Analyzing Privacy of Android Apps

BY

GABRIELE PETRONELLA
B.S., Politecnico di Milano, Milan, Italy, July 2011

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2014

Chicago, Illinois

Defense Committee:
Lenore Zuck, Chair of Advisor
Robert H. Sloan
Stefano Zanero, Politecnico di Milano

To my family

ACKNOWLEDGEMENTS

I want to thank Prof. Lenore D. Zuck for all the support and time she dedicated to me, helping me through this whole thesis work. Then, I would like to thank want to tank Prof. Robert H. Sloan for the time he spent helping me in refining the scope of this thesis work. Another thanks goes to Prof. Stefano Zanero for assisting me with his precious advice.

I finally want to thank all the people who shared with me this wonderful study and life experience in Chicago. I learned from this people more that I could ever possibly learn from books. Thank you for you awesomeness.

GP

TABLE OF CONTENTS

CHAPTER			PAGE
1	**INTRODUCTION**		1
	1.1	Background and Motivations	1
	1.2	Privacy awareness context	2
	1.3	Scope and goals of this thesis	6
	1.4	Steps towards privacy awareness	6
	1.5	Scope of this thesis	8
	1.6	Android OS Permission Model	9
	1.7	Related Work	10
	1.8	Thesis Organization	11
2	**MANUAL ANALYSIS**		12
	2.1	Overview	12
	2.2	Manual analysis	12
	2.2.1	Permissions of Interest	12
	2.3	Relationship with privacy policy	19
	2.3.1	Example: Rovio's Privacy Policy	20
	2.3.2	Example: Halfbrick's Privacy Policy	22
	2.3.3	Lookup table example	22
3	**AUTOMATED ANALYSIS**		23
	3.1	Automated analysis	23
	3.1.1	Privacy policy retrieval	23
	3.1.2	Permissions retrieval	24
	3.1.3	Privacy policy analysis	24
	3.1.4	User's interface	25
	3.2	Example	25
4	**IMPLEMENTATION**		28
	4.1	Permissions collection	28
	4.2	Privacy Policy collection	29
	4.3	Semantic analysis	31
	4.4	Results	33
5	**EXPERIMENTAL RESULTS**		35
	5.1	Quantitative results	35
	5.1.1	Metric definition	35
	5.2	Qualitative results	37

TABLE OF CONTENTS (Continued)

CHAPTER **PAGE**

5.3 Case study: Shopkick 37
5.3.1 RECORD 40
5.3.2 ACCESS_FINE_LOCATION 41
5.3.3 CAMERA 41

6 CONCLUSIONS AND FUTURE WORK 43
6.1 Conclusions 43
6.2 Future Work 43
6.2.1 Live monitoring of behaviors 44
6.2.2 Crowdsourced refinement of lookup tables 44
6.2.3 Improved NLP 45

CITED LITERATURE 46

VITA 48

LIST OF TABLES

TABLE		PAGE
I	TOP 20 REQUESTED PERMISSIONS IN FREE APPS	13
II	TOP PRIVACY-RELATED PERMISSIONS IN FREE APPS	19
III	ACCESS_COARSE_LOCATION LOOKUP TABLE	22
IV	PERMISSION IMPACT SCORES	36
V	SHOPKICK APP PERMISSIONS	38

LIST OF FIGURES

FIGURE		PAGE
1	Relationships between permissions, actions and behaviors	4
2	Privacy awareness steps	8
3	Search interface	26
4	Permission display interface	27
5	Privacy Policy link in the Play Store web interface	30
6	Detail of Rovio's Privacy Policy structure	32
7	Quantitative results (over 4300 applications, December 7, 2013) .	37

SUMMARY

In this thesis we present the design and the implementation of a tool to analyze privacy policies of Android applications, with the purpose of increasing the user's awareness about privacy-related concerns. The goal of this work is to produces a tool, targeted to users who wish to evaluate the compliance of arbitrary Android applications to their own privacy policies. The tool implements a semantic analyzer of privacy policies, able to extract relevant sentences from them and put them in relationship with the corresponding privacy-related permissions requested by applications. This work was inspired by the manual review of privacy policies of Android applications, and noticing how a common informal structure was evident across multiple documents.

CHAPTER 1

INTRODUCTION

1.1 Background and Motivations

During the last few years, mobile applications constantly growing in both number and importance in our everyday life.

Such impressive growth is marking a technology revolution, and, as many revolutions, it carries huge consequences affecting everyone's life. Some of this consequences lead to clear improvements, whereas some others put under spotlight some concerns that were not that relevant just a few years ago.

The increase in penetration and capabilities of mobile devices' has turned them in something most people would find hard to separate from, a sort of extension of their own body. Mobile devices nowadays typically hold a huge amount of information about their owner: email, contacts, bank accounts, social network profiles, location information.

How and under which circumstances such information can be disclosed has become a great concern very quickly, turning privacy related issues in notable examples of the aforementioned worrying consequences.

1.2 Privacy awareness context

We now define the scope in which this thesis work collocate itself, going through the main factor affecting privacy in mobile applications and describing the existing relationships between them.

We identify three main factors to take into consideration:

- permissions
- actions and behaviors
- privacy policies

Permissions determine which data or services the app can access on the user's device, so they effectively define the maximum potential impact an application can have over the user's privacy: the fewer the permissions, the lower the risk. However, a recent study[1] showed how, given only the `INTERNET` permission, an Android application was capable of stealing online account login credentials. This highlights how permissions only represent an upper bound: even apps requesting one single permission can significantly affect the privacy of user.

We define *actions* as the minimum unit of work an application can do. Actions can be divided in two main categories

- actions that cannot be performed without an explicit permission
- actions that do not require an explicit permission to be performed (e.g., impact local state of app)

The former category typically include the actions having any impact on the device's security. Such actions are forbidden by the OS (*Operating System*) by default, and are allowed only if specific permissions have been granted to the application. The latter usually represent actions not affecting the device's security, e.g. actions confined within the bound of the internal application's logic.

We then define *behaviors* as sequences of one or more actions; such definition implies that some behaviors, namely those including actions from the first category, can occur only when specific permissions have been granted.

Example 1. Let us a consider a game application that stores user's top scores and sends them over the Internet to a remote server. We can break this app down into the following actions:

- `save_user's_top_scores` (A_1)
- `send_top_scores_over_the_internet` (A_2)

The sequence of A_1 and A_2 forms the behavior `store_and_send_user's_top_score_to_a_remote_sever` (B_1). A_1 requires the permission `INTERNET` to be granted, whereas A_2 can always be performed. This implies that B_1 can occur only if permission `INTERNET` is granted.

So we have seen how permissions enable actions and how actions can be composed to form behaviors. It is important to notice the cardinality of this relationships:

- one permission enables one or more actions
- one behavior is enabled by one or more actions

Hence, it becomes evident how a many-to-many relationship between permissions and behaviors exist, i.e. enabling one or more permissions can potentially enable one or more behaviors.

While some of this behaviors are expected, and even desirable, some others might result unexpected and potentially undesirable.

