Educating the Mobile Landscape: Building Awareness of Invasive Mobile Applications

Hilmi Abou-Saleh

Department of Computer Science
University of Calgary

Abstract

There is an ever-growing number of people using applications on mobile smartphones, with over 500 million third-party applications downloaded per day[1]. The shift to advertising-led revenue in mobile applications has been met with a simultaneous shift in data collection. Data collected includes user's personally identifying information to aid in behavioural advertising. The move towards these lucrative models has left users without an adequate understanding of the data collected on them. By building a tool that educates users, and suggests alternatives to their current applications we can build a userbase more aware and informed on how to make choice in their applications.

Key words: AppCensus, Security, Android, GDPR, COPPA.

1 Introduction

There exists knowledge gaps between smartphone users and the way third-party smartphone applications use advertising to generate revenue. These applications (apps) use personally identifying information (PII) without sufficient awareness. Such examples of PII are hardware identifiers (e.g., IMEI, WiFI MAC Address), email addresses and sensors like the user's location. In order to fill these gaps we will use the AppCensus project, which aims at giving users better transparency into how their mobile applications misuse their personally identifying information [4]. Over the last decade there has been a shift in how revenue gets collected from users. Traditionally, users purchased an application and could use it thenceforth. The last decade, however, has seen a move away from one-time purchases and towards recurring and advertising-led revenue. Using the collected personally identifying information, companies build a dossier linking the user to various events and companies. Such examples include using location data to advertise local businesses, or tracking how often a user visits to provide them a small discount to lure them in. This practice, called behavioral advertising is lucrative and requires the collection of PII. Simultaneous to this shift in revenue, the use of apps has become just as widespread, with over one hundred billion Android apps downloaded in 2018 [1]. By creating a tool that informs customers about apps on their phones we are able to begin raising awareness and providing transparency in how these apps use their information.

Articles on the invasive nature of apps are wide-spread. Mainstream news media have published articles discussing how "mobile apps take data without permission" since 2012 [3], and this has not slowed down [6]. Due to the nature of the medium, users are unlikely to change their behaviour after reading an article. These articles do not contain a solution users can take. Tom's Guide [2] published an article in late 2018 with the solution being to "reset the advertising ID associated with your device". The New York Times [5] published an article around the same time providing a few suggestions including turning off Location Services for specific applications. Many other articles suggested similar actions: opt out of ads, using a private browser, and limiting Google's own Ad/Location tracking.

These avenues are great for raising awareness with users. In 2018, the European Union released the General Data Protection Regulation (GDPR), which is by far the most ambitious and wide-ranging data protection act yet, with substantial penalties to corporations that violate its provision: 4% of worldwide revenue. Even by following the GDPR and the recommendations that the newsmedia suggest there is still a gap in transparency.

If we follow Tom's Guide's advice to reset the device advertising ID. We are left with a new advertising ID. Companies, however, collect more than one identifier, some of which are hard to reset, or cannot be reset. Limiting ad/location tracking or turning off location services as a whole is more effective, users, however, run into apps not functioning correctly because core functionality depends on it. Disregarding malfunctioning apps, companies can use many identifiers to find the user's location. This includes the use of phone's WiFi MAC Address and

the router's MAC Address. Through the combination of the two addresses and the use of router mapping tools [7] finding the user's location becomes trivial. In summary, the issue with the use of articles to inform users is the lack of transparency and direct solution for the user.

We build a tool to help inform the users on which applications are misusing their personally identifying information. By building a tool that directly tells the user that an application is misusing their privacy, we can influence a more substantial change in behaviour. We believe that this direct approach will have an impact because it encourages good behaviour. This direct approach of helping the user find better alternatives is hypothesized to be successful. In follow-up studies performing user studies will provide us with a conclusive result. Ideally the tool will suggest an alternative, equally successful and capable application. In this ideal situation we hypothesize this switch will have little to no impact on the user. In these situations it is easy for a user to switch to the alternate application. However in situations where the alternate app is not as successful, getting the user to transition will be a challenge. We argue that in these situations, where the network effect is working against the target app (e.g., migrating from Facebook to Nextdoor) a transition will be much more difficult and require additional reasons to switch. One such reason is the platform their peers utilize. A move between equally successful and capable apps (e.g., migrating from Facebook to Instagram) is possible only if their peers also use Instagram or are also making the same move.

