the privacy impact is in an application store. This way the user can take the privacy impact into account when searching for applications. For example, the user is in need of a flashlight application, so he enters the search query ‘flashlight’. Flashlight applications should require access to nothing but the camera flash, so its privacy impact should be zero. However, lots of flashlight applications request additional permissions in order to retrieve personal data, which is being sold to other parties to make revenue. If the user could sort his search results based on the privacy impact, it is easy to select the one that has the least impact on their privacy.

In a separate application

Another suggestion is to create a privacy panel, as part of the Settings menu or as a separate application. In the context of Fairphone this would be their implementation of App Ops, since it already contains a privacy-related feature. Based on the privacy impact the user can decide whether he wants to turn off particular permissions for an application. This way the Privacy Impact would function as a trigger to adjust application permissions.

When first starting an application

Fairphone's App Ops design discussed in Chapter 2 could be modified to include the Privacy Impact. The first screen in Figure 2.7 shows the list of permissions requested by an application, but the user has just reviewed and accepted those on Google Play. The list could be replaced with the Privacy Impact, with as goal to trigger the user to go into advanced mode and adjust the permissions.

3.2 Design

The Privacy Impact design was split into two parts: the technical design and the user interface design.

3.2.1 Technical design

Android's package manager allows you to retrieve the list of permissions an application requests. To calculate the privacy impact from these permissions, an algorithm is needed. This algorithm could range from a trivial calculation to a complex algorithm that also takes other factors into account. The following algorithms all calculate an number between 0 and 100, but this can be scaled to whatever is needed. Here I assume that the higher the number, the more impact the application has on privacy. The variables used can be retrieved from the operating system. For descriptions of the individual permissions, see Appendix A.

Algorithm 1: all permissions

This is just a trivial count of permissions. It assumes that all permissions are equally harmful to the user's privacy, and that less permissions used is better.

```
privacyImpact =
        (amountOfRequestedPermissions / amountOfDevicePermissions) * 100
```

The problem with this algorithm is that in practice, not all permissions are potentially harmful: some allow access to personal data (e.g. `READ_CONTACTS`), while others do not (`SET_WALLPAPER`).

Algorithm 2: harmful permissions

This algorithm only looks at permissions that are known to be harmful to the user's privacy. It uses the top 40 most dangerous permissions as found by Wang et al. [17]. I assume that all those 40 permissions are equally harmful, so they carry the same weight.

```
privacyImpact = 0
for all requestedPermissions
    if requestedPermission is in harmfulPermissions
        privacyImpact += 2.5
```

The problem with this algorithm is that permissions by themselves are not that harmful. An application might have access to your contacts, but without a communication channel (like the Internet) it will not be able to send them to a server. Usually it is combinations of permissions that lead to privacy infringements.

Algorithm 3: dangerous combinations

The third algorithm looks at combinations of permissions known to be harmful. The following rules come from work by Enck et al. [15], who developed a five step process for designing security rules.

```
An application must not have...
(1) the SET_DEBUG_APP permission label.
(2) PHONE_STATE, RECORD_AUDIO, and INTERNET permission labels.
(3) PROCESS_OUTGOING_CALL, RECORD_AUDIO, and INTERNET permission labels.
(4) ACCESS_FINE_LOCATION, INTERNET, and RECEIVE_BOOT_COMPLETE permission
       labels.
(5) ACCESS_COARSE_LOCATION, INTERNET, and RECEIVE_BOOT_COMPLETE
       permission labels.
(6) RECEIVE_SMS and WRITE_SMS permission labels.
(7) SEND_SMS and WRITE_SMS permission labels.
(8) INSTALL_SHORTCUT and UNINSTALL_SHORTCUT permission labels.
(9) the SET_PREFERRED_APPLICATION permission label and receive Intents
       for the CALL action string
```

```
privacyImpact = 0
for all dangerousPermissionCombinations
    if dangerousPermissionCombination is in requestedPermissions
        privacyImpact += 10
```

The calculation is more sophisticated now: instead of a trivial permission count, I now look at combinations of permissions that could be potentially harmful. Quick tests with a demo application have shown that we are able to say something useful: applications like Facebook and Whatsapp that can exfiltrate data score relatively high compared to simple or system applications that have limited access to sensitive data and communication channels. The problem with this algorithm is that it is focused on malware instead of privacy. Another problem of this algorithm is that it hardly ever got below three stars.

Algorithm 4: dangerous combinations, privacy

This algorithm looks at combinations of permissions that relate to privacy risks. Mylonas et al. [16] made a mapping of data assets, permissions and threats. They analyzed every possible permission combination from a big application set, resulting in pairs of permissions that protect an asset and a transmission channel. Their sample contains 89 such pairs, of which I implemented the twenty most frequent ones:

Channel	Access to data
INTERNET	ACCESS_NETWORK_STATE
INTERNET	WRITE_EXTERNAL_STORAGE
INTERNET	READ_EXTERNAL_STORAGE
INTERNET	READ_PHONE_STATE
INTERNET	ACCESS_WIFI_STATE
INTERNET	ACCESS_COARSE_LOCATION
INTERNET	ACCESS_FINE_LOCATION
INTERNET	GET_ACCOUNTS
INTERNET	CAMERA
INTERNET	GET_TASKS
INTERNET	READ_CONTACTS
INTERNET	READ_HISTORY_BOOKMARKS
INTERNET	READ_CALL_LOG
INTERNET	RECORD_AUDIO
INTERNET	READ_LOGS
INTERNET	USE_CREDENTIALS
SEND_SMS	READ_PHONE_STATE
INTERNET	RECEIVE_SMS
SEND_SMS	ACCESS_NETWORK_STATE
SEND_SMS	WRITE_EXTERNAL_STORAGE

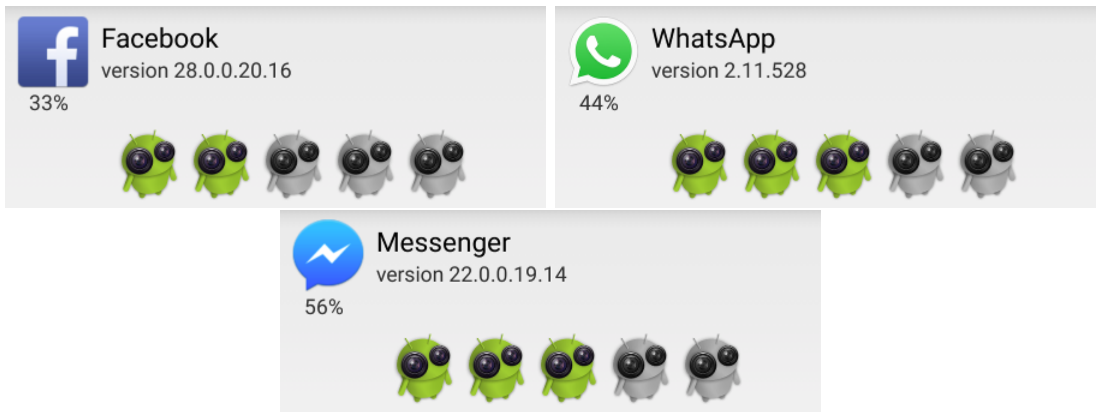


FIGURE 3.1: This figure shows an early prototype of the Privacy Impact. The impact is resembled by spying Androids: the more spying Androids, the less privacy friendly the application is. This prototype was used to demonstrate if the algorithm actually had the desired effect.

```
privacyImpact = 0
for all dangerousPermissionCombinations
    if dangerousPermissionCombination is in requestedPermissions
        privacyImpact += 5
```

Again tests have shown that applications like Facebook and Whatsapp score relatively low compared to simple or system applications.

3.2.2 User interface

This section details the user interface design process, from the early prototypes to the final design.

Prototypes

The first prototype I developed used a five point scale to represent the Privacy Impact. I used spying Androids as images: the more spying Androids, the more privacy invasive an application is (Figure 3.1). The algorithm used here is Algorithm 3.

I pitched the prototype to the Fairphone team, where it received positive feedback. The only remark was that in its current state, it sends out a negative message. This does not fit well with the overall Fairphone messaging strategy, so we decided to change "Privacy Impact" to "Privacy Score": the more privacy friendly an application is, the better it scores. I also decided to switch the algorithm to Algorithm 4, because it focuses more on privacy. This resulted in a second prototype. displayed in Figure 3.2. I used stars as images to resemble a positive message. At this point the decision was made to include the Privacy Impact into Fairphone OS for Fairphone 2, in the form of the third user story.

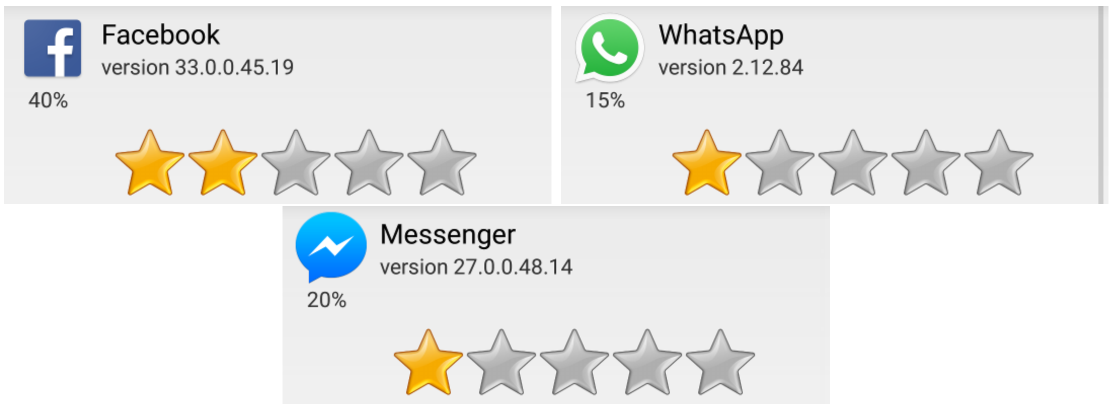


FIGURE 3.2: This figure shows the second prototype of the privacy impact. The score is resembled by stars: the more stars, the more privacy friendly the application is. It also features a different algorithm.

Graphic design

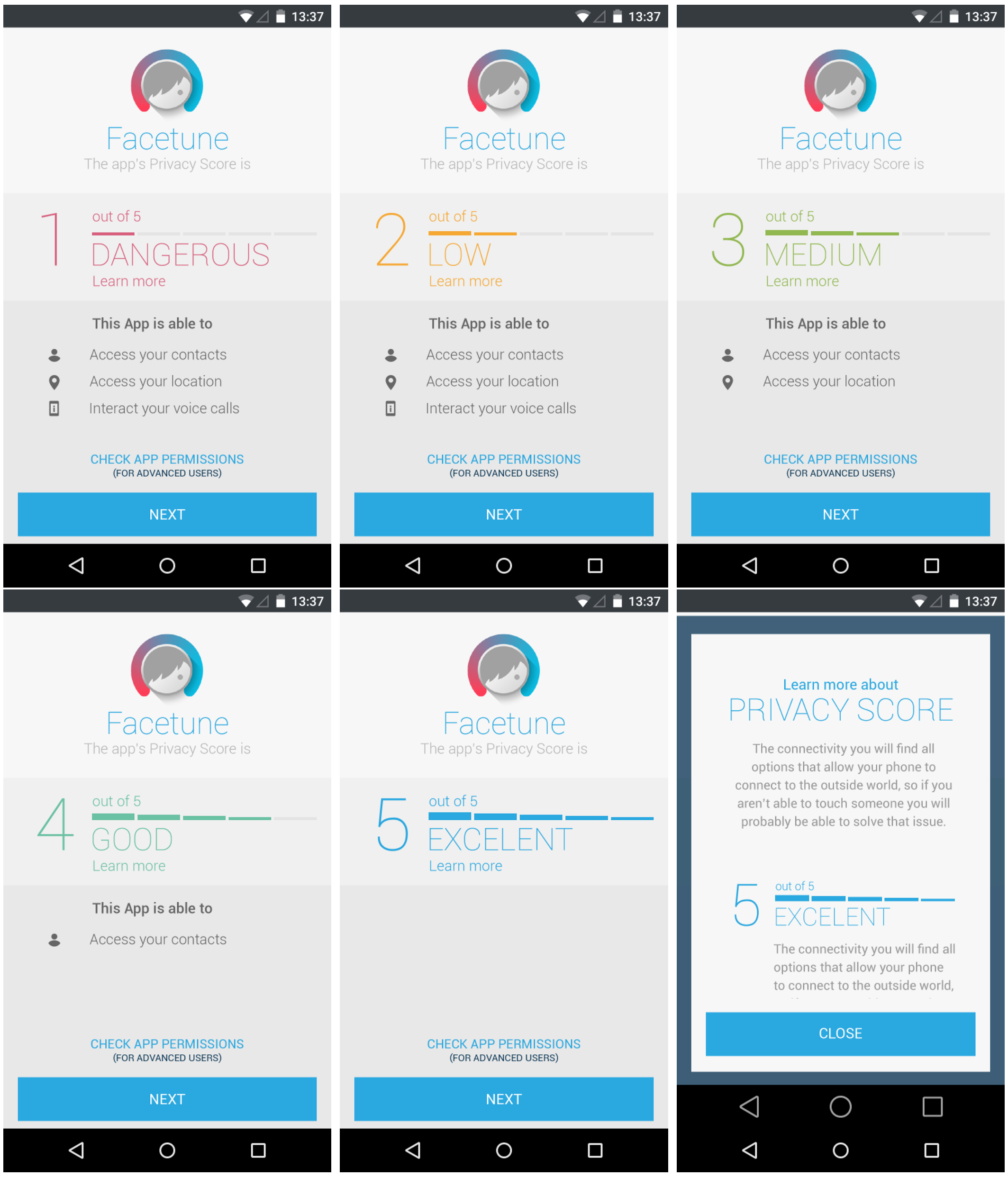
To come up with a proper design I collaborated with Kwamecorp, who does part of the software development for Fairphone. They made a design (Figure 3.3) that was imported into Marvel, a prototyping tool for designers. This resulted in an interactive mockup that could be used for usability testing.

