How to Assure Privacy on Android Phones and Devices?

Christoph Stach†
University of Stuttgart
Institute for Parallel and Distributed Systems
Universitätsstraße 38
70569 Stuttgart, Germany
Email: Christoph.Stach@ipvs.uni-stuttgart.de

***Abstract*—There is an increasing popularity of mobile devices—especially Android devices—particularly because of the huge amount of available third-party applications. Albeit, the number of diagnosed hacker attacks against mobile user increased in unison, as these devices became the prime target of the latest malware, thanks to inexperienced users and a negligent way of dealing with private data. To make matters worse, the Android permission system is much too coarse-grained and too hard to grasp for an average user. However, even if a user is able to comprehend the meaning and impact of a certain permission, in the end, s/he must grant all requested permission anyhow, if s/he wants to install the application.**

Therefore, we introduce PMP a privacy management platform for Android, which enables a user to grant an application fine-grained access rights to critical data. Furthermore, those rights can depend on some contextual constraints (e.g. Internet usage is restricted to certain locations) and the policy rules can be modified at run-time. Depending upon the granted rights an application provides a different scope of service. Moreover, the user is—due to a catchy GUI—always informed what impact the granting or respectively the revocation of a permission has on the application's service quality.

***Index Terms*—privacy management; Android; demonstrator.**

1. Introduction

When looking at the mobile handset market, this manufacturing branch has a steady compound annual growth rate of almost 25% since 2009. Approximately every second sold mobile handset is a smartphone. Thereby, the smartphone market is dominated by the duopoly consisting of Google and Apple [1]. These *smart* devices are able to collect a lot of context information (e.g. location data) and are also used to store a lot of private data (e.g. contacts). A third key feature of these devices is the capability to obtain third-party applications easily. Largely, these applications represent a novel breed of applications combining all sorts of information sources in order to augment the user experience.

However, with great power comes great responsibility, thus even the seemingly most innocent application might be a critical privacy risk, if the user is not able to identify precisely which data is actually accessed by it. A small negligence might result in a great harm, rapidly (e.g. [2]). Occasionally, it is even the smartphone OS vendor itself who accesses some private data of a user for obscure reasons [3].

Unfortunately, it seems that none of the currently existing privacy protection systems is sufficient enough to provide comprehensive security or fulfill the community request for more flexibility and freedom [4]. Thus, we came to realize that there is a tremendous need to manage the permissions of an application in a more fine-grained manner. Here it is all-important that the privacy settings can be modified at run-time without causing an application to crash. On top, it should be possible to provide an application with blurred data instead of the privacy compromising original one. The user has to be informed in more detail what is happening to his or her sensitive data, so that s/he can reasonably decide which application should be granted access to which data. Furthermore, it is a necessity to add contextual constraints to the privacy policies. E.g. imagine an application monitoring crucial health data. There is no need for such an application to reveal this information to anyone—unless in case of an emergency. Starting from this idea we built the *Privacy Management Platform (PMP)* [5] as a result of a close collaboration with Google Munich office. To visualize the necessity of such a platform, we come up with a sample application which serves as an open playground for the PMP.

The remainder of this paper is structured as follows. Section 2 gives an overview about the state of the art and the related work on privacy protection for mobile devices. Section 3 introduces an approach towards a better privacy protection mechanism before we illustrate in Section 4 the proposed demonstration scenario.

2. State of the Art

As smartphone users get more and more aware of privacy issues, this might become a major selling point in the near future. A mobile OS vendor has to map out a strategy how to assure the greatest possible protection for any private data. Basically, one can differentiate two strategy types [6]:

Walled-Garden Model. In the *Walled-Garden Model*, the OS vendor seals its devices off from the rest of the

† *This work was supported in part by a Google Research Award.*

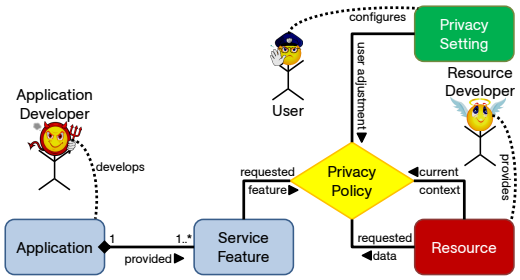
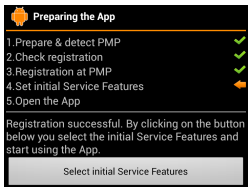
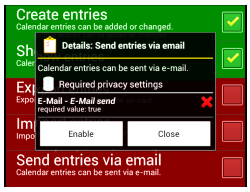


Figure 1. PMP's Privacy Policy



(a) Installation Dialog



(b) Privacy Settings

Figure 2. New Application Installation Process

world. The vendor controls which applications are available via its own market and doesn't allow the user to install an application from anywhere else. However, the user is not only patronized, but s/he has also to rely on the good-natured handling of all personal data by the OS vendor, as usually there is virtually no information for what purpose this data is needed and / or is used.