2 Methodology

To successfully implement a tool providing transparency regarding personally identifying information three properties are required:

Property 1. An easy-to-use interface that allowed users to quickly see the offending apps

Property 2. The ability to determine offending apps on a per-user basis

Property 3. A list of information regarding the offending apps

2.1 The User Interface

The user interface (UI) is by the far the most important piece. Ensuring that the users are able to find what applications are offending, and the steps they can take to protect their PII is paramount to successful adoption. If users spend too much time without a solution they will turn else-where and use another app. Two potential platforms were explored for the user interface: building a website or mobile application. The website provides a cross-platform interface that anyone—on any device—can use. Additionally the website would be easier to support and build as it would be using a single code-base. The website, however, lacked one key requirement: it does not support **Property 2**, the ability to determine offending apps on a per-user basis. Getting this information from the browser is currently not possible, it is abstracted away for security purposes. The team behind AppCensus has already built a website that satisfies **Properties 1 and 3**. As a result our tool's main goal was implementing **Property 2** and leveraging AppCensus's existing website. To implement **Property 2** we build a mobile application. We could either build this using a cross platform tool (e.g., Google Flutter, React Native) or build a native application. A cross-platform tool would be more appealing but requires more overhead for development and design. Another consequence is inability to satisfy **Property 2** on iOS. As a result building a native application written strictly for Android is the approach we will execute.

2.2 Determining offending application on a per-user basis

This is hinted at in the previous section — finding what applications a user has installed is not as simple as it might seem. As we were both targeting mobile platforms our options were iOS and Android. Mobile phone manufacturers are becoming increasingly restrictive to ensure privacy and security of their users. This becomes an issue because getting a list of all installed applications or a list of all running applications is a valuable commodity. As a result iOS denies this capability to app developers, Android does not. Our decision is simple, we will implement this tool on Android only.

2.3 Application Information

Collecting information about applications is not a trivial task. Determining what applications are similar to others requires information that links different applications together. Android does not provide a way to collect this information. In order to make implement this feature an additional piece will be used. This piece will call an internet service that provides information about if two apps are similar (e.g., Facebook and Whatsapp are both in the social media category). On its own this similarity is not useful. There are multiple factors in how people choose applications. As stated earlier the network effect is important in bringing users to the app that their peers use. Collecting all this information requires a formal user study or existing data source. AppCensus has work underway to provide this information. They have a database containing how the application misused the users privacy.

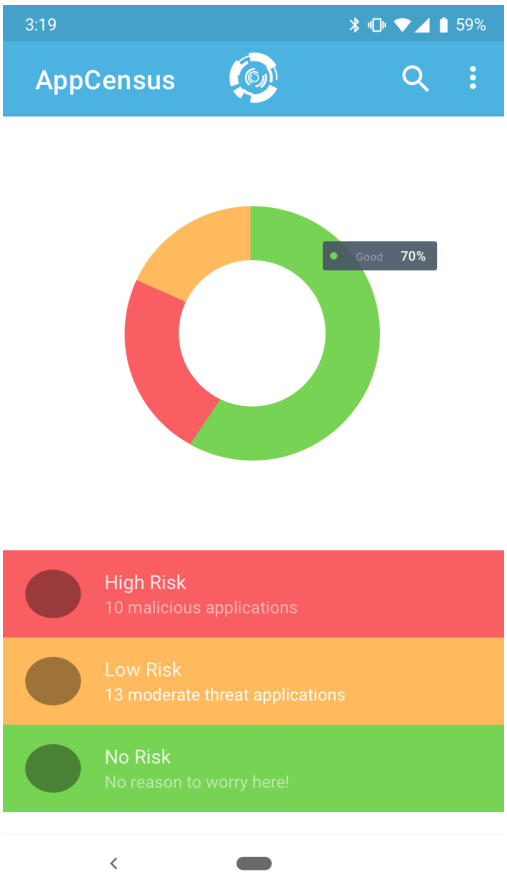

Figure 1: Final iteration of the Overview screen, designed using Sketch, the first page users see.