First usability test

Three participants with basic to intermediate smartphone knowledge tried the App Ops feature excessively. They used the new features it has to offer in every way possible and were given the task to speak their mind while navigating through the feature. Testing was done on Nexus 5 devices with interactive App Ops mockups. All participants first got a brief introduction to application permissions in general and were asked how they felt towards this system integrated in Android, and what the effects of these permissions are.

Participant 1: *Female, age 27, basic/intermediate Android smartphone knowledge*

Participant relies mostly on suggestions from her friends and family in determining if an application is trustworthy or not. The popularity of an application also affects her decisions. She stated that she only reads the Google Play permission dialog when installing unknown applications. It was not immediately clear to the participant that the startup screen (when starting a newly installed application) came from Fairphone, it could also have come from the application developer itself. The participant thought a Low score was a nice call to action and made the participant think about the situation. A score of one (Dangerous) caused some confusion and should be changed into something else. The startup screen is considered a nice feature that adds a level of security. The participant had a strong positive feeling towards Fairphone and trusts Fairphone without


13:37

Facetune
The app's Privacy Score is
1 out of 5
DANGEROUS
Learn more
This App is able to
Access your contacts
Access your location
Interact your voice calls
CHECK APP PERMISSIONS
(FOR ADVANCED USERS)
NEXT

13:37

Facetune
The app's Privacy Score is
2 out of 5
LOW
Learn more
This App is able to
Access your contacts
Access your location
Interact your voice calls
CHECK APP PERMISSIONS
(FOR ADVANCED USERS)
NEXT

13:37

Facetune
The app's Privacy Score is
3 out of 5
MEDIUM
Learn more
This App is able to
Access your contacts
Access your location
CHECK APP PERMISSIONS
(FOR ADVANCED USERS)
NEXT

13:37

Facetune
The app's Privacy Score is
4 out of 5
GOOD
Learn more
This App is able to
Access your contacts
CHECK APP PERMISSIONS
(FOR ADVANCED USERS)
NEXT

13:37

Facetune
The app's Privacy Score is
5 out of 5
EXCELENT
Learn more
CHECK APP PERMISSIONS
(FOR ADVANCED USERS)
NEXT

13:37

Learn more about
PRIVACY SCORE
The connectivity you will find all options that allow your phone to connect to the outside world, so if you aren't able to touch someone you will probably be able to solve that issue.
5 out of 5
EXCELENT
The connectivity you will find all options that allow your phone to connect to the outside world,
CLOSE

FIGURE 3.3: This figure shows the first design for the privacy score. The first five screens show the different levels of the privacy score. The final screen shows a scrollable dialog with explanation for all the levels.

question.
The notification toggle is seen as a great feature that gives a better feeling of control over the particular application.
The App Ops application itself is seen as a useful feature, but required some explanation from the facilitators' side. Once this was done the purpose was much clearer. Also showing Apps and Permissions next to each other caused some confusion, but once the participant made a few clicks it all made sense. All in all, the integration of App Ops created awareness on application privacy for the participant.

List of our most important findings:

- The score is the first thing the participant noticed;
- Participant thought the startup screen was Fairphone's, but was not really sure;
- The term 'check' in 'check app permissions' should be changed into a term suggesting control, i.e. 'adjust';
- Participant prefers seeing the score over the permissions themselves;
- The participant was not bothered/annoyed by this screen;
- The option 'Advanced Users' felt scary, needs to feel more accessible;
- Participant liked the option to turn off notifications;
- Screen is shown again when an application has new permissions in an update;
- Change 'Dangerous' to 'Very low' or something similar;
- Use green as color for the highest score.

Participant 2: *Female, age: 65, basic Android smartphone knowledge*

Participant does not download a great deal of applications, but if she does it is mostly due to the fact friends recommend it. She never reads the permission dialog in Google Play, simply because she does not understand what it says. The App Ops startup screen does not give a proper call to action right away, and because of the text 'for advanced users' she immediately wants to press 'Next' without changing the permissions, even though the score says Low. The participant simply stated that she is not an advanced user and therefore wants to ignore it altogether.
About the score: 'learn more' is not attractive enough. The participant suggested to name it 'what does this mean?'
The participant was under the impression that the application developer showed the screen and not Fairphone, so clearer Fairphone branding is necessary. The participant had a positive feeling towards the whole App Ops feature in general and was not bothered by the startup screen, which created a sense of trust.

List of the most important findings:

- The score is the first thing the participant noticed;
- Has the urge to press 'learn more', but...;
- Thinks 'learn more' should be changed to 'what does this mean?';
- Use a term suggesting control instead of 'check' in 'check app permissions';
- The participant has the urge to press 'Next' after seeing the screen, the term 'advanced user' scares her off;

- Participant thinks the startup screen is from the application itself;
- Participant gets a safe feeling from App Ops.

Participant 3: *Male, age: 27, basic iOS smartphone knowledge*

Participant is an iOS user, who has no experience with Android. He thinks the permission system in Android is poorly implemented, and he prefers the way iOS handles permissions, although in general he thinks permissions are rather cumbersome. He is aware that data is being sold to third parties. Explaining the participant that with App Ops the user is in control of what data applications can access, made the participant enthusiastic.
When confronted with a Low score on application startup, the first question was: ''what does this mean?". Renaming 'learn more' to 'what does this mean' seems like a good solution. Fairphone branding was not apparent for the participant and he was confused by this, so more Fairphone branding is necessary.
The calculation of the score raised questions: the participant wondered if it is good idea to let Fairphone calculate such a score. He thinks the score might not be objective enough. A third party would be more ideal to make it more objective.
'app's privacy score' written in gray was not very visible to the participant and needs to be shown more clearly. The term 'for advanced users' was not attractive and needed to be more approachable.
The App Ops application itself was clear and self explanatory, although it took some time. Combing the Apps and Permissions tabs to create clarity would be a solution. At least make them appear in a way they seem related to each other.

List of the most important findings:

- He would like to be told if an application sells his data to other parties;
- The score is the first thing he sees, but he immediately wonders how it was calculated;
- Thinks the startup screen is by Fairphone;
- Would like an explanation per permission in the screen where you can control them;
- 'Privacy score' in gray is not visible enough, so the participant is not drawn to it;
- Making the score interactive would be interesting: turning off permissions results in a higher score;
- The scale is good (1 to 5);
- Use WhatsApp or Facebook as an example instead of Facetune;
- Change design of App Ops application: the tabs are a bit unclear.

Redesign

Before the feedback from the usability tests could be applied, other decisions changed the course of the project. First of all, Fairphone decided that App Ops would be a too powerful feature to integrate. Using it without proper knowledge could result in applications not functioning properly, or even not functioning at all, which would result in lots of customer service calls. Fairphone opted for a more low profile integration of App Ops by making the feature only accessible through a hidden Settings menu, but in the end they decided to completely remove the feature. This also meant that the link to App Ops in the Privacy Impact screen would be removed.
Another reason for removing App Ops was the fact that Google announced that in Android M users will have the option to control permissions by default. The introduction of Android M and its revised permission system is discussed in Chapter 6.
The decision was also made to change Privacy Score to Privacy Impact. This also implies a negative instead of a positive score: the higher the impact, the more privacy invasive an application is. The amount of levels was also reduced. A new interactive mockup was made by Kwamecorp, as shown in Figure 3.4.

Second usability test

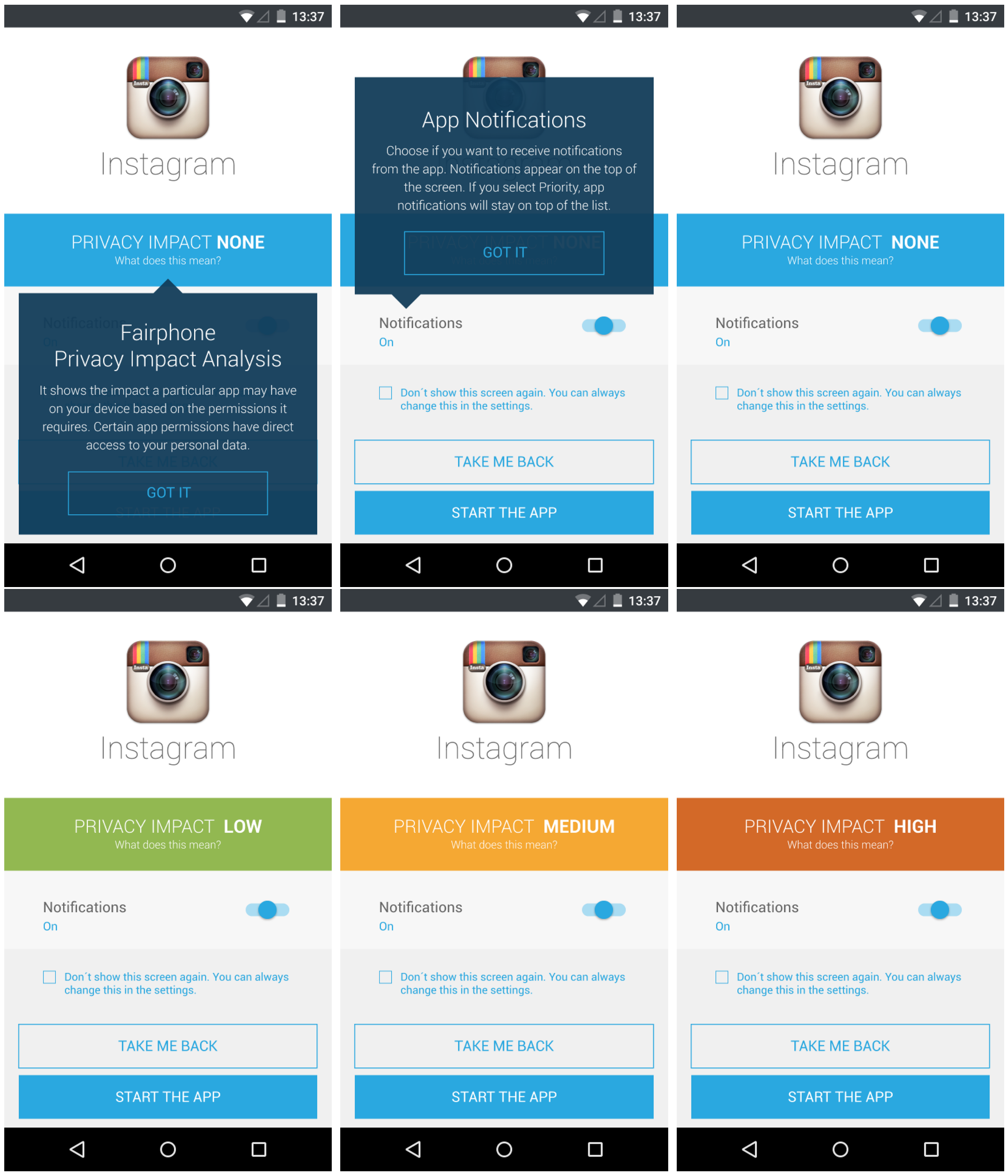
We conducted a second usability test with the new mockup, featuring three new participants. Since the design was considered close to final, we also conducted two usability surveys: the System Usability Scale (SUS) and the Microsoft Reaction Card (MRC) method.

Participant 1: *Male, age: 37, Basic iOS knowledge*

Participant 2: *Female, age: 25, Basic / Intermediate Android knowledge*

Participant 3: *Female, age: 30, Basic / Intermediate iOS knowledge*

- Only the third participant states that she would pay/pays attention to the Google Play permission dialog;
- Privacy Impact levels are fine now (None, Low, Medium and High), though the colors being used for these levels are causing some confusion - i.e. Low and Medium are both greenish. We suggest making the build up steps more logical color wise, i.e. green, yellow/light orange, dark orange;
- The Notification prompt points to the lower part of the screen, but not directly towards the Notification setting only. Users don't know where to look and feel a bit lost.
  - Suggestion 1: Move the prompt's arrow to the left or right side so it is directly above the 'Notifications' text and/or above the slider;



Fairphone Privacy Impact Analysis

It shows the impact a particular app may have on your device based on the permissions it requires. Certain app permissions have direct access to your personal data.

App Notifications

Choose if you want to receive notifications from the app. Notifications appear on the top of the screen. If you select Priority, app notifications will stay on top of the list.

Don´t show this screen again. You can always change this in the settings.

FIGURE 3.4: This figure shows the final design for the privacy impact, including the feedback from the second usability test. The first two screens show the tutorial dialogs. The other screens show the different Privacy Impact levels.

- Suggestion 2: Make the background of the Notification area gray, so it is more clear where the prompt's arrow is pointing towards;
- Suggestion 3: Make both prompts the same design as the first shown prompt, instead of flipping them around like it is now. So both prompts point upwards;
- Suggestion 4: Make the prompt text more clear for the user, or even add a link to an Android Notifications help article;
- Suggestion 5: A combination of the above.

- The newly introduced Notification toasts are a bit redundant. They basically repeat the information already displayed (ON / ON) and toasts are shortly displayed, making it difficult to read a long tutorial text. Toasts are not designed for that in general. Toasts can be used only if they remain short and add value;
- The two elements in the main screen - Privacy Impact and Notifications - are confusing for the user, because they think they are related. By creating more space between the elements or putting something in between can eliminate the issue;
- The opt out text is too long, should be explained in one clear line of text.
- Overall all participants were - just like the previous usability test rounds - content with this new feature and felt it fits Fairphone's values and creates a certain protection level. We clearly need to focus in minor things (i.e. texts, positioning, etc.) to make it perfect.

Developed by Microsoft in 2002 by Joey Benedek and Trish Miner, the MRC method is used to check the emotional response and desirability of a design or product. This method is commonly used in the field of software design. The participant is asked to describe a design or product using any number of a list of 118 words (see Appendix B). After viewing a design or product the participant is asked to pick out the words they feel are relevant. The moderator would then ask the participant to describe their rationale for their selection. A wordcloud containing the most picked words during this usability test can be found in Figure 3.5.

SUS [31] was applied to determine the overall usability of the feature. The SUS test covers effectiveness, efficiency and satisfaction of the system participants tested. It consists of a 10 item questionnaire (included in Appendix C) with five response options for respondents; from Strongly agree to Strongly disagree. The final score for Privacy Impact is 73.3, which means a B-grade. You would need to score above an 80.3 to get an A. This is also the point where users are more likely to be recommending the product to a friend. Based on research (more than 500 studies), a SUS score above a 68 would be considered above average and anything below 68 is below average. The graph in Figure 3.6 shows how the percentile ranks associate with SUS scores and letter grades.