User Control Model. In contrast to the Walled-Garden Model, the *User Control Model* gives the user (almost) total freedom—along with full responsibility—concerning application installation and usage. Therefore, it has to be ensured that the user is provided with sufficient information about the data usage of an application, so s/he can make the best-fitting decision. Unfortunately, this information is typically very sparse and the adjustment possibilities are severely limited. Most likely the user is facing an all-or-nothing decision—either s/he installs the application and accepts all of its requirements or s/he revokes the requested permissions and therewith abandons the application.

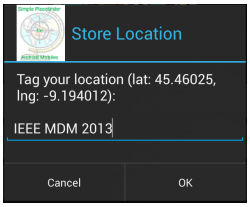
Google applied the *User Control Model* in the Android OS and, to make matters even worse, they ask the user whether the requested private data access of an application shall be permitted at installation time, only. The cause of this access and moreover the usage of this potentially private data remains concealed. Thereby, it is hardly surprising that especially the community of Android users craves for better privacy settings as fine-grained as possible.

3. The Privacy Management Platform

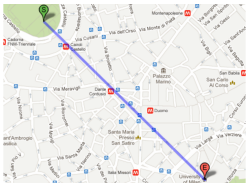
After dealing with currently existing privacy preserving approaches in detail, we came to the reason that none of them provides the protection and flexibility as it is demanded by the mobile community (see [5]). Therefore, we introduced the **Privacy Management Platform (PMP)**, a fine-grained mechanism to enable a user to monitor potential critical data access and protect any private data appropriately. The most prominent features of the PMP are its *extensibility*—in terms of adding further objects to be monitored—its *context sensitivity*—as the privacy policy can be augmented by context references—and its *user interaction*—as a user always gets adequate feedback information.

However, these three purposes cannot be achieved with the current permission system of the Android OS. With this in mind, we introduced our own privacy policy model, as seen in Figure 1. In our model the *privacy policy* consists of three novel components: *Service features*, *resources*, and *privacy settings*. An *application* provides various service features. Each service feature aggregates any given subset of the application's functionality regarding a common topic (e.g. *"Show current location on a map"*). As all applications might originate from potentially hazardous developers, their data access has to be monitored and, if necessary, restricted by the PMP. For that purpose, we encapsulate various Android system services (for instance location information provided by the LocationManager) in so called resources. These resources have influence on the privacy policy in two different ways: On the one hand they serve as a source for the application's requested data and on the other hand they provide the application's context at run-time to determine whether a context-sensitive policy rule has to be applied. The privacy settings specify to which extent a user may adjust the usage of a certain resource. This exceeds the prevailing permission system by far, as e.g. an appropriate setting for the location resource might not only be *on* or *off*, but also *up to a precision of* 100 *meters*. On top of that, trusted developers may enhance existing resources or create further resources and make them public—keep in mind that the PMP resources are, contrary to the Android permissions, not mandatorily linked to a given hardware feature, but can also encapsulate certain features of the system. E.g. there could be a resource which determines the current location and forwards it to one (and only one) particular server address.

But how does that work in practice? First at all, with the PMP the current Android permissions become obsolete and should therefore no longer be used. The PMP works as an information broker between an application and the system. Any request has to be raised to the PMP and the PMP forwards any resulting data—perhaps altered or even faked if the privacy setting stipulates it. So, a user does not have to grant any permission to an application during the installation process. When an application is started for the first time it automatically registers itself to the PMP (see Figure 2a). In this process the PMP informs the user which service features are provided by this application. Due to the current policies, some of these features might be enabled (highlighted in green) and some might be disabled (highlighted in red). When a user changes this preset, s/he gets informed which privacy settings need to be altered for this purpose (see Figure 2b). If a service feature is grayed out, then an associated resource is currently not installed



(a) Tagging a Location



(b) Navigation by Map

(c) Navigation by Compass

(d) Map with all Tagged Locations

Figure 3. The Simple Placefinder for Android Mobiles

on the device. However, any resource available on the PMP server can be directly fetch and installed within this dialog. After the registration the application as well as all resources can be managed via the PMP menu. From there all privacy settings can be subsequently adjusted—even at run-time! We refer the reader to [5] for a detailed description of the PMP.