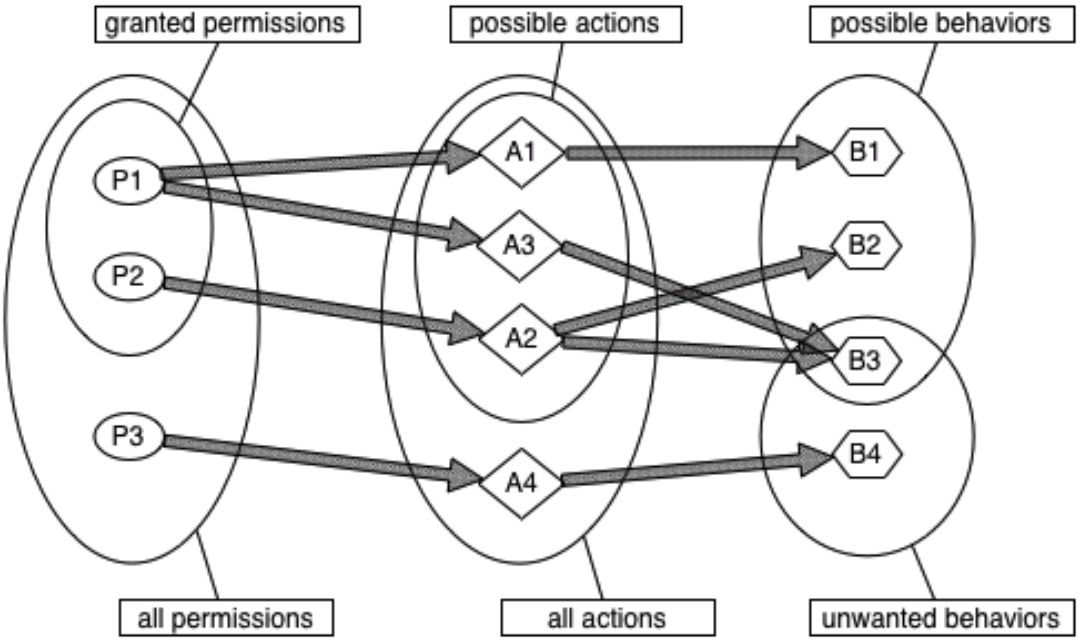

Figure 1: Relationships between permissions, actions and behaviors

Figure 1 highlights this possible scenario: permission P_1 is required to enable action A_1 which in turn enables behavior $B1$. Similarly, permission P_2 enables action A_2 and

consequently behavior B_2. Granting permissions P_1, however, also enables action A_3, and when combined with action A_2, an unwanted behavior B_3 may occur.

Here is a practical instance of this scenario:

Example 2.

- the `READ_PHONE_STATE` permission enables the action `detect_an_incoming_phone_call` (A_1), which enables the behavior `pause_the_game_when_an_incoming_phone_call_arrives` (B_1).
- the `INTERNET` permission enables the action `send_and_receive_data_over_the_ Internet` (A_2), which enables the behavior `send_the_user's_top_score_to_a_remote_server` (B_2).
- the `READ_PHONE_STATE` permission also enables the action `read_the_user's_phone_number` (A_3). The combination of actions A_2 and A_3 enables the behavior `send_the_user's_phone_number_to_a_remote_server`(B_3), which may be undesirable.

Given this potential issues with the permission-based model, it appears that something more has to be done in order to rule out unwanted behaviors.

This something that is commonly covered by privacy policies: legal documents provided together with the application. In the context we just described, privacy policies act should act as a filter on the possible behaviors, telling the final users which of the possible behaviors is the app going to actually generate.

For instance, a privacy policy may explicitly state that the user's phone number is never collected nor accessed, de facto promising the application will never perform *A3*, hence ruling out *B3*. Nonetheless, a real app may still deviate from what stated in its policy.

1.3 Scope and goals of this thesis

This thesis work restricts to Android applications, where permissions are declared up-front, as further explained in Section 1.6 when the application is installed.

This thesis describes a methodology, supported by tools, that enables a user who installs an Android app to gain better understanding of the app's capabilities based on the permission it requires and its privacy policy, and alerts the user to some of the (potentially) unintended consequences that the user grants the application by installing it.

1.4 Steps towards privacy awareness

Given the general context of privacy awareness, we now identify a set of steps that are likely to lead to an increase in users' awareness.

1. Understanding permissions

   Previous studies[2] show how permissions are rarely understood by users. Specifically users appear not be able to correlate a permission with the possible actions it enables, let alone the spectrum of possible behaviors derived from actions interleaving.

   The first step towards awareness is to analyze permissions and derive potential consequences. We are especially interested in permissions that directly affect privacy. As

an example, the `READ_PHONE_STATE` permission is typically requested by apps in order to be able to respond to phone events such as a incoming call, but it also enables the app to read the user's phone and IMEI numbers.

Once permissions have been fully analyzed, one can then identify their effect on the user's privacy.

2. Correlating permissions and privacy policies

   The next step towards privacy awareness is to map each permission the app requests into its impact as stated in its privacy policy. While privacy policies do not share a common defined structure, they do express similar concepts in similar ways, which enable to extracting useful pieces of information from them. For example, an application requesting the `ACCESS_FINE_LOCATION` permission is very likely to be associated to a privacy policy containing expressions alike to *"GPS", "Location Services", 'Global Positioning System', etc.*

   This step takes into consideration every permission that enables an app to affect the user's privacy with the final goal of building a dictionary of common expressions and patterns that associate the permission to natural language sentences in the privacy policy.

3. Correlating apps behavior and privacy policies

   The last step is to monitor the app's actual behavior. Recalling Example 2, the application might never retrieve the user's phone number even though it requested such permission.

This final step would then inform the users about the “goodness” of an application with respect the claims expressed in the privacy policy and the actual actions taken by the app once installed and running on their phone.

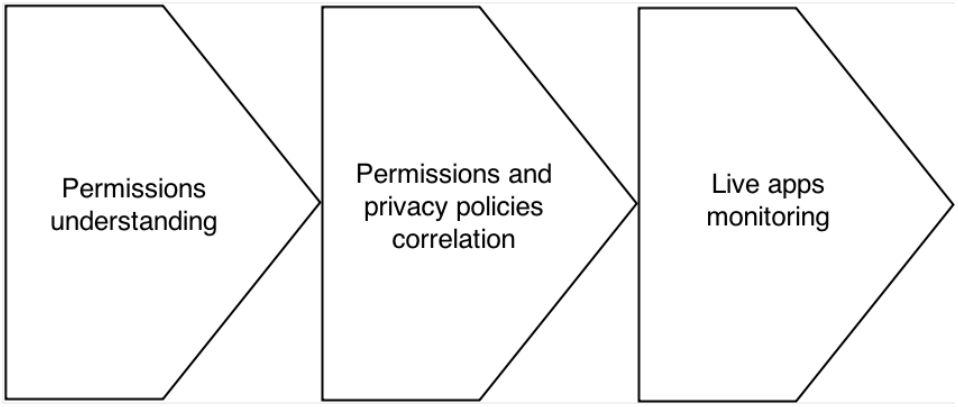

Figure 2: Privacy awareness steps

1.5 Scope of this thesis

This thesis work focuses on the first two steps discussed in the previous section.

The first step requires an in-depth analysis and comprehension of the most requested permissions, in order to identify the potential privacy concerns each one of them carries.

Once the permissions of interest have been identified we will then perform a manual analysis in order to understand how privacy policies deal with the privacy concerns represented by them. The manual analysis will enable an automated process, which, given

an arbitrary Android application published on the Play Store platform, retrieves its privacy policy and produces a human-readable report about the relationships between the permissions list and the analyzed legal document.

The final result will then allow a potential user to aggregate a large amount of privacy-related information in a quick and concise way, marking a clear step towards privacy awareness.

1.6 Android OS Permission Model

As explained in the Android Developer Guide, "Android is a privilege-separated operating system, in which each application runs with a distinct system identity. [...] Additional finer-grained security features are provided through a 'permission mechanism that enforces restrictions on the specific operations that a particular process can perform, and per-URI permissions for granting ad-hoc access to specific pieces of data. [...] A basic Android application has no permissions associated with it by default, meaning it can not do anything that would adversely impact the user experience or any data on the device."[3]. In order to access to the protected features of the device the developer has to declare a list of permissions the application needs. Such list is specified in the `AndroidManifest.xml`, a file containing application metadata, included by every Android application.