3.3 Implementation

At the time of writing, the Privacy Impact feature is scheduled for integration with Fairphone OS for Fairphone 2. The OS's new features (including the Privacy Impact) are currently being beta-tested in the office and among beta-testers from the community.

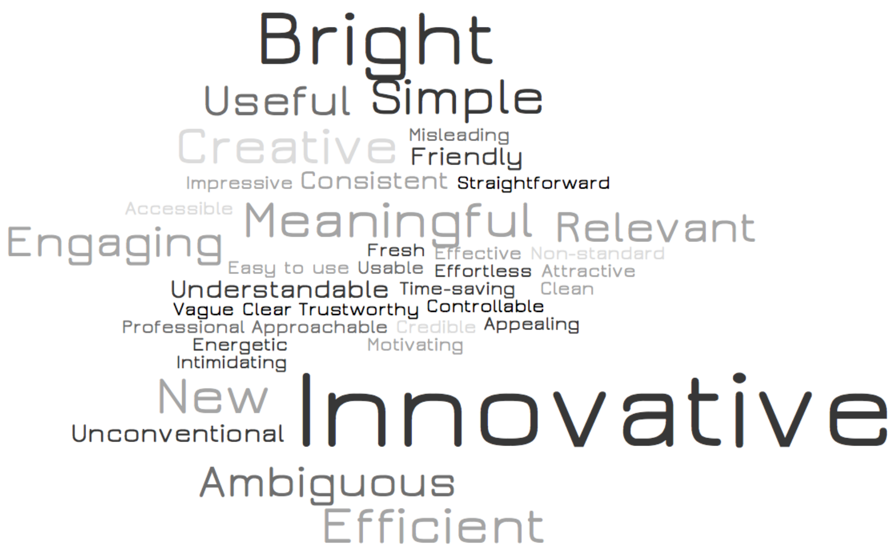


FIGURE 3.5: MRC wordcloud: the larger the font size and the greater the contrast, the more frequently participants selected the adjective.

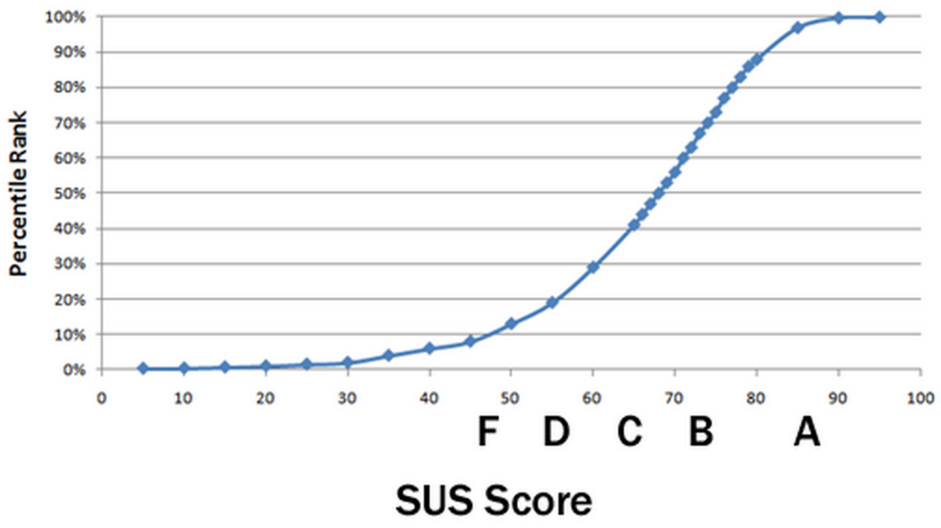


FIGURE 3.6: This graph shows how the percentile ranks associate with SUS scores and letter grades. The Privacy Impact feature scored 73.3 points.

Chapter 4

Permission Usage

The second project addresses the issue of the Android permission system not being transparent to the user. Permissions requested by applications are only displayed at install-time. After the user accepts them, the application has unlimited access to those permissions, while the user has no notion of how the application is actually them. I introduce a method to record permission usage by applications and present it to the user by means of an Android application. This gives users insight into how applications are actually using the permissions they request, for example when setting up a newly installed application or when the phone is on standby.

4.1 User stories

The goal of the Permission Usage project is to give users insight into how Android applications are actually using the permissions they request at install-time. In this section I introduce various usage scenarios for such a feature, which might lead the user to doubt the application developer, research alternatives or adjust permissions with a feature like App Ops.

Permission not used

A user might discover that an application is not using a particular permission it requested at all (so the application is *overpriviliged*). Research by Felt et al. [8] showed that about one third of the applications in their test set included unnecessary permissions, which may raise security problems. One potential cause of overpriviliged applications is unclear or incorrect documentation, where the Android API documentation does not mention that a permission is required or the wrong permission is documented. An example of such a permission is location access: applications often request both the `ACCES_COURSE_LOCATION` and `ACCES_FINE_LOCATION`, but `ACCES_FINE_LOCATION` already allows applications to fall back to course location services.

Permission only used once

A user might discover that an application uses a particular requested permission only once. One example of such a permission is the `READ_SMS` permission in WhatsApp: it uses the permission only once (to read a confirmation SMS message when setting up the application).

Permission used excessively

A user might discover that an application uses a particular permission excessively. For example, an application that has access to your contacts keeps reading them out every day at noon. From actually using the application the user might know that this behavior is unnecessary, so it might indicate suspicious activity.

Permission granted/denied

Instead of studying only permissions that were granted by the system, it would also be valuable to study permissions that were denied. This could be caused by applications requesting a permission they did not specify (so the application is *underpriviliged*) or by the user revoking permissions through a feature like App Ops or XPrivacy. Suppose a user revoked too many permissions for an application, so it no longer functions. The user can now see what particular permission was denied by the system and turn it on again.

Background/standby

A user might discover that applications are using permissions while he is not actively using the application, or even when the phone is on standby.

Particular permissions

Some permissions are particularly interesting when it comes to handling privacy-sensitive data, so they could be studied in detail.

`INTERNET` - In these days almost every application seems to need Internet access for something, but it is usually not clear what this access is needed for. Application functionality often requires Internet access, but connections are also opened for tracking the

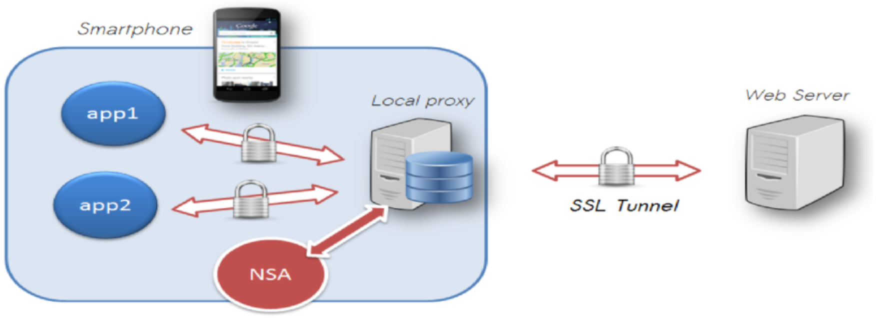


Figure 4.1: NSA application architecture: HTTP traffic is intercepted by a local proxy service

user and downloading advertisements. A methodology for generating descriptions of application network behavior could shed a light on this situation. Vigneri et al. [32] describe a lightweight characterization methodology that makes use of a local proxy (Figure 4.1) that intercepts HTTP traffic. With this application, users are able to understand the different domains an application is communicating with, which enables them to make informed decisions about the desirability of the applications they install.

`ACCES_COURSE_LOCATION` and `ACCES_FINE_LOCATION` - These two permissions relate to location access: `ACCES_COURSE_LOCATION` approximates the phone's location derived from network location sources (such as cell towers and Wi-Fi), whereas `ACCES_FINE_LOCATION` provides precise location from sources such as GPS and Wi-Fi. It would be interesting to show the user where and when he was being tracked by an application, for example by showing a map incorporating this data.

`ACCESS_WIFI_STATE` - This permission was studied by Achara et al. [33]. It falls in the group of 'Network communications', and grants access to the data associated with the Wi-Fi interface. It does so by making some of the methods of the `WifiManager` class available to the application. These methods and their capabilities are presented in Table 4.1, which shows that the permission provides access to uniquely identifying data (such as MAC address). Combined with the `INTERNET` permission this could reveal to the user how they are being tracked by tracking companies.

Permission patterns

Just like the Privacy Impact project, it would be interesting to look at combinations of permissions. But now we can see how the combinations are actually being used. So for example, the user might discover that an application first reads the contacts, then writes to external storage, and then uses the Internet. This might indicate that the application is trying to steal your contact data. Wang et al. [17] discovered permission patterns that can detect malicious behavior with a high level of accurary.

Method name	**Description**	**Retrievable Information**
isWifiEnabled()	Returns whether Wi-Fi is en-abled or disabled.	Returns true if Wi-Fi is enabled.
getWifiState()	Gets the Wi-Fi enabled state.	Enabled, disabled, currently being enabled or disabled, unknown
getConfiguredNetworks()	Returns a list of all configured networks.	For each configured network/AP: SSID, allowed protocols and security schemes
getConnectionInfo()	Returns dynamic information about the current Wi-Fi connection, if any is active.	**About AP:** BSSID, SSID, RSSI **About Device:** Wi-Fi MAC address, IP address
getScanResults()	Returns the results of the last AP scan.	**For each AP:** BSSID, SSID, signal strength, channel, capabilities
getDhcpInfo()	Returns the DHCP-assigned addresses from the last successful DHCP request, if any.	IP address, DNS server address, gateway and netmask

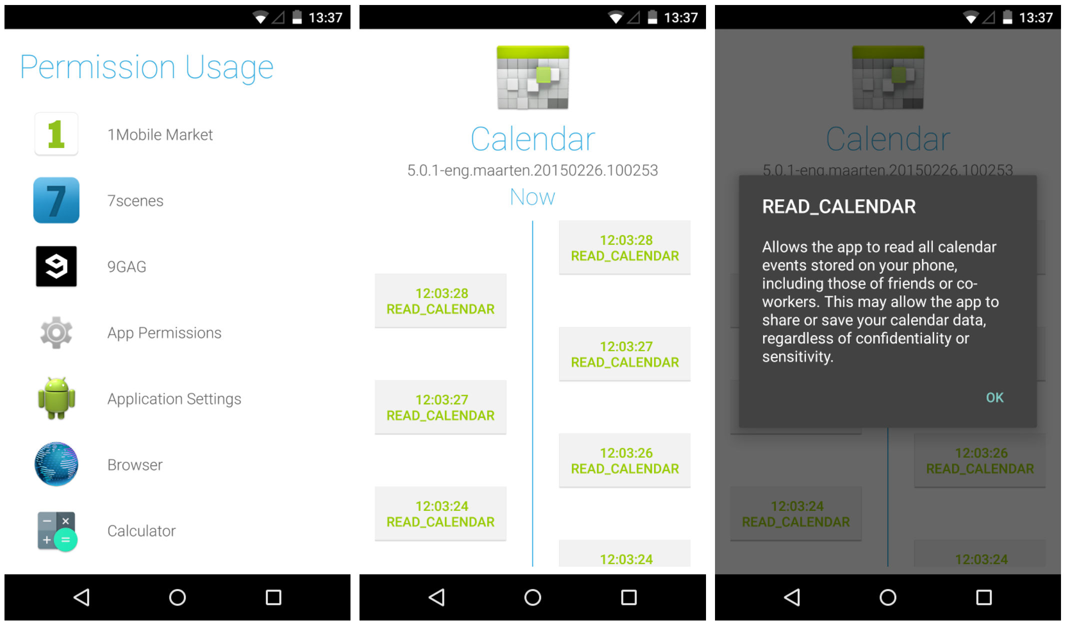
Table 4.1: WifiManager class methods

4.2 Design

The design was split up into two parts: the logging of permission usage and presenting it to the user in an application.

4.2.1 Technical design

The actual granting of permissions by the operating system takes place inside the Android framework: more specifically, within `AppOpsService`. This service contains two methods that handle the granting of permissions: `startOperation` and `noteOperation`. `startOperation` prepares for operations that will last for a longer time, like GPS and vibrator. `finishOperation` must be called when the operation is finished. `noteOperation` is just for short-term operations, like sending or receiving text message. Both methods ask `AppOpsManager` if the requested operation is allowed, if so (or not) a method can be called to log the use of a permission.
Android provides an internal SQLite database to applications that want to store data. Such a database can be used to log the permission usage to. In this case, the database can live either inside the Android framework or in the application that uses its data to present to the user. I decided to put the database inside the application, to lessen coupling between the two. In Chapter 2 I discussed the various Android components



READ_CALENDAR

Allows the app to read all calendar events stored on your phone, including those of friends or co-workers. This may allow the app to share or save your calendar data, regardless of confidentiality or sensitivity.

OK

Figure 4.2: This figure shows the proof of concept that visualizes permission usage.

available for building an application. For exposing the database to the framework I decided to use a Content Provider. The framework checks for presence of the Content Provider (so, if the application containing it is installed on the device) and makes use of its functions to write to the underlying database. This causes some problems though: permissions are also used during device boot up, but then the Content Provider is not available yet. This causes the operating system to get stuck in a boot loop, and an ugly fix is needed to prevent it from looping. A more solid solution would be to use Broadcast Intents for communication instead, because they do not care if the receiver is present or not.

4.2.2 User interface

Besides the framework modification I also developed an application that uses the database to present the permission usage to the user. Upon startup the application shows a list of installed applications that the user can browse, excluding system applications that are not visible to or runnable by the user. The user can scroll through this list and pick an application whose permission usage he wants to study. When the application actually used permissions a timeline is displayed, containing an overview of permission usage over time (Figure 4.2).

4.3 Implementation

I developed a proof of concept that makes it possible to log permission usage and present it to the user, using the design described in the previous section. Quick tests showed that it is possible to discover permission usage while the phone is on standby and permissions denied by the system because they were revoked in App Ops. However, more work is needed in order to improve the technical design and to implement more use cases, possibly with of a statistics module or automated analysis of the used permissions.