4. Demonstration

To demonstrate PMP's functionality, we implemented the **S**imple **P**lacefinder for **A**ndroid **M**obiles (**SPAM**). With SPAM you can store geographical coordinates and tag them with a label (as shown in Figure 3a). Afterwards, SPAM guides you back to that location either by showing the path on a map (see Figure 3b) or simply by indicating the approximate direction with a compass needle (see Figure 3c). SPAM can also output the complete history—i.e. SPAM displays all tagged locations on a map (see Figure 3d).

One imaginable use case for SPAM might be, that you parked your car in a foreign town, did some sightseeing, and cannot remember afterwards where that parking place was—not to mention how to get there. With SPAM, you just store the coordinates as soon as you leave your car and tag them with a meaningful label (e.g. "*parking lot*"). When you want to go back, you simply select that stored location and allow SPAM to guide you.

Such a handy little application requires a lot of Android permissions in order to run properly—inter alia ACCESS_FINE_LOCATION (to get the user's accurate position), INTERNET (to generate the map with the suggested route), or SEND_SMS (to share some points of interest with your clique). However, not each of SPAM's functions requires every permission. The user might want to revoke some permissions even if s/he looses some features. Each of SPAM's functions considered separately seems fairly harmless but with all permissions SPAM is able to create a user profile and share it with anybody which is presumably undesirable (e.g. for personalized spam). In this demo session,

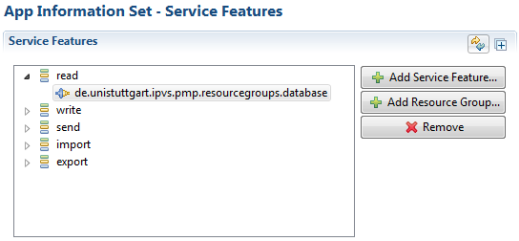


Figure 4. PMP's Eclipse Plugin

any interested visitor can test SPAM and it's interaction with the PMP. S/he can adjust any privacy setting and will notice an immediate effect on SPAM's scope of service.

In addition, we present the so called *PMP Eclipse Plugin*. As the PMP requires some additional information about an application (e.g. its provided service features), admittedly there is a certain overhead for a developer. Therefore, the PMP Eclipse Plugin supports the development of PMP compatible applications (as well as the development of new resources) on the one hand by auto-generating templates for required components and on the other hand by guiding the developer through auto-completing forms, as seen in Figure 4. Furthermore, the plugin is able to parse the entered data and check them for errors.

If you are not able to attend the demo session, feel free to try the PMP on your own device, since it is available online bundled with some small sample applications:

http://code.google.com/p/pmp-android

Acknowledgment

The PMP resulted from a close collaboration with Google Munich office. Hence, we would like to thank Google for their useful advices and ideas in the planing phase, and their given support during the implementation phase.

References

[1] VisionMobile, "Developer economics 2013," Developer Economics, Tech. Rep., 2013.

[2] B. X. Chen and N. Bilton, *Et Tu, Google? Android Apps Can Also Secretly Copy Photos*, The New York Times, Mar. 2012. [Online]. Available: https://bits.blogs.nytimes.com/2012/03/01/android-photos/.

[3] K. M. Heussner, *Why Are Apple, Google Tracking Your Phone?* ABC News, Apr. 2011. [Online]. Available: https://abcnews.go.com/Technology/google-apple-track-users-location-information/story?id=13436330.

[4] A. P. Felt *et al.*, "Android Permissions: User Attention, Comprehension, and Behavior," in *SOUPS '12*, 2012.

[5] C. Stach and B. Mitschang, "Privacy Management for Mobile Platforms - A Review of Concepts and Approaches," in *MDM '13*, 2013.

[6] D. Barrera and P. van Oorschot, "Secure Software Installation on Smartphones," *IEEE Security and Privacy*, vol. 9, pp. 42–48, 2011.