Mobilitics: Analyzing Privacy Leaks in Smartphones

by Jagdish Prasad Achara, Franck Baudot, Claude Castelluccia, Geoffrey Delcroix and Vincent Roca

Who, do you think, is aware of almost everything you do? Well, it's probably right there in your pocket, if you own a smartphone and carry it with you. In order to evaluate the actual privacy risks of smartphones and to raise public awareness of these risks, the CNIL (French data protection authority) and the Inria (French public science and technology institution dedicated to computational sciences) Privatics team started working together in 2012 as part of the Mobilitics project.

It is no surprise, given smartphones' convenience and utility, to see their wide adoption worldwide. Today, there are 1.08 billion smartphone users out of a total of five billion mobile phone users worldwide, and the ratio is constantly increasing. Smartphones are used not only to communicate but also to browse the web and run various internet-enabled Apps. As a result, they contain a lot of information about the cyber activities of their owners, and therefore users' interests and behaviours. Furthermore, smartphones are also equipped with GPS, NFC and Bluetooth units, along with a digital camera and are almost always connected to the Internet; thereby revealing a lot of information about the physical activities of their owners. On top of this, smartphones are very personal to the user and are barely turned off.

For the aforementioned reasons, combined with the fact that users tend to carry smartphones wherever they go, they are an ideal target for marketers who want to profile users to profit from their personal data. Some studies even suggest that the main business model for some developers (in the case of free Apps, for example) is based on the collection of personal data. As a result, many Apps might be leaking personal information to third parties, such as Analytics and Advertising (A&A) companies.

A few insights

The goals of the Mobilitics project are to investigate smartphone Operating Systems (OSs) and Apps for potential privacy leaks and to inform their users about the privacy risks. The project currently targets two OSs, namely Android and iOS, because they cover almost 75% of the whole smartphone OS market share.

As part of this project, we have developed a software solution (an Android version with similar functionalities is under development) for iOS to capture access to private information by various Apps. When an App makes a call to the iOS API to access a broad list of a user's personal data, eg Contacts, Location, Device Name, UDID, Calendar, Reminders, Photos, Notes and Accounts, our software logs this event for later analysis. Note that some Apps do actually need to access personal data to provide the desired service. These applications do not breach user privacy if they only process and use the personal data to provide the desired service and don't transmit the data to remote third parties. In order to detect personal information leakage, we also monitor whether the accessed personal data is sent to a third party, as in [1] and [2] but by using a different approach. Additionally, we are also developing a visualization tool to help people understand the privacy implications by aggregating, interpreting and displaying all private data stored and/or sent by various Apps.

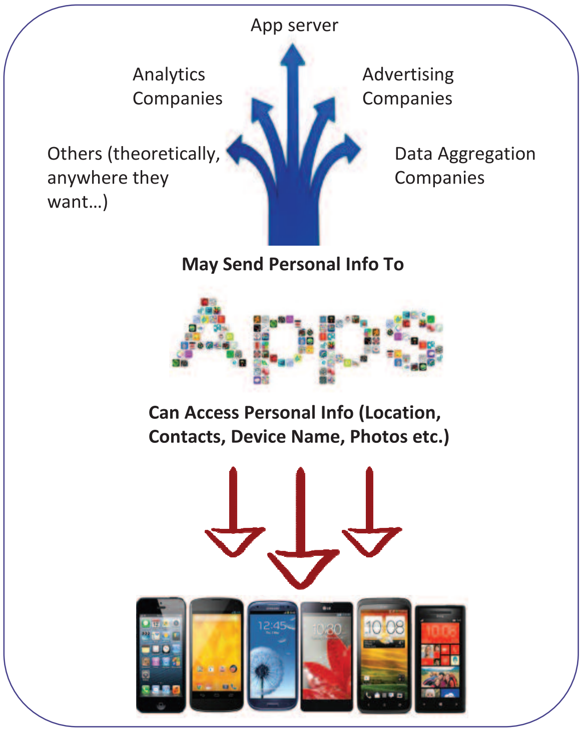
App server
Analytics Companies
Advertising Companies
Others (theoretically, anywhere they want...)
Data Aggregation Companies
May Send Personal Info To
Apps
Can Access Personal Info (Location, Contacts, Device Name, Photos etc.)

Figure 1: Android and iOS currently don't provide any mechanism to let users know how their personal information is being used by various Apps. Will it be used locally on the device or sent to remote servers? Being aware of it, users can probably make better decision whether to allow/deny access to their personal information for a particular App.

For instance, our iOS tool reveals that many Apps are accessing the Unique Device ID multiple times (in the order of hundreds), which implies that it is probably being used for online tracking of the user. Some Apps are also, surprisingly, accessing the user's device name although there does not seem to be any obvious reason to do so. The name of the device is set during the initial device setup and often contains the real name of the user. Moreover, even if the user does not set it to his or her real name, it might easily be used for tracking purposes since the device owner does not generally modify it after the initial setup.