For example, an application that needs to send and receive data over the Internet would specify an `AndroidManifest` as the one in Listing 1.1.

"At application install time, permissions requested by the application are granted to it by the package installer, based on checks against the signatures of the applications

Listing 1.1: Example of permission declaration in AndroidManifext.xml

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

declaring those permissions and/or interaction with the user. No checks with the user are done while an application is running: it either was granted a particular permission when installed, and can use that feature as desired, or the permission was not granted and any attempt to use the feature will fail without prompting the user."[3]

1.7 Related Work

As discussed in Section 1.6, permissions are granted at install time, meaning that a user is supposed to have reviewed the permissions the application requested and to have deemed them acceptable, before granting them altogether.

Such mechanism has been criticized for several reasons: first of all, recent studies[2][4] show how users might not have complete understanding of the meaning and consequences of each permission in the list. The same studies also show how even experienced users are found not pay attention to the permission list, most likely due to its verbosity and length. To further prove this last observation, in recent experiment[1] an ad-hoc application was developed and put on the Play Store; the application requested all possible permissions, enabling the researchers to steal personal data from the user, such as email address and

phone numbers. The application received 1300 downloads over a 3-month period, without being advertised, and collected 1950 email addresses.

1.8 Thesis Organization

The following chapters of this thesis are organized as follows. Chapter 2 presents the manual analysis performed over privacy policies and permissions. Chapter 3 then describes the automatic analysis of Android apps, enabled by the results of Chapter 2. Chapter 4 presents the details of the implementation and of the tools used to support both manual and automated analysis. Chapter 5 presents a metric used for evaluating the compliance of Android applications w.r.t. their privacy policies, as well as quantitative results - measured with such metric - and qualitative results. Chapter 6 concludes this thesis, proposing possible further developments to the work done.

CHAPTER 2

MANUAL ANALYSIS

2.1 Overview

In this and the next chapters we will discuss the analysis approach in details. Two well-distinct stages of such analysis can be identified, namely a manual one and an automated one. The manual analysis involves an accurate investigation of Android permissions and their relationship with privacy policies.

The outcome of this first stage is a formal representation of this relationship, which enables the second - automated - phase of the analysis, thoroughly discussed in Chapter 3.

2.2 Manual analysis

In this section we present the manual steps that needed to be executed in order to enable an automated analysis.

2.2.1 Permissions of Interest

The first step in our research is to identify which of the permissions an app can request have privacy related consequences.

Firstly we are interested into discovering which are the most requested permissions in our domain of interest. There are no official data released by Google, however we were able to retrieve empirical data with the use of unofficial APIs[5], discussed in greater details in Chapter 4.

The Play Store platform divides apps in categories (such as *Games*, *Education*, *Tools* and so on). We run an analysis on the top downloaded free apps for each categories, identifying a total of 4300 applications; we then retrieved the permission list for each one of them and aggregated such data. The top 20 requested permissions are shown in Table I.

TABLE I: TOP 20 REQUESTED PERMISSIONS IN FREE APPS

#	Permission	% apps using it
1	INTERNET	99.35%
2	ACCESS_NETWORK_STATE	98.35%
3	READ_EXTERNAL_STORAGE	92.35%
4	WRITE_EXTERNAL_STORAGE	92.35%
5	ACCESS_WIFI_STATE	85.47%
6	READ_PHONE_STATE	78.39%
7	WAKE_LOCK	59.65%
8	VIBRATE	32.79%
9	GET_ACCOUNTS	32.79%
10	ACCESS_COARSE_LOCATION	19.86%
11	GET_TASKS	14.86%
12	RECEIVE_BOOT_COMPLETED	13.88%
13	ACCESS_FINE_LOCATION	9.93%
14	READ_LOGS	9.88%
15	MOUNT_UNMOUNT_FILESYSTEMS	6.93%
16	RECORD_AUDIO	5.95%
17	CHANGE_WIFI_STATE	4.98%
18	DISABLE_KEYGUARD	4.95%
19	READ_CONTACTS	3.00%
20	WRITE_SETTINGS	2.98%

Generated from 4300 apps on Nov 17, 2013

The general list of permissions, however, includes permissions with no significant impact with respect to privacy matters. The next step in our research is then to identify which permissions affect the user's privacy and how.

What follows is a list of all the permissions which enable actions raising privacy concerns. For each permission a list of enabled actions is provided along with a discussion about the privacy concerns.

INTERNET

Actions enabled

- send data over the Internet
- receive data over the Internet

Privacy concerns This permission is the most requested and also the most dangerous, privacy-wise, as it enables the communication with remote servers over the Internet. Used in combination with other permissions it allows the application to send any retrieved data to an arbitrary remote server.

Examples An application can send any sensitive data retrieved thanks to other permissions over the Internet. For instance on can think of an application reading the user's phone number and sending it to a remote server, perhaps with the purpose of targeted phone advertisement. As described in a recent study[1], this permission can be dangerous by itself.

READ_EXTERNAL_STORAGE

Actions enabled

- read files on the external SD card

Privacy concerns This permission allows to read files on an external SD card, so that anything stored in the external memory can be accessed by the app, including pictures, videos and data stored by other applications.

Examples An application can retrieve all the user's picture stored on the SD card and send them to a remote server, violating the user's privacy.

WRITE_EXTERNAL_STORAGE

Actions enabled

- write files on the external SD card
- read files on the external SD card

Privacy concerns Despite the name, this permission implicitly enables also `READ_EXTERNAL_STORAGE`, so the same privacy concerns apply.

ACCESS_WIFI_STATE

Actions enabled

- access the `WifiManager`

Privacy concerns Accessing the `WifiManager` allows the app to read information about the WiFi network the device is connected to, including the current IP address.

Examples One can think of an application tracking the user's location by estimating the WiFi network's location, as recent studies demonstrate[6]

READ_PHONE_STATE

Actions enabled

- detect in-progress phone calls
- read IMEI and IMSI identifiers
- read the network provider information
- read the user's phone number

Privacy concerns This is one of the most controversial permissions. While most applications request this permission in order to detect incoming phone calls, which is usually a legitimate use, it can also be used to retrieve sensitive pieces of information such as the phone number.

Examples An application can steal the user's phone number and sell it to advertisement companies for profit.

GET_ACCOUNTS

Actions enabled

- read the list of accounts from the Accounts Service

Privacy concerns The list of accounts consists of a list of usernames for each account associated with the device. For instance, the application might retrieve the email address associated with the user's GMail account.

Examples Retrieving account's usernames can be the first step towards identity stealing. A malicious application can use this piece of information to break into a user's email account and access personal data.

ACCESS_COARSE_LOCATION

Actions enabled

- know the (coarse) device location

Privacy concerns The coarse location is determined by the triangulation of GSM tower cells information and WiFi information. Although coarse the location can determine a user's location with a good level of accuracy, therefore representing a privacy concern.

GET_TASKS

Actions enabled

- know which tasks are running or recently run

Privacy concerns Allows an application to get information about the currently or recently running tasks. While not dangerous by itself, it can help in stealing information when combined with other permissions.

ACCESS_FINE_LOCATION

Actions enabled

- know the (fine) device location

Privacy concerns The same privacy concerns as coarse location apply.

READ_LOGS

Actions enabled

- read the low-level system log files

Privacy concerns Not particularly worrying by itself, but it enables the app to read everything other applications might have logged. If some application logs sensible data, this permission will allow them to be read.

RECORD_AUDIO

Actions enabled

- record audio

Privacy concerns While this permission has legitimate uses such as note taking apps or voice search apps, it is a potential tool for eavesdropping and recording of sensible information.