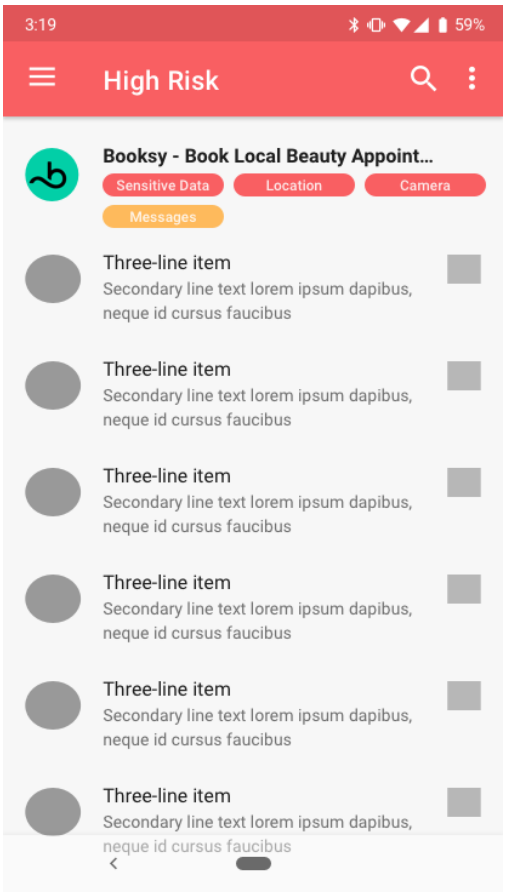

Figure 2: Iteration of the High Risk screen.

The number of entries here is ever growing as this project continues. We will implement a piece take interfaces with AppCensus's dataset.

2.4 Implementation Details

The user interface was designed using Sketch, an app aimed at creating high-fidelity prototypes. By spending time iterating on the prototype it allows us to find issues with the interface. As the requirements are vague—we have a goal to build a tool that helps users understand how applications use their PII—the time we spend designing helps flesh out the kinks. While designing we are able to familiarize ourselves with Android Studio—our primary development environment. To aid in development and because a large amount of documentation and libraries have moved over to Kotlin, we will develop our tool in Kotlin. Tying into the AppCensus's database is currently not possible—there is an ongoing effort for AppCensus to provide an Application Programming Interface (API) that will allow our tool to communicate with their information. From a functionality perspective the app currently allows us to get user feedback and in the future gives us the ability to iterate into a production ready application. **Property 2** was a complete implementation success. Every time the user launches the app, they are provided with a list of their installed applications and a privacy rating for each of the apps. Android provides this capability with a simple call, making implementing this part a breeze.

The user interface is based around three primary pages: The overview page [Figure 1] which provides a high-level health of the user's device. A list of all applications categorized by risk (e.g., High Risk, Low Risk, No Risk)[Figure 2]. Third, a detailed view describing explaining the reason for the application's category placement[Figure 3].

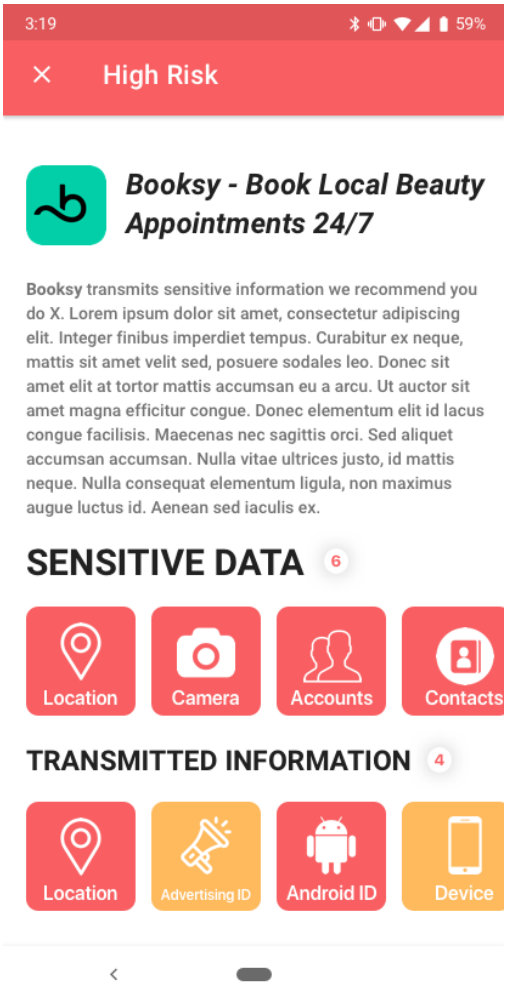

Figure 3: Initial iteration of the details page.