Chapter 5

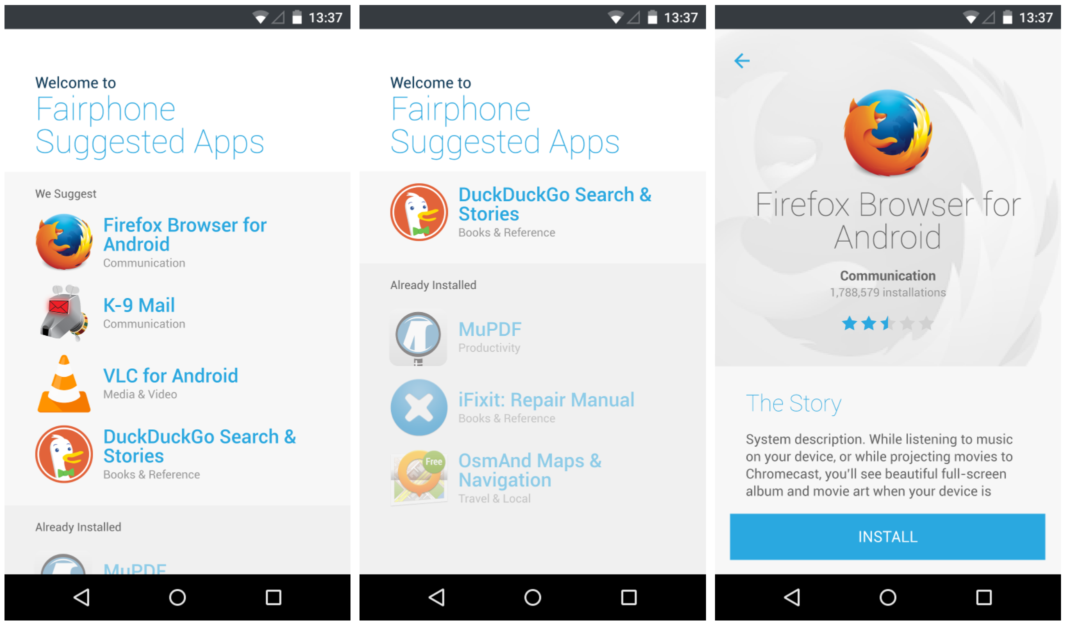
Engaging Users

The third and final project describes various ways to engage users, possibly incorporating the analyses from the first two projects. These projects had the goal to improve user awareness and comprehension increase transparency with respect to the Android permission system. The next step is to trigger users into discussing the privacy impact of the applications that they use and stimulating them to search for more privacy-friendly alternatives.

5.1 Forum

Internet fora are online discussion sites where people can hold conversations in the form of posted messages. These messages are often longer than one line of text, and are at least temporarily archived. A series of messages is called a topic or thread, and such topics are usually organized in a hierarchical structure. Moderators make sure users follow the rules and the forum stays organized. Fora often flourish into active communities with engaged members, sometimes even meeting offline. In these days fora might seem surpassed by social media, but the opposite is proven by communities like XDA Developers,[1] Gathering of Tweakers[2] and the Fairphone forum.[3]
The Fairphone forum is a place where Fairphone users can discuss and help each other with product questions, socialize and get to know each other and discuss fair aspects of the company and its phones. As opposed to most Internet fora, the Fairphone does not have subfora: there is only one list of topics. This is a feature of Discourse, the open source software the forum is comprised of. Another feature of Discourse is that it supports badges: users can earn badges for their first post, helping other users, etc. The forum is moderated by community members that have proven to have the necessary skills to manage the forum.

[1]http://forum.xda-developers.com/
[2]http://gathering.tweakers.net/
[3]https://forum.fairphone.com/



System description. While listening to music on your device, or while projecting movies to Chromecast, you'll see beautiful full-screen album and movie art when your device is

FIGURE 5.1: This figure shows the design for Fairphone Suggested Apps. The first two screens show the suggestions, sorted by already installed or not. Screen 3 shows the application screen, including the Privacy Impact.

The Fairphone forum could be an interesting place to discuss alternatives for bad privacy applications, based on the privacy impact or permission usage reports. Users could earn privacy awareness badges for suggesting alternatives and helping out other users, or contact developers on bad application privacy.

5.2 Suggested apps

In light of creating a custom application store in the future, Fairphone is developing a feature for Fairphone 2, called 'Suggested apps'. This is an application that informs users of alternative applications for all critical functions, like browser and e-mail. The requirements for these suggestions include open source, active, and privacy-friendly projects. A first version developed by Kwamecorp (Figure 5.1) includes the Privacy Impact and is currently being beta-tested.

5.3 Aggregated statistics

Statistics on permission usage by applications (gathered by the Permission Usage project) could be retrieved from every user and aggregated on a server. This way the server can form a complete image of how an application uses its requested permissions, including non-essential, one-time essential and essential permissions. This information can be presented to other users when they install such an application. This way users can make

better informed decisions before installation, based on data from other users.
A similar approach was studied by Yang et al. [30]. They propose crowdsourcing as a tool to enhance user comprehension of Android permissions. The tool they designed (Droidganger) provides information on permission usage to users using two techniques: record/replay and permission suppression. Their experiments showed that even with a simple visual detection of the differences in output, the approach can help users better understand about 40% of application permissions.
The statistics could also be used to detect bad behavior, if an application has a different permission usage profile than others. Applications' behavior could automatically be verified against aggregated behavior in order to spot fraudulent or infected applications. Combined with a feature that allows users to edit permissions (like App Ops) it becomes possible to serve permission profiles that tell the user which permissions they can safely turn off, without endangering the functionality of the application. Such profiles could relate to privacy (i.e. minimal data exposure), but also to child-friendliness or disabling in-app purchases.

Chapter 6

Discussion

In this chapter I summarize the findings of my research. I discuss the results, compare my research to previous reports and define unsettled points. I also discuss the announcement of Android M and its revised permission system.

Privacy Impact

The results of the first project demonstrate that it is possible to draw conclusions from the permissions applications request. This is in line with similar work done by [14–17]. I translated the permission requests into an easy to comprehend representation, the Privacy Impact. However, the algorithm I designed to calculate this representation is subject to discussion. In general, translating complex textual descriptions into simplified representations results in loss of information. This could lead users to blindly accept the simplification instead of trusting their own judgment. The algorithm is also unable to take into the account the intention of the developer: is the application using the permissions for benign purposes (e.g. the application's functionality) or for exfiltrating personal data to advertisers? Introducing multiple variables (such as heuristics of static code analysis results) or using a different scale (more levels, exponential instead of linear) could improve accuracy here.
In Chapter 3 I propose several ways to implement the Privacy Impact. The current implementation is arguably not be the most effective one, because the user has already installed the application at the time the Privacy Impact is displayed. This implementation offers users the same binary choice as they have in Google Play: either accept the privacy impact and start using the application, or cancel and look for an alternative. The most effective place to integrate the Privacy Impact would be in an application store, as confirmed in research by Benton et al. [13]. This way users can take the Privacy Impact into account at install-time, together with other factors that influence the selection of an application, such as user reviews and amount of downloads.

I conducted usability tests to measure if the Privacy Impact increases awareness and comprehension on Android permissions among users. The results show that the majority of the participants is not triggered by the permissions dialog in Google Play. However, the participants were triggered by the Privacy Impact on application startup. The SUS and MRC tests show that the participants desire the feature and positively respond to its usability. The results demonstrate that such a representation of permissions can increase awareness and comprehension with respect to Android permissions among users.

Permission Usage

The second project makes the actual use of permissions by applications visible. The proof of concept I developed demonstrates that it is possible to record permission usage and present it to the user in a comprehensible way. This information could be more useful than the static permission requests, because it shows how applications actually use permissions. The question is if users are capable of interpreting this information and translating it into actions that protect their privacy, which should be studied by conducting usability tests. If not, further research into automatically interpreting the data and deriving conclusions is needed, for example to detect overpriviliged applications or suspicious permission combinations.
Another way to increase transparency would be to prompt the user every time an application uses a permission. Such a system places the request for a permission in context, both in flow and in time. However, such a system could also lead to warning fatigue. It also requires the user to make split-second decisions, and it does not give a complete overview of what is happening.

Engaging Users

The third and final project explores various ways of starting a discussion among users on the privacy impact of the applications that they use, possibly incorporating the results from the first two projects. Suggestions include engaging users through a forum, stimulating them to research alternatives and using aggregated statistics to draw conclusions on the behavior of applications. The suggestions discussed are in a conceptual state and require further research in order to prove if they have the desired effect. Earlier research by Yang et al. [30] has shown that it is possible to crowdsource data on permission usage and present it to the user in a way that is actually understood by users.

Android M

During the course of this research Google announced the next version of Android, Android M. At the time of writing it is unclear what version number will be assigned to this new version, either Android 5.2 or Android 6. One of its new features is a revised permission system. Applications will no longer ask for permissions at install-time: instead, applications will ask for permissions the first time they need them. For example, if an application wants access to the microphone at some point, a dialog gives the user the option to either allow or deny the permission. If users change their mind later, they can toggle switches for each permission in an App Ops-like settings menu.
Ever since the hidden integration of App Ops in Android 4.3, Android users have been looking for granular application permission controls. With the new permission system, Google wants to put privacy and security back into the hands of users. The announcement confirms that Fairphone was on the right track with their integration of App Ops. However, there are some notable differences between App Ops and the new permission system.
One of the reasons Fairphone pulled App Ops was the fact that it could prevent applications from functioning properly. Android developers have been designing their applications for the current permission model: applications get all the permissions, or the user does not get the application. Developers could code applications to fail gracefully when individual permission are denied, but the App Ops feature did not allow that. This would result in complicated error messages or even application crashes, something the average Android user is not prepared for. Android Central discovered in their testing [34] that such catastrophic failure in the revised permission system was quite rare.
Another difference is that the revised permission system is less granular than App Ops. The new system bundles existing permissions into groups. The current list of groups includes Calendar, Camera, Contacts, Location, Microphone, Phone, Sensors and SMS. When an application runs on Android M (and supports the new system), it must issue a request to the operating system for a specific permission, at which point users are prompted to allow access to whatever group that permission belongs. Control is less granular than with App Ops, where users could, for example, allow the reading and writing of contacts separately.
Android Police notes [35] that an important permission group seems to be missing: the Internet permission group. As it turns out, users will never be asked to allow access to the outside world, and it is not possible to revoke it. The logic for this decision was based on the idea that if all of a user's data is protected by default, there is little reason to be concerned over applications communicating with the outside world. As for the specific `INTERNET` permission, it is still mandatory for applications that will access the Internet. Publishing an application without defining the permission in the Android manifest results in exceptions being thrown the first time a connection attempt is made, and the application could possibly crash.
The developer documentation for the Android M preview [36] explains that permissions

with the 'normal' protection level are automatically granted at install-time and will never require authorization from the user:

> ***Limited Permissions Granted at Install Time:*** *When the user installs or updates the app, the system grants the app all permissions that the app requests that fall under PROTECTION_NORMAL. For example, alarm clock and internet permissions fall under PROTECTION_NORMAL, so they are automatically granted at install time.*

This seems reasonable, since the permissions capable of interrupting a user's workflow are marked as 'dangerous'. But the `INTERNET` permission has always been classified as dangerous, at least up until now (see Appendix A). The logical assertion should be that the Internet permission will be downgraded for Android M.

From a usability perspective it makes sense to exclude a revocable Internet permission. These days almost every application seems to need Internet access for something. If setting up a device would mean hitting 'Allow' for every application, it would become a major annoyance. Google has made clear that the new approach to permissions is a balancing act, where they want the dialog to be shown to users only as often as is necessary.

A downside to this approach is that allowing access to a piece of information automatically means that it can be sent out over the Internet, even when that is not required. Imagine an application that manages the address book: access to the Contacts permission group is obviously necessary, but access to the Internet is not necessarily needed. Users would have to trust application developers not to send that data to a server.

But the real reason for not including a revocable Internet permission is advertising. A large number of free applications on Google Play only rely on the Internet to download and display advertisements. If every user could turn off Internet access, ad-supported applications would lose their only method of monetization, which would seriously discourage developers. In the process of blocking advertisements, users would also unknowingly disable other useful features like crash reporting and usage metrics, that are important parts of improving applications.

Android M's new permission system is not perfect, but it does offer new opportunities for a safer and more privacy-aware environment. The system is still under development, so it will probably undergo changes before (and after) it hits the market. A method for revoking the Internet permission might arise in the future, or another approach to restrict connectivity might be taken. Otherwise custom ROMs and mods will step up to take that place, as we have seen before.

Chapter 7

Conclusion and Future Work

In this thesis I explored various ways to improve the usability of the Android permission system, in order to increase user awareness and comprehension with respect to privacy. My suggestions for improvement were broken down into three projects:

1. **Privacy Impact**. Interprets and translates application permissions requests into an easy to comprehend graphic element, to address the issue of users not understanding permissions. Its effectiveness was demonstrated by conducting usability tests. The feature is currently scheduled for implementation into Fairphone OS for Fairphone 2.
2. **Permission Usage**: A modification to the Android framework that allows for recording of permission usage by applications. The data is presented to the user for review in an Android application. The project fixes the static information on permission usage. The proof of concept I developed will be open sourced and Fairphone is considering making it into a project in the future.
3. **Engaging Users**: Engages users into discussing the privacy impact of the applications that they use, possibly incorporating the results from the first two projects. Users are stimulated to search for alternative, and I also reasoned about application flows when permissions are revoked.

The usability tests conducted during the Privacy Impact project confirm that users have a hard time comprehending and assessing the Android permission system. Recent revisions, such as the ability to silently add permissions in application updates, have not improved upon this situation. However, the announcement of Android M and its revised permission system offers a promising take on improving usability and putting control back into the hands of users. From the perspective of a developer the Android permission system is a valuable tool to protect application components against unwanted access by other applications, although laziness and unclear documentation also result in

poorly specified permission requests. Permissions are also abused in order to retrieve personally identifiable data for tracking or advertising purposes.
The usability tests also showed that the translation of permissions into a easier to comprehend visual representation is a valuable effort in improving user awareness and comprehension. The permission usage project shows that it is possible to give insight into the actual use of permissions by applications after installation, which makes the Android permission system more transparent to the user. The results of these projects can be used to trigger users into discussing the privacy impact of the applications they use and stimulating them to search for alternatives, to find out what is truly fair. In general the outcome of this research shows that continued effort by Google or other parties into making Android permissions more accessible to the user is needed, so they will become a more effective privacy protecting tool.