READ_CONTACTS

Actions enabled

- read the user's contacts data.

Privacy concerns The whole user's address book can be read.

Table Table II summarizes the privacy-related permissions we will consider in our analysis.

TABLE II: TOP PRIVACY-RELATED PERMISSIONS IN FREE APPS

#	Permission	% apps using it
1	INTERNET	99.35%
2	READ_EXTERNAL_STORAGE	92.35%
3	WRITE_EXTERNAL_STORAGE	92.35%
4	ACCESS_WIFI_STATE	85.47%
5	READ_PHONE_STATE	78.39%
6	GET_ACCOUNTS	32.79%
7	ACCESS_COARSE_LOCATION	19.86%
8	GET_TASKS	14.86%
9	ACCESS_FINE_LOCATION	9.93%
10	READ_LOGS	9.88%
11	RECORD_AUDIO	5.95%
12	READ_CONTACTS	3.00%

Generated from 4300 apps on Nov 17, 2013

2.3 Relationship with privacy policy

Now that we identified the permissions we are interested in, we want to see how each permission relates to the privacy policy, i.e. in which terms the privacy policy deals with permissions the app requested.

Our analysis involved an initial corpus of twenty policies. For each permission we went through each privacy policy of the corpus, manually extracting common pattern and terms.

The result of this manual investigation is a lookup table associating each permission with a list of common words or expressions used in the privacy policies to refer to the actions enabled by it.

2.3.1 Example: Rovio's Privacy Policy

We now illustrate what expressed in the previous section, taking a popular app's privacy policy as an example. The application in question is *Angry Birds* by Rovio Entertainment Ltd. If we take the `ACCESS_COARSE_LOCATION` permission into consideration we can find several parts of the privacy policy referring to it. Once such parts have been identified, the relevant words and expressions concerning the specific matter are then extracted. What follow are the relevant sections of Rovio's privacy policy concerning the user's location matter[7]:

> *"Rovio or third parties operating the ad serving technology may use demographic and **geo-location** information (for more information regarding use of Location Data see below Section 3) as well as information logged from your hardware or device to ensure that relevant advertising is presented within the Service."*
>
> *"To the extent Rovio makes **location** enabled Services available and you use such Services, Rovio may collect and process your **location** data to provide **location** related Services and advertisements."*
>
> *"The **location** data is processed and stored only for the duration that is required for the provision of the **location** related Services."*
>
> *"Rovio may use, depending on the service (1)IP-based **location** based on the IP address presented by the end-user, (2) fine **geo-location** data based on coordi-*

*nates obtained from a mobile device's **GPS** radio, or (3) coarse, network-based **geo-location** data based on proximity of network towers or the **location** of WiFi networks."*

*"Your fine, **GPS-based geo-location** is not accessed without your consent."*

"Notwithstanding Rovio's partners who are providing location related parts of the Service, Rovio will not share your GPS geo-location with third parties without your consent."

"To the extent Rovio makes available GPS geo-location to third parties in accordance with this Privacy Policy, it will be provided anonymously."

"This includes, for example, collection of IP-based geolocation data to ensure that the product, service or features served comply with applicable laws of that nation."

All of the above sentences are relevant to the matter of establishing what is the app expected behavior with respect to the user's location information. It is particularly interesting to observe a few characteristics of some of the cited sentences. Specifically Rovio's privacy policy states

*"Your fine, **GPS-based geo-location** is not accessed without your consent."*

This provides a false sense of assurance, since in Android application the consent has already been given at installation time, so the geo-location can always be accessed by the application without further notice to the user.

2.3.2 Example: Halfbrick's Privacy Policy

Similar examples can be found in many popular apps. Let us for instance consider the case of Halfbrick, a company most known for a game called Fruit Ninja. It the app's privacy policy one can read

"Where you allow us access to such information, we may also collect information from your device such as your geographic location" [8]

Again, we can see how similar matters are mentioned similarly in different privacy policies, and also how again such sentence provides false assurance: Android apps always have permissions granted upfront, so the phrase *"Whenever you allow us"*, realistically means *"Whenever the app is installed"* on an Android device.

2.3.3 Lookup table example

Based on this manual analysis we built a look up table of permissions and the way they are referred to in policies. An sample entry of this table is shown in Table III.

TABLE III: ACCESS_COARSE_LOCATION LOOKUP TABLE

Permission	Keywords
`ACCESS_COARSE_LOCATION`	*"gps", "IP based location", "location", "location services", "geo-location", "geographic location"*

CHAPTER 3

AUTOMATED ANALYSIS

3.1 Automated analysis

Once the lookup table has been constructed is then possible to proceed with an automated analysis of applications. In this section we present the high level steps taken by the analysis, along with the challenges face during this process, while the implementation details will be discussed in Chapter 4.

3.1.1 Privacy policy retrieval

The first step is to retrieve the privacy policy given an arbitrary application. This proved to be one significant challenge, for the following main reasons:

- there is no legal requirement enforcing an Android application to have a privacy policy. Some application simply don't provide one.
- there is no standard format for such documents. While the Play Store has a standard interface for providing a link to an app's privacy policy, the content of the link itself is arbitrary and completely at the discretion of the developer.
- some application developers do not provide a link to the privacy policy in the Play Store.

For the purposes of this thesis work, we limit our search for the privacy policy to the Play Store web interface. If the developer provided a privacy policy link, it is followed

through and analyzed, otherwise the search fails and we assume the application not to have a privacy policy at all. This may not hold true and it is a known downside of this approach, which can be object of further improvement in future development of this analysis.

When present, the *Privacy Policy* link in the Play Store interface can be followed to access the actual document, which can then be fetched in order to perform semantic analysis over it, as discussed in Section 3.1.3.

3.1.2 Permissions retrieval

The next step of the analysis involves retrieving the permissions list given an arbitrary Android application. This is a much easier task than retrieving the privacy policy, since every Android application is guaranteed to declared the requested permissions in an uniform format. The main challenge of this step regards the implementation, due to the lack of official APIs to retrieve such piece of information. The details about how this issue was overcome are provided in Chapter 4.

3.1.3 Privacy policy analysis

Once the privacy policy document and the permissions list are both available, we can then proceed with the semantic analysis.

Firstly, the document is broken down into semantic sections, such as paragraphs and sentences. Secondly the keyword and expressions contained in the lookup table of each permission are matched against each section. For each permission the relevant matching sentences are then collected.

Such result is then presented to the final user, as discussed in the next section.

3.1.4 User's interface

The automated analysis is made available to the user via a web interface. The interface allows the user to search for an arbitrary application on the Play Store; the permissions list and the privacy policy are then automatically retrieved, whenever possible. The user has then the ability of selecting a specific permission, and a list of relevant sentences will be extracted by the privacy policy and presented to them, along with an accurate description of the permission itself.

Such interface allows the user to quickly evaluate the privacy-related risks of an Android application, by highlighting the relevant sections of the privacy policy and by providing useful information about sensible permissions.

3.2 Example

We now present an example in order to better summarize the steps discussed in the previous section. Figure 3 shows the search interface: in the example we are searching for the game *Angry Birds* and, as we digit, a list of suggestions is dynamically computed by live-querying the Play Store and presented to the user.

Once the application has been selected from the list, the permissions list and the privacy policy are automatically retrieved and displayed. The user can then select one of the permissions requested by the application in order to see all the relevant sections of the privacy policy.

In Figure 4 the user selected the `ACCESS_COARSE_LOCATION` permission and a list of relevant sentences is displayed right under the permission description.