3 Results

3.1 Experiment Setup

To measure success we apply our initial goal, to create a tool that better equips users in today's advertising landscape, to the three properties we discuss above:

Property 1. An easy-to-use interface that allowed users to quickly see the offending apps

Property 2. The ability to determine offending apps on a per-user basis

Property 3. A list of information regarding the offending apps

From these properties we proposed a functioning, and usable Android application with the following abilities:

- to easily see offending applications,
- to identify how an application is using PII, and
- to maintain an up-to-date list of applications

To determine whether this tool is successful ideally we would undertake a large-scale user study, we leave this as future work. We recommend that this is done to inform and iterate the design and development of the tool.

Using the above measures we attempt this user study in small-scale. To accurately represent a user we download applications based on popularity, not on what is contained in the AppCensus database. This is done by first setting up a phone with at least ten popular Android applications and games. The applications are each launched once. This ensures any additional content is downloaded and the applications are functioning normally.

We now launch our tool and compare on the above measures. We evaluate each attempt in the study by giving marks to each task on a scale out of thirty-five (35), it is broken down as follows:

(a) the information provided is clear and concise, with a relevant action, **[5 marks]**

(b) a usable interface, if the application crashes this mark is a 0, **[10 marks]**

(c) the application's offences are clearly marked and, **[10 marks]**

(d) the list contains no applications that the user does not have installed. **[10 marks]**

Trial	Number of Applications Installed	(a)	(b)	(c)	(d)	Overall
1	2	2	5	10	10	27
2	2	2	6	10	10	28
3	10	3	8	10	10	31
4	20	2	9	10	10	31
5	15	3	10	10	10	33

Table 1: Experiment results

A mark of thirty is required for this to be considered a fulsome success. The number thirty was used because we understand that some of the features described above require AppCensus to finish developing their API. Without this API, providing "a relevant action" becomes difficult. As such we have removed five marks.

During development a modified version of this experiment is used to determine success. Functionality is not important during design, using items **(b)** without any weight on crashing and **(c)** we can indicate progress through design. To move on to development a score of fifteen (15) or higher is required. The reason for accepting a score under 100% is because during development further progress will be made. We anticipate that revisions will be made to overcome obstacles faced during development, and as we become comfortable with Android Studio.

The experiment is performed during development and in preparation for the conclusion of the research. To aid in development time, AppCensus has provided a sample data-set that contains a list of applications and the personally identifying information that are violated. During development this experiment is carried out with two applications installed, one with a limited number of transmitted PII, and the second with a large number of transmitted PII. The two extremes are chosen in an attempt to catch edge cases. After the end of development the experiment will be performed again, this time with over ten applications installed and with the use of physical phones.

3.2 Experiment Types

The below results are an average of the trials in Table 1:

Simulated experiment

- **(a)** the information provided is clear and concise, with a relevant action, **[2 marks]**
- **(b)** a usable interface, if the application crashes this mark is a 0, **[6 marks]**
- **(c)** the applications offences are clearly marked and, **[10 marks]**
- **(d)** the list contains no applications the user does not have installed. **[10 marks]**

Final experiment

- **(a)** the information provided is clear and concise, with a relevant action, **[3 marks]**
- **(b)** a usable interface, if the application crashes this mark is a 0, **[9 marks]**
- **(c)** the applications offences are clearly marked and, **[10 marks]**
- **(d)** the list contains no applications the user does not have installed. **[10 marks]**

The final experiment is carried out five times. Two of those times are carried out on an software emulated device. The Three runs were carried out on a physical phone, the Google Nexus 6P. During each run a different set of applications is installed on the device. Each is marked independently. The marks seen above are an average of each run, two for the simulated experiment, and three for the final experiment. Changes to our tool are made after the simulated experiment is complete. This allows us to iterate and improve on our final product.