Future Work

The development of permission comprehension tools for Android could be further explored.
The Privacy Impact project showed that it is possible to trigger the user by translating permissions into an easy to comprehend graphic element. However, the current implementation has its limits, so continued effort into making permissions comprehensible would be a valuable effort. First of all, it would be interesting to study the effect of improved calculation algorithms. Including or removing particular permissions, assigning different weights to combinations or using a different scale could improve accuracy. Finding a way to make clear to the user how the actual calculation is done would also be interesting, as some usability test participants wondered how Fairphone calculated the impact. Another limitation of the current implementation is the fact that it is not able to take into account the intentions of the developer: are the requested permissions needed for the functionality of the application, or are they used for different purposes (e.g. tracking)? It would be interesting to incorporate different elements in the algorithm, besides just the requested permissions. This way it might be possible to identify malware or viruses. From a usability perspective it would also be interesting to measure the effect of placing the Privacy Impact in an application store, to see if users take it into account when selecting applications. Finally, longer lasting usability tests (with a working prototype instead of a mockup) are needed to see if the initial welcoming of the Privacy Impact lasts.
Providing transparency on permission usage after application installation could be a valuable effort in increasing user comprehension of permissions. The proof of concept I built demonstrates that it is possible to record permission usage and present it to the user in a comprehensible way. Continued effort into implementing more use cases could result in a elaborate privacy panel. It would also be interesting to taint data, to see what data protected by a permission is actually being handled by an application. Usability

tests should be conducted to measure the desirability and effectiveness of such a feature. The third project made some suggestions for publishing the analyses from the first two projects, with the goal of starting a discussion among users on the privacy impact of the applications that they use. The suggestions I proposed are not exhaustive, so further research into interesting applications of both projects could be valuable. The actual design and implementation of the suggestions is also something that could be further explored, possibly combined with usability research to measure the effectiveness.

Appendix A

Android Permission Table

This table was generated from `AndroidManifest.xml`,[1] which is part of the Android framework. The XSL stylesheet I developed for this purpose can be found on GitHub.[2] Emphasized text indicates that an API is exposed only for use by bundled system applications.

[1]https://android.googlesource.com/platform/frameworks/base/+/master/core/res/AndroidManifest.xml
[2]https://github.com/dubwise101/manifesttransform

Permissions for things that cost money

Group	Description
COST_MONEY	Used for permissions that can be used to make the user spend money without their direct involvement.

Used as a flag for permissions that can cost money, like sending SMS messages.

Permissions for accessing messages

Group	Description	
MESSAGES	Used for permissions that allow an application to send messages on behalf of the user or intercept messages being received by the user. This is primarily intended for SMS/MMS messaging, such as receiving or reading an MMS.	
Permission	**Level**	**Description**
SEND_SMS	dangerous	Allows an application to send SMS messages.
SEND_RESPOND_VIA_MESSAGE	signature/ system	*Allows an application (Phone) to send a request to other applications to handle the respond-via-message action during incoming calls.* Not for use by third-party applications.
RECEIVE_SMS	dangerous	Allows an application to monitor incoming SMS messages, to record or perform processing on them.
RECEIVE_MMS	dangerous	Allows an application to monitor incoming MMS messages, to record or perform processing on them.
CARRIER_FILTER_SMS	signature/ system	Allows an application to filter carrier specific sms.
RECEIVE_EMERGENCY_BROADCAST	signature/ system	*Allows an application to receive emergency cell broadcast messages, to record or display them to the user.* Not for use by third-party applications. Pending API council approval
READ_CELL_BROADCASTS	dangerous	Allows an application to read previously received cell broadcast messages and to register a content observer to get notifications when a cell broadcast has been received and added to the database. For emergency alerts, the database is updated immediately after the alert dialog and notification sound/vibration/speech are presented. The “read” column is then updated after the user dismisses the alert. This enables supplementary emergency assistance apps to start loading additional emergency information (if Internet access is available) when the alert is first received, and to delay presenting the info to the user until after the initial alert dialog is dismissed. Pending API council approval
READ_SMS	dangerous	Allows an application to read SMS messages.
WRITE_SMS	dangerous	Allows an application to write SMS messages.

RECEIVE_WAP_PUSH	dangerous	Allows an application to monitor incoming WAP push messages.
RECEIVE_BLUETOOTH_MAP	signature/ system	

Permissions for accessing social info (contacts and social)

Group	**Description**
SOCIAL_INFO	Used for permissions that provide access to the user's social connections, such as contacts, call logs, social stream, etc. This includes both reading and writing of this data (which should generally be expressed as two distinct permissions).

Permission	**Level**	**Description**
READ_CONTACTS	dangerous	Allows an application to read the user's contacts data.
WRITE_CONTACTS	dangerous	Allows an application to write (but not read) the user's contacts data.
BIND_DIRECTORY_SEARCH	signature/ system	*Allows an application to execute contacts directory search. This should only be used by ContactsProvider.* Not for use by third-party applications.
READ_CALL_LOG	dangerous	Allows an application to read the user's call log. **Note:** If your app uses the READ_CONTACTS permission and *both* your minSdkVersion and targetSdkVersion values are set to 15 or lower, the system implicitly grants your app this permission. If you don't need this permission, be sure your targetSdkVersion is 16 or higher.
WRITE_CALL_LOG	dangerous	Allows an application to write (but not read) the user's contacts data. **Note:** If your app uses the WRITE_CONTACTS permission and *both* your minSdkVersion and targetSdkVersion values are set to 15 or lower, the system implicitly grants your app this permission. If you don't need this permission, be sure your targetSdkVersion is 16 or higher.
READ_SOCIAL_STREAM	dangerous	Allows an application to read from the user's social stream. This functionality will be unsupported in the future; cursors returned will be empty. Please do not use.
WRITE_SOCIAL_STREAM	dangerous	Allows an application to write (but not read) the user's social stream data. This functionality will be unsupported in the future; cursors returned will be empty. Please do not use.

Permissions for accessing information about the device owner

Group	**Description**

PERSONAL_INFO	Used for permissions that provide access to information about the device user such as profile information. This includes both reading and writing of this data (which should generally be expressed as two distinct permissions).

Permission	**Level**	**Description**
READ_PROFILE	dangerous	Allows an application to read the user's personal profile data.
WRITE_PROFILE	dangerous	Allows an application to write (but not read) the user's personal profile data.
BODY_SENSORS		Allows an application to access data from sensors that the user uses to measure what is happening inside his/her body, such as heart rate.

Permissions for accessing the device calendar

Group	**Description**
CALENDAR	Used for permissions that provide access to the device calendar to create / view events.

Permission	**Level**	**Description**
READ_CALENDAR	dangerous	Allows an application to read the user's calendar data.
WRITE_CALENDAR	dangerous	Allows an application to write (but not read) the user's calendar data.

Permissions for accessing the user dictionary

Group	**Description**
USER_DICTIONARY	Used for permissions that provide access to the user calendar to create / view events.

Permission	**Level**	**Description**
READ_USER_DICTIONARY	dangerous	Allows an application to read the user dictionary. This should really only be required by an IME, or a dictionary editor like the Settings app.
WRITE_USER_DICTIONARY	normal	Allows an application to write to the user dictionary.

Permissions for accessing the user bookmarks

Group	**Description**

BOOKMARKS	Used for permissions that provide access to the user bookmarks and browser history.

Permission	**Level**	**Description**
READ_HISTORY_BOOKMARKS	dangerous	Allows an application to read (but not write) the user's browsing history and bookmarks.
WRITE_HISTORY_BOOKMARKS	dangerous	Allows an application to write (but not read) the user's browsing history and bookmarks.

Permissions for setting the device alarm

Group	**Description**
DEVICE_ALARMS	Used for permissions that provide access to device alarms.

Permission	**Level**	**Description**
ermission.SET_ALARM	normal	Allows an application to broadcast an Intent to set an alarm for the user.

Permissions for accessing the user voicemail

Group	**Description**
VOICEMAIL	Used for permissions that provide access to the user voicemail box.

Permission	**Level**	**Description**
ADD_VOICEMAIL	dangerous	Allows an application to add voicemails into the system.
WRITE_VOICEMAIL	system/ signature	Allows an application to modify and remove existing voicemails in the system
READ_VOICEMAIL	system/ signature	Allows an application to read voicemails in the system.

Permissions for enabling accessibility features

Group	**Description**
ACCESSIBILITY_FEATURES	Used for permissions that allow requesting certain accessibility features.

Used for permissions that allow requesting certain accessibility features, like a screen reader.

Permissions for accessing location info

Group	Description
`LOCATION`	Used for permissions that allow access to the user's current location.

Permission	Level	Description
`ACCESS_FINE_LOCATION`	dangerous	Allows an app to access precise location from location sources such as GPS, cell towers, and Wi-Fi.
`ACCESS_COARSE_LOCATION`	dangerous	Allows an app to access approximate location derived from network location sources such as cell towers and Wi-Fi.
`ACCESS_MOCK_LOCATION`	dangerous	Allows an application to create mock location providers for testing
`ACCESS_LOCATION_EXTRA_COMMANDS`	normal	Allows an application to access extra location provider commands
`INSTALL_LOCATION_PROVIDER`	signature/ system	*Allows an application to install a location provider into the Location Manager.* Not for use by third-party applications.
`HDMI_CEC`	signature OrSystem	*Allows HDMI-CEC service to access device and configuration files. This should only be used by HDMI-CEC service.*
`LOCATION_HARDWARE`	signature/ system	*Allows an application to use location features in hardware, such as the geofencing api.* Not for use by third-party applications.

Permissions for accessing networks

Group	Description
`NETWORK`	Used for permissions that provide access to networking services. The main permission here is internet access, but this is also an appropriate group for accessing or modifying any network configuration or other related network operations.

Permission	Level	Description
`INTERNET`	dangerous	Allows applications to open network sockets.
`ACCESS_NETWORK_STATE`	normal	Allows applications to access information about networks
`ACCESS_WIFI_STATE`	normal	Allows applications to access information about Wi-Fi networks
`CHANGE_WIFI_STATE`	dangerous	Allows applications to change Wi-Fi connectivity state
`READ_WIFI_CREDENTIAL`	signature/ system	*Allows applications to read Wi-Fi credential.* Not for use by third-party applications.
`ACCESS_WIMAX_STATE`	normal	
`CHANGE_WIMAX_STATE`	dangerous	

SCORE_NETWORKS	signature/ system	

Permissions for short range, peripheral networks

Group	Description	
BLUETOOTH_NETWORK	Used for permissions that provide access to other devices through Bluetooth.	
Permission	**Level**	**Description**
BLUETOOTH	dangerous	Allows applications to connect to paired bluetooth devices
BLUETOOTH_ADMIN	dangerous	Allows applications to discover and pair bluetooth devices
BLUETOOTH_PRIVILEGED	system/ signature	*Allows applications to pair bluetooth devices without user interaction, and to allow or disallow phonebook access or message access. This is not available to third party applications.*
BLUETOOTH_MAP	signature	Control access to email providers exclusively for Bluetooth
BLUETOOTH_STACK	signature	Allows bluetooth stack to access files This should only be used by Bluetooth apk.
NFC	dangerous	Allows applications to perform I/O operations over NFC
CONNECTIVITY_INTERNAL	signature/ system	*Allows an internal user to use privileged ConnectivityManager APIs.*
RECEIVE_DATA_ACTIVITY_CHANGE	signature/ system	
LOOP_RADIO	signature/ system	*Allows access to the loop radio (Android@Home mesh network) device.*
NFC_HANDOVER_STATUS	signature/ system	Allows sending and receiving handover transfer status from Wifi and Bluetooth

Permissions for accessing accounts

Group	Description	
ACCOUNTS	Permissions for direct access to the accounts managed by the Account Manager.	
Permission	**Level**	**Description**

`GET_ACCOUNTS`	normal	Allows access to the list of accounts in the Accounts Service
`AUTHENTICATE_ACCOUNTS`	dangerous	Allows an application to act as an AccountAuthenticator for the AccountManager
`USE_CREDENTIALS`	dangerous	Allows an application to request authtokens from the AccountManager
`MANAGE_ACCOUNTS`	dangerous	Allows an application to manage the list of accounts in the AccountManager
`ACCOUNT_MANAGER`	signature	*Allows applications to call into AccountAuthenticators.* Not for use by third-party applications.

Permissions for accessing hardware that may effect battery life

Group	Description
`AFFECTS_BATTERY`	Used for permissions that provide direct access to the hardware on the device that has an effect on battery life. This includes vibrator, flashlight, etc.

Permission	Level	Description
`CHANGE_WIFI_MULTICAST_STATE`	dangerous	Allows applications to enter Wi-Fi Multicast mode
`VIBRATE`	normal	Allows access to the vibrator
`FLASHLIGHT`	normal	Allows access to the flashlight
`WAKE_LOCK`	normal	Allows using PowerManager WakeLocks to keep processor from sleeping or screen from dimming
`TRANSMIT_IR`	normal	Allows using the device's IR transmitter, if available

Permissions related to changing audio settings

Group	Description
`AUDIO_SETTINGS`	Used for permissions that provide direct access to speaker settings the device.