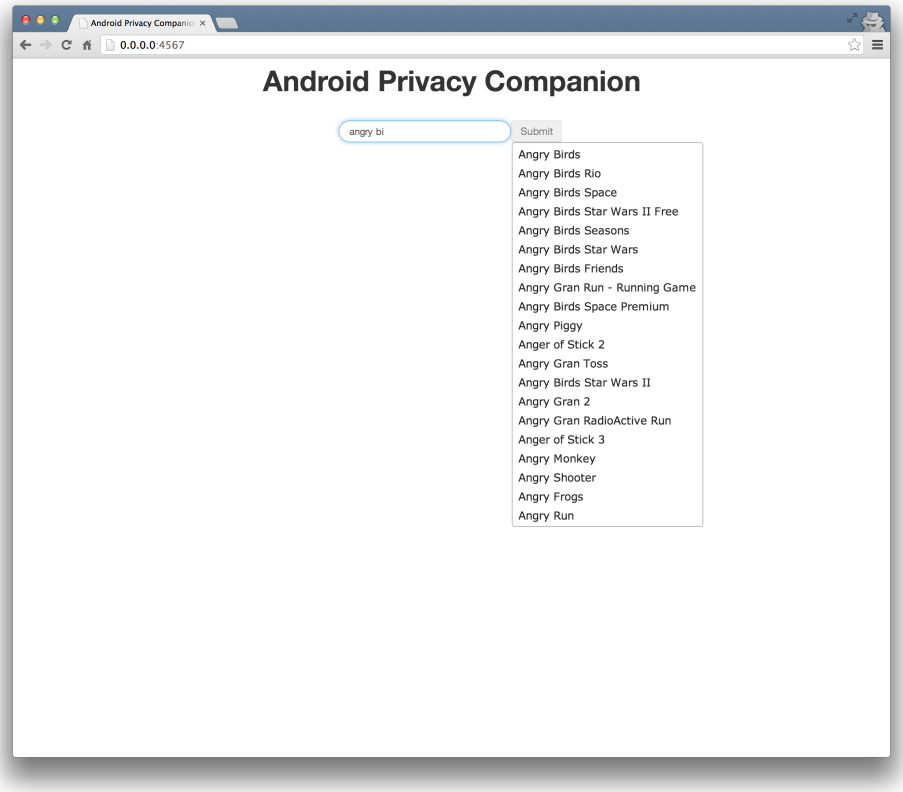

Figure 3: Search interface

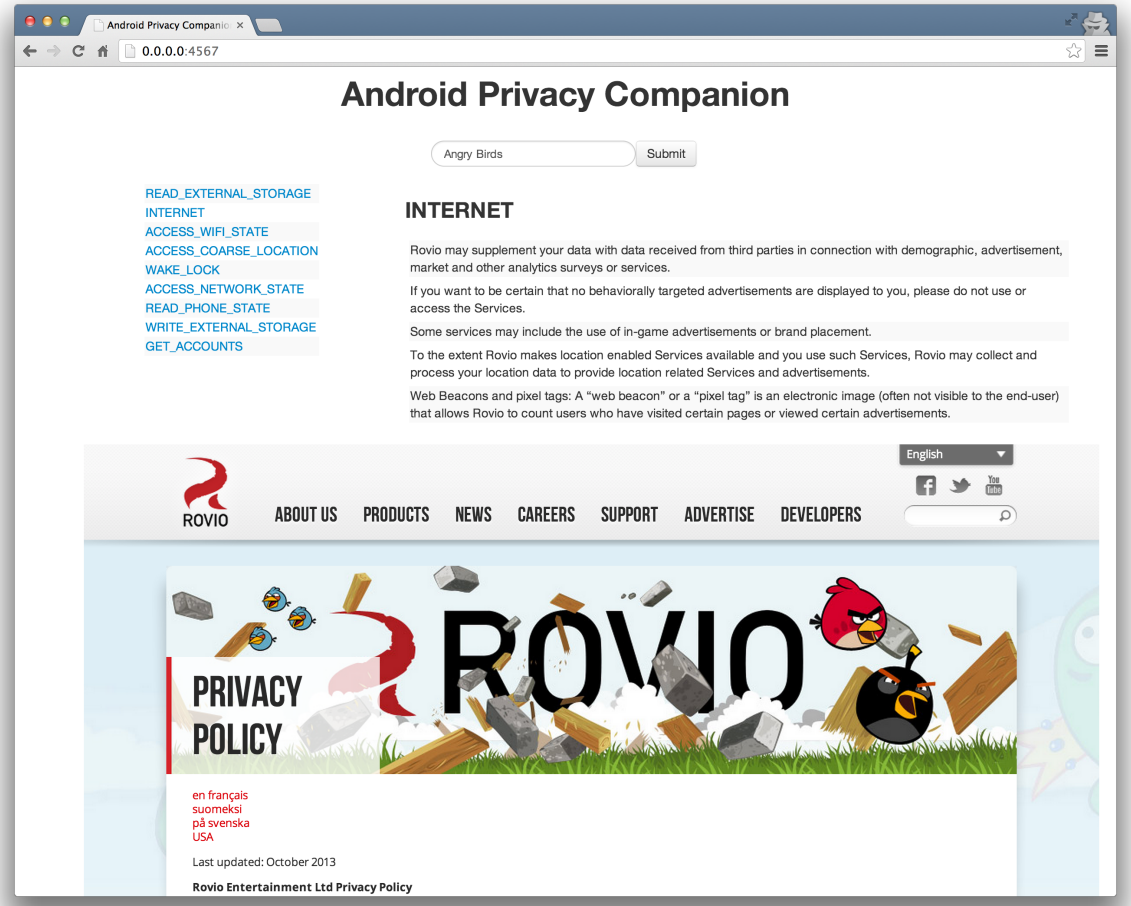

Figure 4: Permission display interface

CHAPTER 4

IMPLEMENTATION

In this chapter we present the details of the implementation. We first describe the data collection process, for privacy policies and permissions; we then discuss how the data collected were subsequently analyzed, preprocessed and selected. Finally we present the implementation of the algorithm discussed in the previous chapter.

4.1 Permissions collection

As discussed in Section 1.6, Android apps are required to declare upfront a list of permissions they need. Such list is stored in the `AndroidManifest.xml` file of each app. At installation time the user is able to review the permissions and decide whether to grant them or not.

Due to the lack of public official API for retrieving the permissions list, we first attempted to retrieve it through the Play Store web interface.

From a programmatic point of view, however, some issues arise. First of all the permissions as presented to the user are in a natural language format, whereas the permissions in the `AndroidManifest.xml` file are expressed with a canonical name. For instance the permission `READ_EXTERNAL_STORAGE` correspond to the natural language description *"modify or delete the contents of your USB storage"*. This would require an extra processing step to map the natural language description to the correspondent.

Secondly, and most importantly, the permission list is accessible from the web interface only after pressing the install button and this step is allowed only from a registered Google account with at least one Android device registered.

While this issues can be overcome, they added unexpected complexity to this step and therefore an alternative path was explored.

As mentioned above, Google does not provide an official API for retrieving applications metadata, such as the permission list, however an unofficial Python implementation exist and it is publicly available[5]. There also exist another open-source project[9], based on the unofficial API, featuring the ability of performing search queries, downloading apps and retrieving apps permissions.

Thank to the use of the unofficial API, the issues mentioned above were solved and we were able to retrieve the permissions from an arbitrary app available on the Play Store.

4.2 Privacy Policy collection

Automatically retrieving a privacy policy document for an arbitrary Android app is a much harder task than retrieving its permission list.

Whenever present, the Privacy Policy link appears in the *Additional Information* section on the Play Store web interface, as shown in Figure 5.

However, while the `AndroidManifest.xml` file is guaranteed to be present for any application on the Play Store, this does not hold true for the Privacy Policy link.