3.3 Experiment Results

As Table 1 shows, each application has its offences clearly marked, and the list does not contain any application the user does not have installed. This success is attributed to the ease of implementation and heavy emphasis on **(c)** from early in development. The interface is designed around the transmitted identifiers that the sample data-set AppCensus provides. As Table 1 also shows, as development continues the interface becomes more usable. This is largely due to improvements and bug fixes that occur during this period. Finally, during the trials there was no improvement in **(a)**. While information clarity is improving between trials, the ability to suggest alternative applications is not implemented. Therefore no additional marks can be given to that feature.

Overall these results are successful. This is largely due to the success of our design and initial research. The design captures the requirements of our tool, providing a usable interface that clearly marks the application's offences.

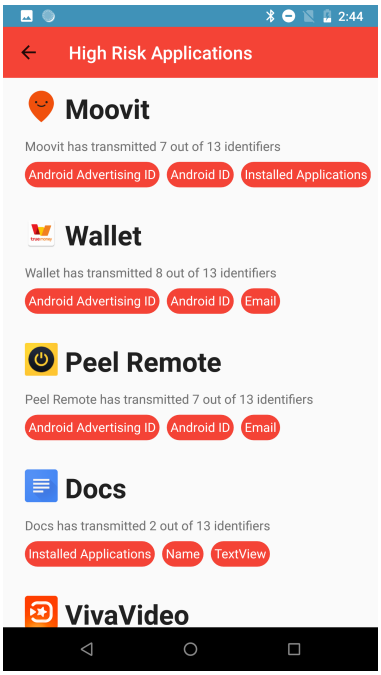

Figure 4: Final app listing

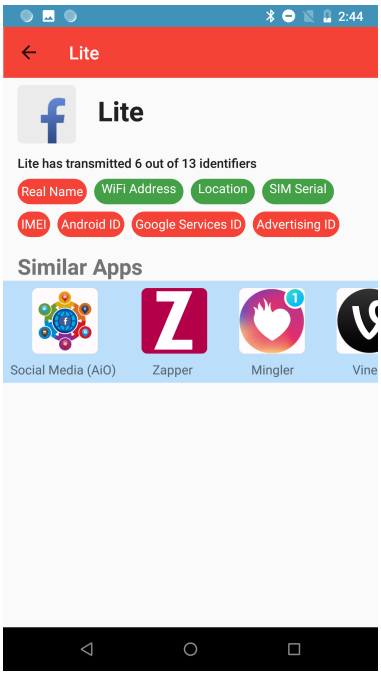

Figure 5: Facebook Lite details

4 Discussion

The progress made during this research is promising. We believe that with further development, by testing and iterating on the current product, launching AppCensus for Android will be successful. In order to be successful the user interface should be iterated upon, including suitable alternatives to the offending application and educating the user on the harms of each identifier. The designs for alternative applications are present in our high-fidelity prototype. After completion of AppCensus's API, it becomes possible to add these as functioning pages within our tool.

Developing for Android using Android Studio has steep a learning curve. Part of the success is as a result of the previous experience developing mobile applications. If our tool is to be further developed, we recommend additional time be dedicated to development. Furthermore, investigating iOS should be done for the case Apple begin to provide support for installed or running applications. Developing an application for both platforms, iOS and Android, doubles the amount of work required. To alleviate this, a multi-platform tool (e.g., Google Flutter, React Native) is highly recommended for development.

Acknowledgements

We thank the over at AppCensus for their dataset and work. This research would not have been possible without the work they have already done.

References

[1] Annual number of mobile app downloads worldwide 2022 — statista, 2017.

[2] Thousands of android apps break google's privacy rules, Feb 2019.

[3] Nicole Perlroth Bilton and Nick. Mobile apps take data without permission, Feb 2012.

[4] Irwin Reyes, Primal Wijesekera, Joel Reardon, Amit Elazari Bar On, Abbas Razaghpanah, Narseo Vallina-Rodriguez, and Serge Egelman. wont somebody think of the children? examining coppa compliance at scale. *Proceedings on Privacy Enhancing Technologies*, 2018(3):6383, 2018.

[5] Jennifer Valentino-devries and Natasha Singer. How to stop apps from tracking your location, Dec 2018.

[6] Jennifer Valentino-devries, Natasha Singer, Michael H. Keller, and Aaron Krolik. Your apps know where you were last night, and they're not keeping it secret, Dec 2018.

[7] Jenna Wortham. Skyhook lets wi-fi signals take the place of gps, May 2009.