Permission	Level	Description
`MODIFY_AUDIO_SETTINGS`	normal	Allows an application to modify global audio settings

Permissions for accessing hardware

Group	Description	
HARDWARE_CONTROLS	Used for permissions that provide direct access to the hardware on the device. This includes audio, the camera, vibrator, etc.	
Permission	**Level**	**Description**
MANAGE_USB	signature/ system	*Allows an application to manage preferences and permissions for USB devices*
ACCESS_MTP	signature/ system	*Allows an application to access the MTP USB kernel driver. For use only by the device side MTP implementation.*
HARDWARE_TEST	signature	Allows access to hardware peripherals. Intended only for hardware testing. Not for use by third-party applications.
ACCESS_FM_RADIO	signature/ system	*Allows access to FM This is not a third-party API (intended for system apps).*
NET_ADMIN	signature	Allows access to configure network interfaces, configure/use IPSec, etc.
REMOTE_AUDIO_PLAYBACK	signature	Allows registration for remote audio playback.
TV_INPUT_HARDWARE	signature OrSystem	*Allows TvInputService to access underlying TV input hardware such as built-in tuners and HDMI-in's. This should only be used by OEM's TvInputService's.*
CAPTURE_TV_INPUT	signature OrSystem	*Allows to capture a frame of TV input hardware such as built-in tuners and HDMI-in's.* Not for use by third-party applications.
OEM_UNLOCK_STATE	signature	Allows enabling/disabling OEM unlock Not for use by third-party applications.
ACCESS_PDB_STATE	signature	Allows querying state of PersistentDataBlock Not for use by third-party applications.

Permissions associated with audio capture

Group	Description	
MICROPHONE	Used for permissions that are associated with accessing microphone audio from the device. Note that phone calls also capture audio but are in a separate (more visible) permission group.	
Permission	**Level**	**Description**
RECORD_AUDIO	dangerous	Allows an application to record audio

Permissions associated with camera and image capture

Group	**Description**	
CAMERA	Used for permissions that are associated with accessing camera or capturing images/video from the device.	
Permission	**Level**	**Description**
CAMERA	dangerous	Required to be able to access the camera device. This will automatically enforce the `<uses-feature>` manifest element for *all* camera features. If you do not require all camera features or can properly operate if a camera is not available, then you must modify your manifest as appropriate in order to install on devices that don't support all camera features.
CAMERA_DISABLE_TRANSMIT_LED	signature/ system	*Allows disabling the transmit-indicator LED that is normally on when a camera is in use by an application.*

Permissions associated with telephony state

Group	**Description**	
PHONE_CALLS	Used for permissions that are associated with accessing and modifying telephony state: placing calls, intercepting outgoing calls, reading and modifying the phone state.	
Permission	**Level**	**Description**
PROCESS_OUTGOING_CALLS	dangerous	Allows an application to see the number being dialed during an outgoing call with the option to redirect the call to a different number or abort the call altogether.
MODIFY_PHONE_STATE	signature/ system	*Allows modification of the telephony state - power on, mmi, etc. Does not include placing calls.* Not for use by third-party applications.
READ_PHONE_STATE	dangerous	Allows read only access to phone state. **Note:** If *both* your minSdkVersion and targetSdkVersion values are set to 3 or lower, the system implicitly grants your app this permission. If you don't need this permission, be sure your targetSdkVersion is 4 or higher.
READ_PRECISE_PHONE_STATE	signature/ system	Allows read only access to precise phone state. Pending API council approval
READ_PRIVILEGED_PHONE_STATE	signature/ system	*Allows read access to privileged phone state. Used internally.*

`CALL_PHONE`	dangerous	Allows an application to initiate a phone call without going through the Dialer user interface for the user to confirm the call being placed.
`USE_SIP`	dangerous	Allows an application to use SIP service
`REGISTER_SIM_SUBSCRIPTION`	system/ signature	*Protects the ability to register any PhoneAccount with PhoneAccount CAPABILITY_SIM_SUBSCRIPTION. This capability indicates that the PhoneAccount corresponds to a device SIM.*
`REGISTER_CALL_PROVIDER`	system/ signature	*Protects the ability to register any PhoneAccount with PhoneAccount CAPABILITY_CALL_PROVIDER.*
`REGISTER_CONNECTION_MANAGER`	system/ signature	*Protects the ability to register any PhoneAccount with PhoneAccount CAPABILITY_CONNECTION_MANAGER*
`BIND_INCALL_SERVICE`	system/ signature	*Allows an application to bind to InCallService implementations.*
`BIND_CONNECTION_SERVICE`	system/ signature	*Allows an application to bind to ConnectionService implementations.*
`CONTROL_INCALL_EXPERIENCE`	system/ signature	*Allows an application to control the in-call experience.*

Permissions for sdcard interaction

Group	**Description**	
`STORAGE`	Group of permissions that are related to SD card access.	
Permission	**Level**	**Description**

`READ_EXTERNAL_STORAGE`	normal	Allows an application to read from external storage. Any app that declares the WRITE_EXTERNAL_STORAGE permission is implicitly granted this permission. This permission is enforced starting in API level 19. Before API level 19, this permission is not enforced and all apps still have access to read from external storage. You can test your app with the permission enforced by enabling *Protect USB storage* under Developer options in the Settings app on a device running Android 4.1 or higher. Also starting in API level 19, this permission is *not* required to read/write files in your application-specific directories returned by android.content.ContextgetExternalFilesDir and android.content.ContextgetExternalCacheDir. **Note:** If *both* your minSdkVersion and targetSdkVersion values are set to 3 or lower, the system implicitly grants your app this permission. If you don't need this permission, be sure your targetSdkVersion is 4 or higher.
`WRITE_EXTERNAL_STORAGE`	dangerous	Allows an application to write to external storage. **Note:** If *both* your minSdkVersion and targetSdkVersion values are set to 3 or lower, the system implicitly grants your app this permission. If you don't need this permission, be sure your targetSdkVersion is 4 or higher. Starting in API level 19, this permission is *not* required to read/write files in your application-specific directories returned by android.content.ContextgetExternalFilesDir and android.content.ContextgetExternalCacheDir.
`WRITE_MEDIA_STORAGE`	signature/ system	*Allows an application to write to internal media storage*
`MANAGE_DOCUMENTS`	signature	Allows an application to manage access to documents, usually as part of a document picker.

Permissions for screenlock

Group	**Description**	
`SCREENLOCK`	Group of permissions that are related to the screenlock.	
Permission	**Level**	**Description**
`DISABLE_KEYGUARD`	dangerous	Allows applications to disable the keyguard

Permissions to access other installed applications

Group	Description
APP_INFO	Group of permissions that are related to the other applications installed on the system. Examples include such as listing running apps, or killing background processes.

Permission	Level	Description
GET_TASKS	normal	No longer enforced.
REAL_GET_TASKS	signature/ system	New version of GET_TASKS that apps can request, since GET_TASKS doesn't really give access to task information. We need this new one because there are many existing apps that use add libraries and such that have validation code to ensure the app has requested the GET_TASKS permission by seeing if it has been granted the permission... if it hasn't, it kills the app with a message about being upset. So we need to have it continue to look like the app is getting that permission, even though it will never be checked, and new privileged apps can now request this one for real access.
START_TASKS_FROM_RECENTS	signature/ system	Allows an application to start a task from a ActivityManagerRecentTaskInfo.
INTERACT_ACROSS_USERS	signature/ system/ development	*Allows an application to call APIs that allow it to do interactions across the users on the device, using singleton services and user-targeted broadcasts. This permission is not available to third party applications.*
INTERACT_ACROSS_USERS_FULL	signature	Fuller form of android.Manifest.permission INTERACT_ACROSS_USERS that removes restrictions on where broadcasts can be sent and allows other types of interactions.
MANAGE_USERS	signature/ system	*Allows an application to call APIs that allow it to query and manage users on the device. This permission is not available to third party applications.*
GET_DETAILED_TASKS	signature	Allows an application to get full detailed information about recently running tasks, with full fidelity to the real state.
REORDER_TASKS	normal	Allows an application to change the Z-order of tasks
REMOVE_TASKS	signature	Allows an application to change to remove/kill tasks
MANAGE_ACTIVITY_STACKS	signature/ system	*Allows an application to create/manage/remove stacks*
START_ANY_ACTIVITY	signature	Allows an application to start any activity, regardless of permission protection or exported state.
RESTART_PACKAGES	normal	The android.app.ActivityManagerrestartPackage API is no longer supported.

KILL_BACKGROUND_PROCESSES	normal	Allows an application to call android.app.ActivityManagerkillBackgroundProcesses.

Permissions affecting the display of other applications

Group	Description
DISPLAY	Group of permissions that allow manipulation of how another application displays UI to the user.

Permission	Level	Description
SYSTEM_ALERT_WINDOW	dangerous	Allows an application to open windows using the type android.view.WindowManager.LayoutParams TYPE_SYSTEM_ALERT, shown on top of all other applications. Very few applications should use this permission; these windows are intended for system-level interaction with the user.

Permissions affecting the system wallpaper

Group	Description
WALLPAPER	Group of permissions that allow manipulation of how another application displays UI to the user.

Permission	Level	Description
SET_WALLPAPER	normal	Allows applications to set the wallpaper
SET_WALLPAPER_HINTS	normal	Allows applications to set the wallpaper hints

Permissions for changing the system clock

Group	Description
SYSTEM_CLOCK	Group of permissions that are related to system clock.

Permission	Level	Description
SET_TIME	signature/ system	*Allows applications to set the system time.* Not for use by third-party applications.
SET_TIME_ZONE	normal	Allows applications to set the system time zone

Permissions related to changing status bar

Group	**Description**	
STATUS_BAR	Used for permissions that change the status bar	
Permission	**Level**	**Description**
EXPAND_STATUS_BAR	normal	Allows an application to expand or collapse the status bar.

Permissions related to adding/removing shortcuts from Launcher

Permission	**Level**	**Description**
INSTALL_SHORTCUT	dangerous	Allows an application to install a shortcut in Launcher
UNINSTALL_SHORTCUT	dangerous	Allows an application to uninstall a shortcut in Launcher

Permissions related to accessing sync settings

Group	**Description**	
SYNC_SETTINGS	Used for permissions that access the sync settings or sync related information.	
Permission	**Level**	**Description**
READ_SYNC_SETTINGS	normal	Allows applications to read the sync settings
WRITE_SYNC_SETTINGS	normal	Allows applications to write the sync settings
READ_SYNC_STATS	normal	Allows applications to read the sync stats

Permissions for low-level system interaction

Group	**Description**
SYSTEM_TOOLS	Group of permissions that are related to system APIs. Many of these are not permissions the user will be expected to understand, and such permissions should generally be marked as ”normal” protection level so they don’t get displayed. This can also, however, be used for miscellaneous features that provide access to the operating system, such as writing the global system settings.

Permission	Level	Description
SET_SCREEN_COMPATIBILITY	signature	*Change the screen compatibility mode of applications*
ACCESS_ALL_EXTERNAL_STORAGE	signature	Allows an application to access all multi-user external storage
CHANGE_CONFIGURATION	signature/ system/ development	*Allows an application to modify the current configuration, such as locale.*
WRITE_SETTINGS	normal	Allows an application to read or write the system settings.
WRITE_GSERVICES	signature/ system	*Allows an application to modify the Google service map. Not for use by third-party applications.*
FORCE_STOP_PACKAGES	signature/ system	*Allows an application to call android.app.ActivityManagerforceStopPackage.*
RETRIEVE_WINDOW_CONTENT	signature/ system	*Allows an application to retrieve the content of the active window An active window is the window that has fired an accessibility event.*
SET_ANIMATION_SCALE	signature/ system/ development	*Modify the global animation scaling factor.* Not for use by third-party applications.
PERSISTENT_ACTIVITY	normal	This functionality will be removed in the future; please do not use. Allow an application to make its activities persistent.
GET_PACKAGE_SIZE	normal	Allows an application to find out the space used by any package.
SET_PREFERRED_APPLICATIONS	signature	No longer useful, see android.content.pm.PackageManageraddPackageToPreferred for details.
RECEIVE_BOOT_COMPLETED	normal	Allows an application to receive the android.content.Intent ACTION_BOOT_COMPLETED that is broadcast after the system finishes booting. If you don't request this permission, you will not receive the broadcast at that time. Though holding this permission does not have any security implications, it can have a negative impact on the user experience by increasing the amount of time it takes the system to start and allowing applications to have themselves running without the user being aware of them. As such, you must explicitly declare your use of this facility to make that visible to the user.
BROADCAST_STICKY	normal	Allows an application to broadcast sticky intents. These are broadcasts whose data is held by the system after being finished, so that clients can quickly retrieve that data without having to wait for the next broadcast.

MOUNT_UNMOUNT_FILESYSTEMS	system/ signature	*Allows mounting and unmounting file systems for removable storage.* Not for use by third-party applications.
MOUNT_FORMAT_FILESYSTEMS	system/ signature	*Allows formatting file systems for removable storage.* Not for use by third-party applications.
ASEC_ACCESS	signature	Allows access to ASEC non-destructive API calls
ASEC_CREATE	signature	Allows creation of ASEC volumes
ASEC_DESTROY	signature	Allows destruction of ASEC volumes
ASEC_MOUNT_UNMOUNT	signature	Allows mount / unmount of ASEC volumes
ASEC_RENAME	signature	Allows rename of ASEC volumes
WRITE_APN_SETTINGS	signature/ system	*Allows applications to write the apn settings.* Not for use by third-party applications.
SUBSCRIBED_FEEDS_READ	normal	Allows an application to allow access the subscribed feeds ContentProvider.
SUBSCRIBED_FEEDS_WRITE	dangerous	Allows an application to allow access the subscribed feeds ContentProvider.
CHANGE_NETWORK_STATE	normal	Allows applications to change network connectivity state
CLEAR_APP_CACHE	dangerous	Allows an application to clear the caches of all installed applications on the device.
ALLOW_ANY_CODEC_FOR_PLAYBACK	signature/ system	*Allows an application to use any media decoder when decoding for playback*
MANAGE_CA_CERTIFICATES	signature/ system	*Allows an application to install and/or uninstall CA certificates on behalf of the user.*
RECOVERY	signature/ system	*Allows an application to do certain operations needed for interacting with the recovery (system update) system.*
BIND_JOB_SERVICE	signature	Allows the system to bind to an application's task services

Permissions for special development tools

Group	**Description**	
DEVELOPMENT_TOOLS	Group of permissions that are related to development features. These are not permissions that should appear in third-party applications; they protect APIs that are intended only to be used for development purposes.	
Permission	**Level**	**Description**

`WRITE_SECURE_SETTINGS`	signature/ system/ development	*Allows an application to read or write the secure system settings.* Not for use by third-party applications.
`DUMP`	signature/ system/ development	*Allows an application to retrieve state dump information from system services.* Not for use by third-party applications.
`READ_LOGS`	signature/ system/ development	*Allows an application to read the low-level system log files.* Not for use by third-party applications, because Log entries can contain the user's private information.
`SET_DEBUG_APP`	signature/ system/ development	*Configure an application for debugging.* Not for use by third-party applications.
`SET_PROCESS_LIMIT`	signature/ system/ development	*Allows an application to set the maximum number of (not needed) application processes that can be running.* Not for use by third-party applications.
`SET_ALWAYS_FINISH`	signature/ system/ development	*Allows an application to control whether activities are immediately finished when put in the background.* Not for use by third-party applications.
`SIGNAL_PERSISTENT_PROCESSES`	signature/ system/ development	*Allow an application to request that a signal be sent to all persistent processes.* Not for use by third-party applications.