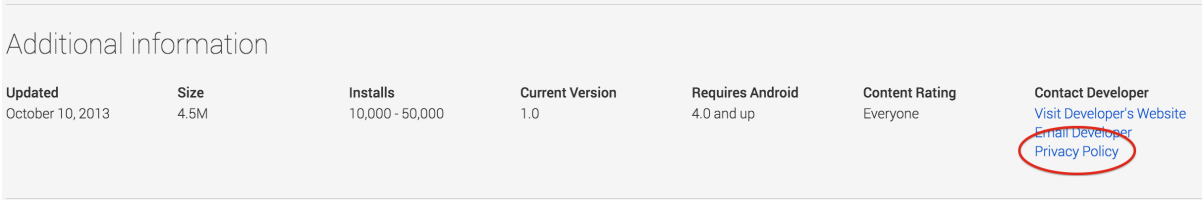

Figure 5: Privacy Policy link in the Play Store web interface

It is not in fact enforced by neither Play Store policies for an app to have a Privacy Policy at all. There is also no legal requirement (although deceptive practices are prohibited by the law).

So it two cases can occur: either the app does not have a Privacy Policy at all, or the developer has not inserted the Privacy Policy on the Play Store. In both cases the automatic retrieval of the Privacy Policy given an app is made impossible, so we will not further distinguish between them.

From the data we collected, it appears that out of the top 1093 downloaded free games apps, about the 39.79% does not have a privacy policy publicly available through the Play Store.

That being said, a Privacy Policy link is still no guarantee of the ability of retrieving a sensible Privacy Policy document. The link can point to anything the developer decides, and this leads to extremely heterogeneous paths to reach the final document of our interest.

The first issue encountered is the redirection mechanism that is very often in place. As an example, the Privacy Policy URL for *Angry Birds*, by Zynga, is:

`https://www.google.com/url?q=http://m.zynga.com/about/privacy-center/privacy-policy`

which redirects to

`http://m.zynga.com/about/privacy-center/privacy-policy`

which redirects to

`http://company.zynga.com/privacy/policy`

which contains the Privacy Policy document.

4.3 Semantic analysis

Once the document has been retrieved, it need to be semantically processed. For this purpose, we take advantage of Treat, a natural language processing framework for Ruby[10]. The Treat project aims to build a language-agnostic NLP framework for Ruby with support for tasks such as document retrieval, text chunking, segmentation and tokenization, natural language parsing, part-of-speech tagging, keyword extraction and named entity recognition.

The privacy policy document is firstly split into its logical subdivision using a SRX chunker, which implements the approach proposed in a study by Marcin Milkowski and Jaroslaw Lipski[11].

The the document is furthed split into sentences with the aid of a SRX segment, again proposed by Marcin Milkowski and Jaroslaw Lipski[11].

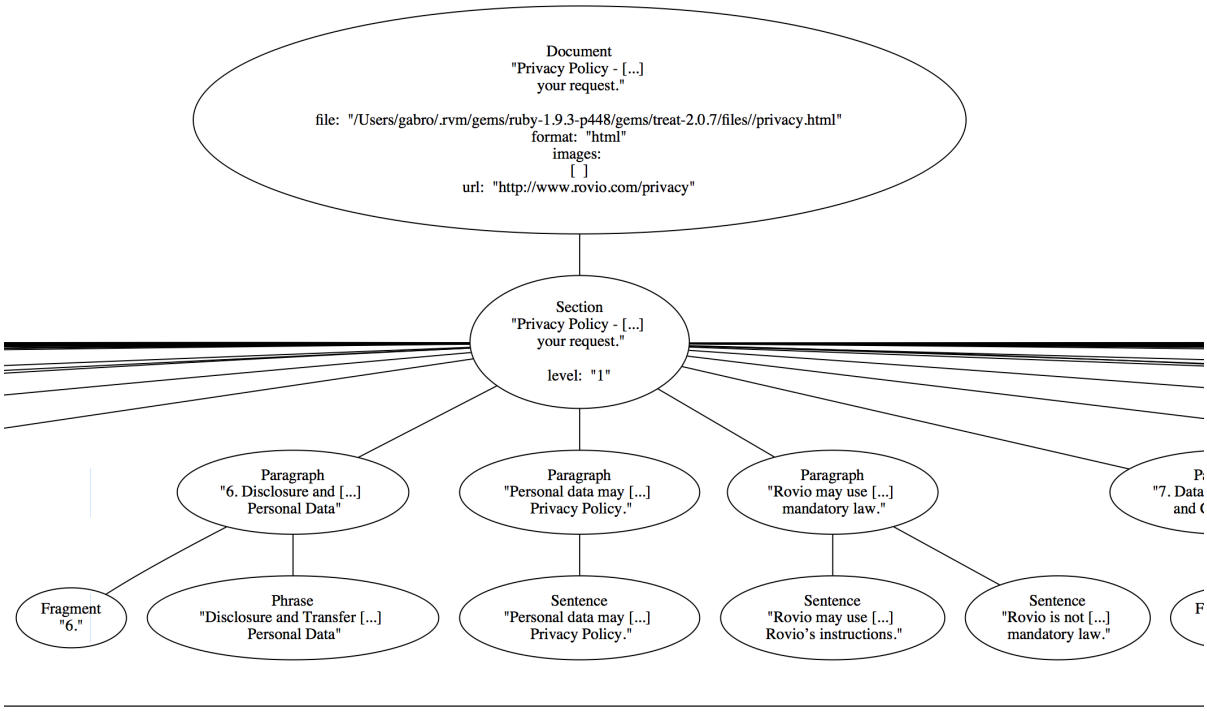

Figure 6: Detail of Rovio's Privacy Policy structure

Figure Figure 6 shows a detail of the semantic tree in which the original policy has been divided. Each internal node represents either paragraphs or sections of the document, whereas the leaf nodes are phrases and sentences.

Once sentences and phrases have been obtained, they can be searched for the expressions contained in the lookup table of each permission. For example let us take the sentence.

> *"Rovio or third parties operating the ad serving technology may use demographic and* ***geo-location*** *information (for more information regarding use of Location Data see below Section 3) as well as information logged from your hardware or device to ensure that relevant advertising is presented within the Service."* [7]

The lookup table of `ACCESS_COARSE_LOCATION` contains the word "location", hence the above sentence will be matched and will be considered relevant to such permission.

4.4 Results

Results are discussed in details in Section 5.1, however their collection brought up several technical challenges that required a rather sophisticated implementation to be dealt with. The main issue is represented by the significant number of applications we want to analyze; for each one of them we need to retrieve their privacy policy, their permission list and then analyze such information.

The challenges then become two:

- Performing thousands of simoultaneous requests to Play Store servers
- Performing thousands of simoultaneous analysis on the same machine

The first challenge derives from Google's anti-bot protection, which result in a IP-ban in case of too many requests in a short amount of time. The second challenges is instead an architectural limitation: spawning thousands of simultaneous computations easily hogs any personal computer's CPU, most likely leading to a system crash.

A naive approach to both challenges would be to serialize the operations, analyzing only one application at the time. Hoever, considering an average processing time of 10 seconds per application, analyzing thousands of applications would require several hours of computation and such architeture wouldn't scale within reasonable bounds in case of an increased number of applications (e.g. if one would like to analyze a significant fraction of the Play Store).

What we want is then a fixed amount of computations running concurrently, in order to achieve a fast computation without hogging computer's resources. We achieved this result taking a functional approach, namely utilizing Celluloid, a concurrent object oriented programming framework for Ruby.

Using Celluloid, we spawn a new computation for each thread - or in other terms, an *actor* - which runs asynchronously and writes the results back on a MongoDB database. We use a fixed amout of actors, collectively referred to as a pool, in order to prevent the computation to use all the computer's resources and also to prevent to be banned from Google. The result is a satisfying compromise between speed and available resources that allows to terminate the computation within reasonable time horizons.