Our software solution was developed in 2012 for iOS 5.x before Apple launched iOS 6 in September 2012. iOS 5.x didn't seek the user's permission for private data access except for location information. In iOS 6, Apple decided to change its strategy and introduced a new privacy-specific setting giving the user control over whether an App can access private data: the user is prompted the first time an App tries to access Contacts, Location, Reminders, Photos, Calendar and Social Networking accounts and later, iOS remembers and follows the user preferences.

In our opinion, this is a decent step by Apple towards making iOS privacy-friendly. However, several questions still remain open: is the list of private data included in their privacy-settings sufficient? Is an authorization that does not consider any behavioural analysis sufficient? For instance, accessing the device location upon App installation, to enable a per-country personalization, is not comparable to accessing the location every five minutes. Also, does the App keep the personal information locally for internal purposes, or is it communicated to external servers? If the latter, where exactly are these servers? Moreover, A&A libraries included by the App developer also have access to the same set of user's private data as the App itself. However, a user giving access to his or her Contacts doesn't necessarily indicate consent for these data to be shared with A&A companies. Might this pave the path for privacy invasion? These are the questions that Mobilitics will attempt to answer.

Conclusion

Our preliminary results and the various scandals that occured in 2012 show that privacy considerations are of utmost importance if we want to continue using these devices with serenity. We believe that smartphones can't be, in the long run, black boxes to their owners because nobody wants these great devices in our pockets to be the ultimate spy.

Mobilitics is a CNIL-Inria project that involved the following participants: Jagdish Prasad Achara, Franck Baudot, Claude Castelluccia, Geoffrey Delcroix, James Douglas Lefruit, Gwendal Le Grand, Stéphane Petitcolas, and Vincent Roca.

Links:
https://team.inria.fr/privatics/mobilitics/
http://www.go-gulf.com/blog/smartphone/
http://kwz.me/04
http://blogs.wsj.com/wtk-mobile/
http://kwz.me/00

References:

[1] W. Enck et al: "TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones", in proc. of OSDI 2010

[2] M. Egele et al: "PiOS: Detecting Privacy Leaks in iOS Applications", in proc. of NDSS 2011.

Please contact:

Jagdish Prasad Achara, Claude Castelluccia, Vincent Roca
Inria, France
Tel : +33476615215
E-mail:
{Jagdish.Achara,Claude.Castelluccia, Vincent.Roca}@inria.fr

Privacy-Preserving Interest-Cast for Android Smartphones

by Gianpiero Costantino, Fabio Martinelli and Paolo Santi

We present an implementation of the FairPlay framework for secure two-party function computation on Android smartphones, which we call MobileFairPlay. Our application was developed to preserve the users' privacy within opportunistic networks considering the interest-casting model. Our tests show that the running times of the protocol on several Android phones, are very reasonable (up to five seconds in the worst case).

Crowded places present an opportunity for people to share personal information. In addition to sharing information through traditional, web-based platforms and applications such as Facebook and Twitter, the availability of short range radio interfaces in smartphones, tablet PCs, etc. allows individuals to share information with one another through direct, opportunistic communication (typically using the Bluetooth or WiFi interface).

This model of store-carry and forwarding data to others is known as opportunistic networking (OppNets). A common feature of these approaches is that, before making a decision about whether to share information with an individual, users have to exchange some sensitive information, such as history of past encounters [1], interest profiles, etc. Given that the person encountered is generally a stranger, this exchange of sensitive information (which occurs in plain text in the approaches mentioned) is likely to be deemed unacceptable by the user in real-world scenarios, owing to privacy concerns.

To address this issue, we present a feasible implementation of a cryptographic framework for secure multi-party computation (the FairPlay framework proposed in [2]) targeted to the interest-cast model and running on the Android mobile platform. Our application, "Mobile-FairPlay" [3], has been developed with the aims of: 1) finding people in the user's (Alice) neighbourhood through a Bluetooth scan operation, 2) connecting to another user (Bob) and determining whether Bob and Alice have similar interest profiles without disclosing sensitive information, and 3) sharing messages between Alice's and Bob's devices in the event that their profiles are similar.

When Alice and Bob have established a new connection, Bob, who received the connection, randomly selects different topics to verify their similarity with respect to these interests. Then, they start matching interests using the secure framework implemented in the App. During this execution, both Bob and Alice use their own value for the selected topic, extracted from the interest profile. However, these values are not sent to the other participants in plain, but are encoded in the garbled Boolean circuits exchanged through MobileFairPlay. At the end of the handshaking phase, Alice and Bob only know the result of the jointly computed