Private (signature-only) permissions

Permission	**Level**	**Description**
`DIAGNOSTIC`	signature	*Allows applications to RW to diagnostic resources.* Not for use by third-party applications.
`STATUS_BAR`	signature/ system	*Allows an application to open, close, or disable the status bar and its icons.* Not for use by third-party applications.
`STATUS_BAR_SERVICE`	signature	Allows an application to be the status bar. Currently used only by SystemUI.apk

`FORCE_BACK`	signature	Allows an application to force a BACK operation on whatever is the top activity. Not for use by third-party applications.
`UPDATE_DEVICE_STATS`	signature/ system	*Allows an application to update device statistics.* Not for use by third-party applications.
`GET_APP_OPS_STATS`	signature/ system/ development	*Allows an application to collect battery statistics*
`UPDATE_APP_OPS_STATS`	signature/ system	*Allows an application to update application operation statistics. Not for use by third party apps.*
`INTERNAL_SYSTEM_WINDOW`	signature	Allows an application to open windows that are for use by parts of the system user interface. Not for use by third-party applications.
`MANAGE_APP_TOKENS`	signature	Allows an application to manage (create, destroy, Z-order) application tokens in the window manager. Not for use by third-party applications.
`FREEZE_SCREEN`	signature	Allows the application to temporarily freeze the screen for a full-screen transition.
`INJECT_EVENTS`	signature	Allows an application to inject user events (keys, touch, trackball) into the event stream and deliver them to ANY window. Without this permission, you can only deliver events to windows in your own process. Not for use by third-party applications.
`FILTER_EVENTS`	signature	Allows an application to register an input filter which filters the stream of user events (keys, touch, trackball) before they are dispatched to any window.
`RETRIEVE_WINDOW_TOKEN`	signature	Allows an application to retrieve the window token from the accessibility manager.
`FRAME_STATS`	signature	Allows an application to collect frame statistics
`TEMPORARY_ENABLE_ACCESSIBILITY`	signature	Allows an application to temporary enable accessibility on the device.
`SET_ACTIVITY_WATCHER`	signature	Allows an application to watch and control how activities are started globally in the system. Only for is in debugging (usually the monkey command). Not for use by third-party applications.
`SHUTDOWN`	signature/ system	*Allows an application to call the activity manager shutdown() API to put the higher-level system there into a shutdown state.*
`STOP_APP_SWITCHES`	signature/ system	*Allows an application to tell the activity manager to temporarily stop application switches, putting it into a special mode that prevents applications from immediately switching away from some critical UI such as the home screen.*

`GET_TOP_ACTIVITY_INFO`	signature	Allows an application to retrieve private information about the current top activity, such as any assist context it can provide. Not for use by third-party applications.
`READ_INPUT_STATE`	signature	Allows an application to retrieve the current state of keys and switches. Not for use by third-party applications. The API that used this permission has been removed.
`BIND_INPUT_METHOD`	signature	Must be required by an android.inputmethodservice.InputMethodService, to ensure that only the system can bind to it.
`BIND_ACCESSIBILITY_SERVICE`	signature	Must be required by an android.accessibilityservice.AccessibilityService, to ensure that only the system can bind to it.
`BIND_PRINT_SERVICE`	signature	Must be required by a android.printservice.PrintService, to ensure that only the system can bind to it.
`BIND_NFC_SERVICE`	signature	Must be required by a android.nfc.cardemulation.HostApduService or android.nfc.cardemulation.OffHostApduService to ensure that only the system can bind to it.
`BIND_PRINT_SPOOLER_SERVICE`	signature	Must be required by the PrintSpooler to ensure that only the system can bind to it.
`BIND_TEXT_SERVICE`	signature	Must be required by a TextService (e.g. SpellCheckerService) to ensure that only the system can bind to it.
`BIND_VPN_SERVICE`	signature	Must be required by a android.net.VpnService, to ensure that only the system can bind to it.
`BIND_WALLPAPER`	signature/ system	Must be required by a android.service.wallpaper.WallpaperService, to ensure that only the system can bind to it.
`BIND_VOICE_INTERACTION`	signature	Must be required by a android.service.voice.VoiceInteractionService, to ensure that only the system can bind to it.
`MANAGE_VOICE_KEYPHRASES`	signature/ system	Must be required by hotword enrollment application, to ensure that only the system can interact with it. Not for use by third-party applications.
`BIND_REMOTE_DISPLAY`	signature	Must be required by a com.android.media.remotedisplay.RemoteDisplayProvider, to ensure that only the system can bind to it.
`BIND_TV_INPUT`	signature/ system	Must be required by a android.media.tv.TvInputService to ensure that only the system can bind to it.
`MODIFY_PARENTAL_CONTROLS`	signature/ system	*Allows an application to modify parental controls* Not for use by third-party applications.
`BIND_DEVICE_ADMIN`	signature	Must be required by device administration receiver, to ensure that only the system can interact with it.
`MANAGE_DEVICE_ADMINS`	signature/ system	*Required to add or remove another application as a device admin.* Not for use by third-party applications.

`SET_ORIENTATION`	signature	Allows low-level access to setting the orientation (actually rotation) of the screen. Not for use by third-party applications.
`SET_POINTER_SPEED`	signature	Allows low-level access to setting the pointer speed. Not for use by third-party applications.
`SET_INPUT_CALIBRATION`	signature	Allows low-level access to setting input device calibration. Not for use by normal applications.
`SET_KEYBOARD_LAYOUT`	signature	Allows low-level access to setting the keyboard layout. Not for use by third-party applications.
`INSTALL_PACKAGES`	signature/ system	*Allows an application to install packages.* Not for use by third-party applications.
`CLEAR_APP_USER_DATA`	signature	Allows an application to clear user data. Not for use by third-party applications.
`DELETE_CACHE_FILES`	signature/ system	*Allows an application to delete cache files.* Not for use by third-party applications.
`DELETE_PACKAGES`	signature/ system	*Allows an application to delete packages.* Not for use by third-party applications.
`MOVE_PACKAGE`	signature/ system	*Allows an application to move location of installed package.*
`CHANGE_COMPONENT_ENABLED_STATE`	signature/ system	*Allows an application to change whether an application component (other than its own) is enabled or not.* Not for use by third-party applications.
`GRANT_REVOKE_PERMISSIONS`	signature	Allows an application to grant or revoke specific permissions.
`ACCESS_SURFACE_FLINGER`	signature	Allows an application to use SurfaceFlinger's low level features. Not for use by third-party applications.
`READ_FRAME_BUFFER`	signature/ system	*Allows an application to take screen shots and more generally get access to the frame buffer data.* Not for use by third-party applications.
`ACCESS_INPUT_FLINGER`	signature	Allows an application to use InputFlinger's low level features.
`CONFIGURE_WIFI_DISPLAY`	signature	Allows an application to configure and connect to Wifi displays
`CONTROL_WIFI_DISPLAY`	signature	Allows an application to control low-level features of Wifi displays such as opening an RTSP socket. This permission should only be used by the display manager.
`CONTROL_VPN`	signature/ system	*Allows an application to control VPN.* Not for use by third-party applications.

`CAPTURE_AUDIO_OUTPUT`	signature/ system	*Allows an application to capture audio output.* Not for use by third-party applications.
`CAPTURE_AUDIO_HOTWORD`	signature/ system	*Allows an application to capture audio for hotword detection.* Not for use by third-party applications.
`MODIFY_AUDIO_ROUTING`	signature/ system	*Allows an application to modify audio routing and override policy decisions.* Not for use by third-party applications.
`CAPTURE_VIDEO_OUTPUT`	signature/ system	*Allows an application to capture video output.* Not for use by third-party applications.
`CAPTURE_SECURE_VIDEO_OUTPUT`	signature/ system	*Allows an application to capture secure video output.* Not for use by third-party applications.
`MEDIA_CONTENT_CONTROL`	signature/ system	*Allows an application to know what content is playing and control its playback.* Not for use by third-party applications due to privacy of media consumption
`BRICK`	signature	Required to be able to disable the device (very dangerous!). Not for use by third-party applications..
`REBOOT`	signature/ system	*Required to be able to reboot the device.* Not for use by third-party applications.
`DEVICE_POWER`	signature	Allows low-level access to power management. Not for use by third-party applications.
`USER_ACTIVITY`	signature/ system	*Allows access to the PowerManager.userActivity function.* Not for use by third-party applications.
`NET_TUNNELING`	signature	Allows low-level access to tun tap driver
`FACTORY_TEST`	signature	Run as a manufacturer test application, running as the root user. Only available when the device is running in manufacturer test mode. Not for use by third-party applications.
`BROADCAST_PACKAGE_REMOVED`	signature	Allows an application to broadcast a notification that an application package has been removed. Not for use by third-party applications.
`BROADCAST_SMS`	signature	Allows an application to broadcast an SMS receipt notification. Not for use by third-party applications.
`BROADCAST_WAP_PUSH`	signature	Allows an application to broadcast a WAP PUSH receipt notification. Not for use by third-party applications.

`BROADCAST_NETWORK_PRIVILEGED`	signature/ system	*Allows an application to broadcast privileged networking requests.* Not for use by third-party applications.
`MASTER_CLEAR`	signature/ system	*Not for use by third-party applications.*
`CALL_PRIVILEGED`	signature/ system	*Allows an application to call any phone number, including emergency numbers, without going through the Dialer user interface for the user to confirm the call being placed.* Not for use by third-party applications.
`PERFORM_CDMA_PROVISIONING`	signature/ system	*Allows an application to perform CDMA OTA provisioning*
`CONTROL_LOCATION_UPDATES`	signature/ system	*Allows enabling/disabling location update notifications from the radio.* Not for use by third-party applications.
`ACCESS_CHECKIN_PROPERTIES`	signature/ system	*Allows read/write access to the "properties" table in the checkin database, to change values that get uploaded.* Not for use by third-party applications.
`PACKAGE_USAGE_STATS`	signature/ development appop	*Allows an application to collect component usage statistics*
`BATTERY_STATS`	signature/ system/ development	*Allows an application to collect battery statistics*
`BACKUP`	signature/ system	*Allows an application to control the backup and restore process.* Not for use by third-party applications. pending API council
`CONFIRM_FULL_BACKUP`	signature	Allows a package to launch the secure full-backup confirmation UI. Only the system process may hold this permission.
`BIND_REMOTEVIEWS`	signature/ system	*Must be required by a android.widget.RemoteViewsService, to ensure that only the system can bind to it.*
`BIND_APPWIDGET`	signature/ system	*Allows an application to tell the AppWidget service which application can access AppWidget's data. The normal user flow is that a user picks an AppWidget to go into a particular host, thereby giving that host application access to the private data from the AppWidget app. An application that has this permission should honor that contract.* *Not for use by third-party applications.*

BIND_KEYGUARD_APPWIDGET	signature/ system	*Private permission, to restrict who can bring up a dialog to add a new keyguard widget*
MODIFY_APPWIDGET_BIND_PERMISSIONS	signature/ system	*Internal permission allowing an application to query/set which applications can bind AppWidgets.*
CHANGE_BACKGROUND_DATA_SETTING	signature	Allows applications to change the background data setting. Not for use by third-party applications. pending API council
GLOBAL_SEARCH	signature/ system	*This permission can be used on content providers to allow the global search system to access their data. Typically it used when the provider has some permissions protecting it (which global search would not be expected to hold), and added as a read-only permission to the path in the provider where global search queries are performed. This permission can not be held by regular applications; it is used by applications to protect themselves from everyone else besides global search.*
GLOBAL_SEARCH_CONTROL	signature	Internal permission protecting access to the global search system: ensures that only the system can access the provider to perform queries (since this otherwise provides unrestricted access to a variety of content providers), and to write the search statistics (to keep applications from gaming the source ranking).
READ_SEARCH_INDEXABLES	signature/ system	*Internal permission to allows an application to read indexable data.*
SET_WALLPAPER_COMPONENT	signature/ system	*Allows applications to set a live wallpaper.*
READ_DREAM_STATE	signature/ system	*Allows applications to read dream settings and dream state.*
WRITE_DREAM_STATE	signature/ system	*Allows applications to write dream settings, and start or stop dreaming.*
ACCESS_CACHE_FILESYSTEM	signature/ system	*Allow an application to read and write the cache partition.*
COPY_PROTECTED_DATA	signature	Must be required by default container service so that only the system can bind to it and use it to copy protected data to secure containers or files accessible to the system.
CRYPT_KEEPER	signature/ system	*Internal permission protecting access to the encryption methods*
READ_NETWORK_USAGE_HISTORY	signature/ system	*Allows an application to read historical network usage for specific networks and applications.*

MANAGE_NETWORK_POLICY	signature	Allows an application to manage network policies (such as warning and disable limits) and to define application-specific rules.
MODIFY_NETWORK_ACCOUNTING	signature/ system	*Allows an application to account its network traffic against other UIDs. Used by system services like download manager and media server. Not for use by third party apps.*
C2D_MESSAGE	signature	C2DM permission. Used internally.
PACKAGE_VERIFICATION_AGENT	signature/ system	*Package verifier needs to have this permission before the PackageManager will trust it to verify packages.*
BIND_PACKAGE_VERIFIER	signature	Must be required by package verifier receiver, to ensure that only the system can interact with it.
SERIAL_PORT	signature/ system	*Allows applications to access serial ports via the SerialManager.*
ACCESS_CONTENT_PROVIDERS_EXTERNALLY	signature	Allows the holder to access content providers from outside an ApplicationThread. This permission is enforced by the ActivityManagerService on the corresponding APIs, in particular ActivityManagerServicegetContent-ProviderExternal(String) and ActivityManagerServiceremoveContentProviderExternal(String).
UPDATE_LOCK	signature OrSystem	_Allows an application to hold an UpdateLock, recommending that a headless OTA reboot *not* occur while the lock is held._
ACCESS_NOTIFICATIONS	signature/ system	*Allows an application to read the current set of notifications, including any metadata and intents attached.*
ACCESS_KEYGUARD_SECURE_STORAGE	signature	Allows access to keyguard secure storage. Only allowed for system processes.
CONTROL_KEYGUARD	signature	Allows an application to control keyguard. Only allowed for system processes.
TRUST_LISTENER	signature	Allows an application to listen to trust changes. Only allowed for system processes.
PROVIDE_TRUST_AGENT	signature OrSystem	*Allows an application to provide a trust agent. For security reasons, this is a platform-only permission.*
LAUNCH_TRUST_AGENT_SETTINGS	signature OrSystem	Allows an application to launch the trust agent settings activity.
BIND_TRUST_AGENT	signature	*Must be required by an android.service.trust.TrustAgentService, to ensure that only the system can bind to it.*
BIND_NOTIFICATION_LISTENER_SERVICE	signature	Must be required by an android.service.notification.NotificationListenerService, to ensure that only the system can bind to it.
BIND_CONDITION_PROVIDER_SERVICE	signature	*Must be required by a android.service.notification.ConditionProviderService, to ensure that only the system can bind to it.*
BIND_DREAM_SERVICE	signature	Must be required by an android.service.dreams.DreamService, to ensure that only the system can bind to it.