CHAPTER 5

EXPERIMENTAL RESULTS

In this chapter the results of our investigation are presented. We expose the quantitative results deriving from an automated analysis of several application on the Play Store; subsequently we present a qualitative analysis and observation about the experiment.

5.1 Quantitative results

In this section we present a metric used to evaluate the compliance of an application w.r.t. its privacy policy and we expose the quantitative results in terms of such metric.

5.1.1 Metric definition

We now define a metric to evaluate the goodness of one application with respect to its compliance to its privacy policy.

First, we manually assign to each privacy-related permission a score from 1 to 3, representing the severity of its potential impact on the user's privacy, where 1 signifies a permission with low impact and 3 signifies a permission carrying a very high danger.

Such scores are arbitrarily defined, in accordance with observation and existing literature on permission analysis, and are shown in Table IV.

Secondly, we use the scores to compute a weighted sum of the number of permissions that lack an explicit mention in the privacy policy, formally defined in Equation Equation 5.1

TABLE IV: PERMISSION IMPACT SCORES

#	Permission impact scores	
1	INTERNET	3
2	READ_EXTERNAL_STORAGE	2
3	WRITE_EXTERNAL_STORAGE	2
4	ACCESS_WIFI_STATE	1
5	READ_PHONE_STATE	3
6	GET_ACCOUNTS	3
7	ACCESS_COARSE_LOCATION	3
8	GET_TASKS	1
9	ACCESS_FINE_LOCATION	3
10	READ_LOGS	1
11	RECORD_AUDIO	2
12	READ_CONTACTS	3

$$\sum_{i=1}^{n} = w_i p_i \tag{5.1}$$

$$w_i = \text{ score of the} i^{th} \text{permission} \tag{5.2}$$

$$p_i = \begin{cases} 0 \text{ if the permission is mentioned in the privacy policy} \\ 1 \text{ otherwise} \end{cases} \tag{5.3}$$

The final result is a metric estimating the compliance of an Android application to its own privacy policy. The lower the score, the more compliant the application.

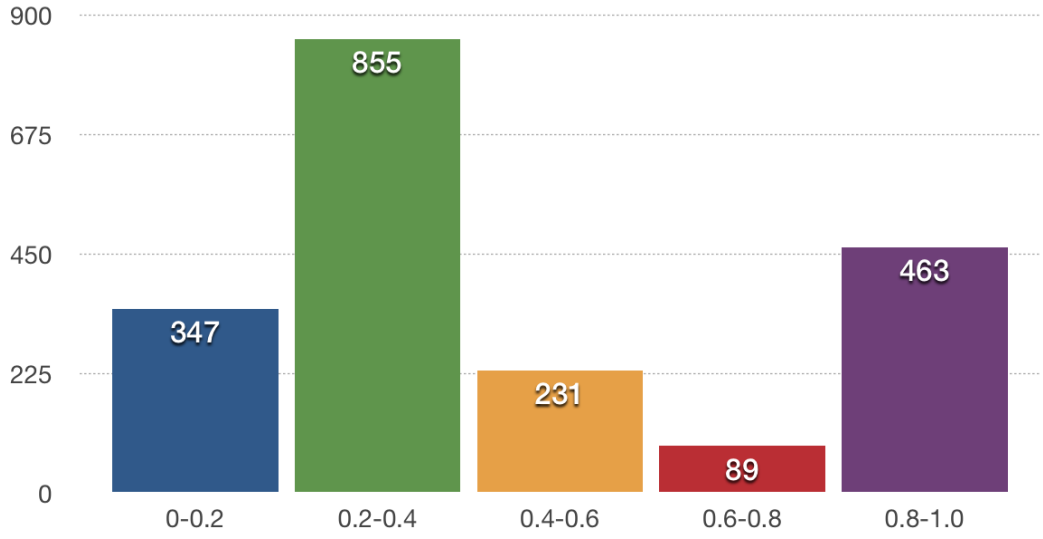

Figure 7: Quantitative results (over 4300 applications, December 7, 2013)

We run the analysis on the same 4300 applications used to generate the top used permissions and the results are shown in Figure 7

5.2 Qualitative results

Out of the thousands of applications analyzed, we now focus our attention on a few notable cases.

5.3 Case study: Shopkick

Shopkick is a popular shopping rewards app and it is known[12] to require some sensitive permissions that should worry any user of this app. Table V show the complete list of permissions of the Android application.

TABLE V: SHOPKICK APP PERMISSIONS

Permissions
INTERNET
ACCESS_NETWORK_STATE
ACCESS_COARSE_LOCATION
ACCESS_FINE_LOCATION
READ_PHONE_STATE
WRITE_EXTERNAL_STORAGE
ACCESS_WIFI_STATE
RECORD_AUDIO
CAMERA
FLASHLIGHT
VIBRATE
BLUETOOTH
GET_ACCOUNTS
RECEIVE_BOOT_COMPLETED
READ_CONTACTS
CALL_PHONE
WAKE_LOCK
READ_EXTERNAL_STORAGE
READ_CALL_LOG

We can immediately spot a few permissions with a very high impact score, for example `RECORD_AUDIO`, `CAMERA` and `ACCESS_FINE_LOCATION`.

`RECORD_AUDIO` grants the application the ability to access the device's microphone to record audio, without any explicit consent by the user other than installing the app itself. This means that the app is virtually enabled to record audio at any time, with no possibility of being disabled. Especially in combination with the `RECEIVE_BOOT_COMPLETED` permission, that allows the app to be launched when the phone has booted, and the `INTERNET`

permission, which enables sending data over the Internet, this is considerably worrying: an application could easily start itself as soon as the phone has been turned on, constantly record any sound going through the device's microphone and finally send everything over the Internet to a remote server, where the content can be stored and accessed in a later time.

To make things worse, the application also requests the permission `ACCESS_FINE_LOCATION`, meaning that the audio recording can be triggered according to the user's location, perhaps the workplace, home or other sensitive locations.

It is not hard to see how this capabilities can turn the app into a roving spyware, i.e. an application with whose hidden purpose is eavesdropping and spying on the device owners, let alone the people they are have contacts with.

Further investigations reveal how the app apparently uses the device's microphone in order to validate the physical location of the user in a store. According to the New York Times, *"The app knows someone is in a store by listening for an audio transmitter placed in each participating store; the phone's microphone picks up the signal, which people cannot hear."* [13].

We can formalize a subset of this situation in terms of the representation previously discussed in Chapter 1.

- The permission `RECORD_AUDIO` (P_1) enables the action *record_audio_from_the_device_microphone* (A_1);

- the permission `INTERNET` (P_2) enables the action *send_data_over_the_internet* (A_2);
- the permission `CAMERA` (P_3) enables the action *record_images_from_the_device_camera* (A_3);
- the permission `ACCESS_FINE_LOCATION` enables the action *detect_location_of_device* (A_4)

The combination of A_1 and A_2 enables the behavior *validate_presence_in_store* (B_1). On the other hand A_1, A_2, A_3 and A_4 can also be combined enabling the behavior *record_audio_and_video_when_user_is_at_home* (B_2).

Both behaviors result unexpected to the user, but while B_1 is probably considered legit - and even desirable -, B_2 is definitely unexpected, undesirable and possibly unlawful.

We now look at Shopkick's privacy policy looking for references of the aforementioned permissions.