INVOKE_CARRIER_SETUP	signature/ system	*Allows an application to call into a carrier setup flow. It is up to the carrier setup application to enforce that this permission is required This is not a third-party API (intended for OEMs and system apps).*
ACCESS_NETWORK_CONDITIONS	signature/ system	*Allows an application to listen for network condition observations. This is not a third-party API (intended for system apps).*
ACCESS_DRM_CERTIFICATES	signature/ system	*Allows an application to provision and access DRM certificates This is not a third-party API (intended for system apps).*
MANAGE_MEDIA_PROJECTION	signature	*Allows an application to manage media projection sessions. This is not a third-party API (intended for system apps).*
READ_INSTALL_SESSIONS		*Allows an application to read install sessions This is not a third-party API (intended for system apps).*
REMOVE_DRM_CERTIFICATES	signature/ system	*Allows an application to remove DRM certificates This is not a third-party API (intended for system apps).*
BIND_CARRIER_MESSAGING_SERVICE	signature/ system	Must be required by a android.service.carrier.CarrierMessagingService. Any service that filters for this intent must be a carrier privileged app.

Appendix B

Microsoft Reaction Card Method

Step 1: Read over the following list of words. Considering the product you have just used, tick those words that best describe your experience with it. You can choose as many words as you wish.

☐ Understandable	☐ Trustworthy	☐ Overwhelming
☐ Awkward	☐ Usable	☐ Comprehensive
☐ Approachable	☐ Fresh	☐ Secure
☐ Motivating	☐ Contradictory	☐ Ineffective
☐ Faulty	☐ Reliable	☐ Meaningful
☐ Time-saving	☐ Powerful	☐ Advanced
☐ Insecure	☐ Innovative	☐ Consistent
☐ Dated	☐ Too technical	☐ Cutting edge
☐ Credible	☐ Responsive	☐ Old
☐ Easy to use	☐ Compelling	☐ Slow
☐ Controllable	☐ Busy	☐ Effortless
☐ Effective	☐ Entertaining	☐ Creative
☐ Patronising	☐ Hard to Use	☐ Annoying
☐ Expected	☐ Ambiguous	☐ Stressful
☐ Friendly	☐ Clear	☐ Fun
☐ Flexible	☐ Cluttered	☐ Intuitive
☐ Inconsistent	☐ Empowering	☐ Sophisticated
☐ Ordinary	☐ Boring	☐ Fast
☐ Confusing	☐ Difficult	☐ Simple
☐ System-oriented	☐ Familiar	☐ Business-like
☐ Time-consuming	☐ Appealing	☐ Energetic
☐ Efficient	☐ Complex	☐ Desirable
☐ Exciting	☐ Convenient	☐ Incomprehensible
☐ Stimulating	☐ Straightforward	☐ Intimidating
☐ Unattractive	☐ Frustrating	☐ New
☐ Rigid	☐ Useful	☐ Irrelevant
☐ Poor quality	☐ Misleading	☐ Satisfying
☐ Bright	☐ Predictable	☐ Counter-intuitive
☐ Impressive	☐ Stable	☐ Simplistic
☐ Unconventional	☐ Professional	☐ Unrefined
☐ Dull	☐ Non-standard	☐ Vague
☐ Engaging	☐ Inadequate	☐ Illogical
☐ Obscure	☐ Unpredictable	☐ Organised
☐ Clean	☐ Attractive	☐ Distracting
☐ Relevant	☐ Accessible	☐ High quality

Step 2: Now look at the words you have ticked. Circle five of these words that you think are most descriptive of the product.

Appendix C

System Usability Scale

Please read the following questions and answer as honestly as possible, always keep in mind that it is not you who is being tested but the system that you used.

I think that I would like to use this system frequently.

Strongly Disagree				Strongly Agree
1	2	3	4	5
●	●	●	●	●

I found the system unnecessarily complex.

Strongly Disagree				Strongly Agree
1	2	3	4	5
●	●	●	●	●

I thought the system was easy to use.

Strongly Disagree				Strongly Agree
1	2	3	4	5
●	●	●	●	●

I think that I would need the support of a technical person to be able to use this system.

Strongly Disagree				Strongly Agree
1	2	3	4	5
●	●	●	●	●

I found the various functions in this system were well integrated.

Strongly Disagree				Strongly Agree
1	2	3	4	5
●	●	●	●	●

I thought there was too much inconsistency in this system.

Strongly Disagree				Strongly Agree
1	2	3	4	5
●	●	●	●	●

I would imagine that most people would learn to use this system very quickly.

Strongly Disagree				Strongly Agree
1	2	3	4	5
●	●	●	●	●

I found the system very cumbersome to use.

Strongly Disagree				Strongly Agree
1	2	3	4	5
●	●	●	●	●

I felt very confident using the system.

Strongly Disagree				Strongly Agree
1	2	3	4	5
●	●	●	●	●

I needed to learn a lot of things before I could get going with this system.

Strongly Disagree				Strongly Agree
1	2	3	4	5
●	●	●	●	●

If you could change anything in the system what would you change?

Bibliography

[1] Digitimes Research. Global smartphone shipments in 2015 to reach 1.401 billion units, March 2015. URL http://www.digitimes.com/news/a20150319PD213.html. Retrieved: 14-04-2015.

[2] I.D. Corporation. Smartphone OS Market Share, Q1 2015, May 2015. URL http://www.idc.com/prodserv/smartphone-os-market-share.jsp. Retrieved: 07-05-2015.

[3] Statista. Number of apps available in leading app stores as of May 2015. URL http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/. Retrieved: 10-06-2015.

[4] Android Authority. Google Play Store vs the Apple App Store: by the numbers (2015), April 2015. URL http://www.androidauthority.com/google-play-store-vs-the-apple-app-store-601836/. Retrieved: 07-05-2015.

[5] Yahoo Tech. Report: 1 in 5 Android Apps Is Malware, April 2015. URL https://www.yahoo.com/tech/report-one-in-five-android-apps-is-malware-117202610899.html. Retrieved: 06-05-2015.

[6] J.R. Raphael. Inside Android 4.2's Powerful New Security System, November 2012. URL http://www.computerworld.com/article/2473570/android/exclusive--inside-android-4-2-s-powerful-new-security-system.html. Retrieved: 29-07-2015.

[7] Xuxian Jiang. An Evaluation of the Application (”App”) Verification Service in Android 4.2, December 2012. URL http://www.csc.ncsu.edu/faculty/jiang/appverify/. Retrieved: 03-08-2015.

[8] Adrienne Porter Felt, Erika Chin, Steve Hanna, Dawn Song, and David Wagner. Android Permissions Demystified. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 627–638. ACM, 2011.

[9] Alexios Mylonas, Anastasia Kastania, and Dimitris Gritzalis. Delegate the Smartphone User? Security Awareness in Smartphone Platforms. *Computers & Security*, 34:47–66, 2013.

[10] Alexios Mylonas, Dimitris Gritzalis, Bill Tsoumas, and Theodore Apostolopoulos. A Qualitative Metrics Vector for the Awareness of Smartphone Security Users. In *Trust, Privacy, and Security in Digital Business*, pages 173–184. Springer, 2013.

[11] Patrick Gage Kelley, Sunny Consolvo, Lorrie Faith Cranor, Jaeyeon Jung, Norman Sadeh, and David Wetherall. A Conundrum of Permissions: Installing Applications on an Android Smartphone. In *Financial Cryptography and Data Security*, pages 68–79. Springer, 2012.

[12] Adrienne Porter Felt, Elizabeth Ha, Serge Egelman, Ariel Haney, Erika Chin, and David Wagner. Android Permissions: User Attention, Comprehension, and Behavior. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, page 3. ACM, 2012.

[13] Kevin Benton, L Jean Camp, and Vaibhav Garg. Studying the Effectiveness of Android Application Permissions Requests. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on*, pages 291–296. IEEE, 2013.

[14] Ryo Sato, Daiki Chiba, and Shigeki Goto. Detecting android malware by analyzing manifest files. *Proceedings of the Asia-Pacific Advanced Network*, 36:23–31, 2013.

[15] William Enck, Machigar Ongtang, and Patrick McDaniel. On Lightweight Mobile Phone Application Certification. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 235–245. ACM, 2009.

[16] Alexios Mylonas, Marianthi Theoharidou, and Dimitris Gritzalis. Assessing Privacy Risks in Android: A User-Centric Approach. In *Risk Assessment and Risk-Driven Testing*, pages 21–37. Springer, 2014.

[17] Wei Wang, Xing Wang, Dawei Feng, Jiqiang Liu, Zhen Han, and Xiangliang Zhang. Exploring Permission-induced Risk in Android Applications for Malicious Application Detection. 2014.

[18] Veelasha Moonsamy, Jia Rong, and Shaowu Liu. Mining Permission Patterns for Contrasting Clean and Malicious Android Applications. *Future Generation Computer Systems*, 36:122–132, 2014.

[19] Alan Westin. Privacy and Freedom. *Washington and Lee Law Review*, 25(1):166, 1968.

[20] Daniel J Solove. Conceptualizing Privacy. *California Law Review*, pages 1087–1155, 2002.

[21] Samuel D Warren and Louis D Brandeis. The Right to Privacy. *Harvard Law Review*, pages 193–220, 1890.

[22] Jaap-Henk Hoepman and Marc van Lieshout. Privacy. In E.R. Leukfeldt and W. P. Stol, editors, *Cyber Safety: An Introduction*, pages 75–87. Eleven International Publishing, The Hague, 2012.

[23] The OWASP Foundation. An In-Depth Introduction to the Android Permission Model and How to Secure Multi-Component Applications, April 2012. URL https://www.owasp.org/images/c/ca/ASDC12-An_InDepth_Introduction_to_the_Android_Permissions_Modeland_How_to_Secure_MultiComponent_Applications.pdf. Retrieved: 27-07-2015.

[24] William Enck, Machigar Ongtang, and Patrick McDaniel. Understanding Android Security. *IEEE security & privacy*, (1):50–57, 2009.

[25] Android Central. How to install Android apps from the Google Play website, May 2014. URL http://www.androidcentral.com/installing-android-apps-google-play-website. Retrieved: 20-05-2015.

[26] Android Police. Simplified Permissions UI In The Play Store Could Allow Malicious Developers To Silently Add Permissions, June 2014. URL http://www.androidpolice.com/2014/06/10/simplified-permissions-ui-in-the-play-store-could-allow-malicious-developers-to-silently-add-permissions/. Retrieved: 30-06-2015.

[27] iamtubeman. What latest changes to Play Store app means for privacy, June 2014. URL http://www.reddit.com/r/Android/comments/27n7yr/what_latest_changes_to_play_store_app_means_for/. Retrieved: 26-06-2015.

[28] Adrienne Porter Felt, Kate Greenwood, and David Wagner. The Effectiveness of Application Permissions. In *Proceedings of the 2nd USENIX conference on Web application development*, pages 7–7, 2011.

[29] Bhaskar Pratim Sarma, Ninghui Li, Chris Gates, Rahul Potharaju, Cristina Nita-Rotaru, and Ian Molloy. Android permissions: a perspective combining risks and benefits. In *Proceedings of the 17th ACM symposium on Access Control Models and Technologies*, pages 13–22. ACM, 2012.

[30] Liu Yang, Nader Boushehrinejadmoradi, Pallab Roy, Vinod Ganapathy, and Liviu Iftode. Short Paper: Enhancing Users' Comprehension of Android Permissions. In *Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices*, pages 21–26. ACM, 2012.

[31] John Brooke. SUS-A Quick and Dirty Usability Scale. *Usability evaluation in industry*, 189(194):4–7, 1996.

[32] Luigi Vignere, Jaideep Chandrashekar, Ioannis Pefkianakis, and Olivier Heen. Taming the Android AppStore: Lightweight Characterization of Android Applications. Technical report, EURECOM, April 2015.

[33] Jagdish Achara, Mathieu Cunche, Vincent Roca, and Aurélien Francillon. WifiLeaks: Underestimated Privacy Implications of the ACCESS_WIFI_STATE Android Permission. 2014.

[34] Android Central. Using App Permissions in Android M, June 2015. URL http://www.androidcentral.com/using-app-permissions-android-m?utm_source=related&utm_medium=module&utm_campaign=next. Retrieved: 23-07-2015.

[35] Android Police. Android M Will Never Ask Users For Permission To Use The Internet, And That's Probably Okay, June 2015. URL http://www.androidpolice.com/2015/06/06/android-m-will-never-ask-users-for-permission-to-use-the-internet-and-thats-probably-okay/. Retrieved: 13-07-2015.

[36] Android M Developer Preview - Permissions. URL https://developer.android.com/preview/features/runtime-permissions.html. Retrieved: 13-07-2015.