5.3.1 RECORD

Our tool identifies this paragraph as relevant to the matter of recording audio:

> *"(iv) record, determine or use information about or from another content delivery platform (for example, to unlock potential rewards or offers based on your watching of a specific a commercial or show that is broadcast on your television or on the web, the shopkick application may ask you to open the app while you are watching TV, and then* ***we may record or analyze the audio signal***

> ***from the television set via the shopkick app and your cell phone's microphone**, to determine the commercial, and/or program, including the date and/or time)"* [14]

A manual inspection of the policy confirms that this is indeed the relevant section and that the permissions is covered by the privacy policy.

5.3.2 ACCESS_FINE_LOCATION

Concerning the user's location, the tool identifies this sentence as relevant:

> *(i) automatically record information that your mobile phone/device sends or transmits, including [âĂę] geographical location (if you consent to that)*

While it is true that the privacy policy covers this matter, it is also worth noting how the last phrase is misleading: as we saw before the user grants permissions at install time, on an Android device, so the consensus has already been given. Stating *If you consent to do that* gives a sense of false assurance, when it is actually a tautology on the Android platform.

5.3.3 CAMERA

The tool signals that no references have been found in the privacy policy regarding the `CAMERA` permission and a manual inspections confirms that Shopkick's privacy policy doesn't mention in any way the use of the device's camera as a medium of acquiring data.

As it currently stands, Shopkick's application can collect any image from the user's camera without they being notified and the privacy policy does not restrict this by any mean.

Our tool successfully detected this behavior, helping in identifying a gap in the privacy policy of this popular application.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

We presented a novel approach to the analysis of privacy policies in the context of Android applications. We introduced a framework for reasoning and proving properties of privacy policies, laying down the foundation for a new area of investigation.

The tool we implemented greatly eases the process of understanding the privacy implications of installing third party apps and it has already been proven able to highlight worrisome instances of applications.

The tool is developed with expandability in mind, and further developments in the approach can easily be integrated in order to increase the reliability and effectiveness.

6.2 Future Work

This thesis aims at laying the foundation for a new area of investigation, namely the relationship between mobile applications capabilities and behaviors and their privacy policies. As we mentioned in Chapter 1 several steps can be taken towards user awareness about privacy matters and this work covers the first necessary ones: identifying and analyzing the privacy-relevant permissions and examining their relationships with the privacy policies language. This enables further steps in the investigation and we now outline some of them.

6.2.1 Live monitoring of behaviors

As anticipated in Section 1.1, the first natural steps following the present work would be to live monitor the application's behavior. A static analysis can provide useful information about the *potential* behaviors that can occur, but only a dynamic observation of applications running on real device can give insights about the *actual* behaviors.

The first implementation one can think of is a passive monitoring of applications, with the final purpose of reporting such behaviors and further refine the "goodness" score presented in Section 5.1.1.

One can also think of taking a step further and turn the monitoring into an active defense: if the application is found performing a behavior clearly in contrast with its privacy policy, the monitoring tool can immediately inform the user or even prevent such behavior from happening.

6.2.2 Crowdsourced refinement of lookup tables

One the main challenges of the current approach is reliably mapping privacy policies to permissions. The current implementation occasionally incurs in false positives and negatives. For example, if we consider the sentence:

Merchandise can only be shipped to approved U.S. shipping locations.

taken from the aforementioned Shopkick's privacy policy, it is immediately evident to a human reader that, in this context, the word "location" does not refer to the collection of the user's location, whereas the current implementation considers it as such.

In order to face this challenges there are a few approaches one can think of doing: one of them is allowing the users of this tool to provide feedback on each sentence. They could either mark the sentence as *relevant* or *not relevant* and therefore improve the scoring of an application. The same approach could then be used to identify false negatives: relevant sentences can be not recognized and a user can signal such fact indicating which relevant portions of the privacy policy apply to the selected permission.

6.2.3 Improved NLP

As we mentioned above, a few approaches can be taken in order to refine the mapping of privacy policies to permissions. A complimentary approach to the aforementioned crowdsourced solution, is to improved the NLP analysis. For examples, the sentence structure can be taken into consideration. If we consider the previous example

Merchandise can only be shipped to approved U.S. shipping locations.

one can think of a smarter classifier that identifies “can be shipped” as a verbal expression and recognizes it as semantically not relevant to privacy matters when used to refer to “shipping locations”.

Such classifier would of course be much more complex to build, although simplified by the common patterns found in many privacy policies, that would make it more reliable.

CITED LITERATURE

1. Stickley, J.: The hidden risks of mobile applications. http://www.cybersecuritysummit.org/2014/pastevent/documents/Cyber Security Summit 2013 - Jim Stickley.pdf, 2013.
2. Felt, A. P., Ha, E., Egelman, S., Haney, A., Chin, E., and Wagner, D.: Android permissions: user attention, comprehension, and behavior. In Proceedings of the Eighth Symposium on Usable Privacy and Security, SOUPS '12, pages 3:1–3:14, New York, NY, USA, 2012. ACM.
3. Inc, G.: Android Developer Guide - Permissions. `http://developer.android.com/guide/topics/security/permissions.html`, 2013. [Online; accessed 23-November-2013].
4. Kelley, P. G., Consolvo, S., Cranor, L. F., Jung, J., Sadeh, N., and Wetherall, D.: A conundrum of permissions: Installing applications on an android smartphone. In Proceedings of the 16th International Conference on Financial Cryptography and Data Security, FC'12, pages 68–79, Berlin, Heidelberg, 2012. Springer-Verlag.
5. egirault: Google Play Unofficial Python API. `https://github.com/egirault/googleplay-api`, 2012. [Online; accessed 3-November-2013].
6. Yang, Q., Pan, S. J., and Zheng, V. W.: Estimating location using wi-fi. IEEE Intelligent Systems, 23(1):8–13, January 2008.
7. Limited, R. E.: Rovio Privacy Policy. `www.rovio.com/privacyâĂŐ`, 2013. [Online; accessed 24-November-2013].
8. Ltd, H. S. P.: Halfbrick Privacy Policy. `http://wac.76ff.edgecastcdn.net/0076FF/DOCS/PrivacyPolicy.htm`, 2013. [Online; accessed 24-November-2013].
9. Akdeniz: Google Play Crawler JAVA API. `https://github.com/Akdeniz/google-play-crawler`, 2012. [Online; accessed 3-November-2013].
10. Mullie, L.: Treat. `https://github.com/elouismullie/treat`, 2012. [Online; accessed 24-November-2013].
11. Milkowski, M. and Lipski, J.: Using srx standard for sentence segmentation. In Proceedings of the 4th Conference on Human Language Technology: Challenges for Computer Science and Linguistics, LTC'09, pages 172–182, Berlin, Heidelberg, 2011. Springer-Verlag.

12. Pinola, M.: Life Hacker - I Know My Phone's "Spying" on Me, But How Bad Is It? http://lifehacker.com/5864518/is-my-phone-spying-on-me-and-what-can-i-do-about-it, 2011. [Online; accessed 23-November-2013].

13. Aisle by Aisle, an App That Pushes Bargains. http://www.nytimes.com/2010/08/17/technology/17app.html?_r=0, 2010. [Online; accessed 3-November-2013].

14. Inc, S.: Shopkick Privacy Policy. http://shopkick.com/privacy-and-tos, 2013. [Online; accessed 24-November-2013].

VITA

Gabriele Petronella

Education

B.S., Engineering of Computing Systems
Politecnico di Milano
2011

M.S., Computer Science (*current*)
University of Illinois at Chicago, Chicago, IL
2013

Working experience

Co-founder
Metwit Ltd 2011-2012
Dubai, UAE - London, UK
Founded a crowdsourced weather startup and lead the develpment of the iOS client.

Co-founder
buildo s.r.l.s. 2013-present
Milan, Italy
Software architect and developer.

Research Assistant
University of Illinois at Chicago Sept 2012-Dec 2013
Chicago, IL
Research activity focused on privacy policies formal analysis.