Respecting User Privacy in Mobiles: Privacy by Design Permission System for Mobile Applications

Karina Sokolova*†, Marc Lemercier*

*University of Technology of Troyes
Troyes, France
{karina.sokolova, marc.lemercier}@utt.fr

Jean-Baptiste Boisseau†

†EUTECH SSII
La Chapelle Sain Luc, France
{k.sokolova, jb.boisseau}@eutech-ssii.com

Abstract—**The Privacy by Design concept proposes to integrate respect for user privacy into systems managing user data from the early design stage. This concept has increased in popularity and the European Union (EU) is enforcing it with a Data Protection Directive. Mobile applications have emerged onto the market and the current law and future directive are applicable to all mobile applications designed for EU users. It has so far been shown that mobile applications do not suit the Privacy by Design concept and lack transparency, consent and security. Current permission systems are judged as unclear for users. In this paper, we introduce a novel permission model suitable for mobile application that respects Privacy by Design. We show that such adapted permission system can improve not only the transparency and consent but also the security of mobile applications. Finally, we propose an example of the use of our system in mobile application.**

Keywords–permission, permission system, mobile, privacy by design, privacy, transparency, control, Android, iOS, application, development, software design, pattern, mobility, design, modelling, trust

I. Introduction

The idea of this paper was first outlined in [1] and extended in this article.

Mobile devices continue to gain in popularity. Thousands of services and applications are proposed on mobile markets and downloaded every day. Smart devices have a high data flow, processing and storing large amounts of data including private and sensitive data. Most applications propose personalized services but simultaneously collect user data even without the user's awareness or consent. More and more users feel concerned about their privacy and care about the services they use. The survey conducted by TRUSTe in February 2011 shows that smartphone users are concerned about privacy even more than about security (second in the survey results) [2].

Nowadays, people are aware of the lack of privacy, especially while using new technological devices where information is collected, used and stored en masse (Big Data). Privacy regulation aiming to control personal data use is in place in many countries. European Union privacy regulation includes the European Data Protection Directive (Directive 95/46/EC) and the ePrivacy Directive. United States regulation includes the Children's Online Privacy Protection Act (COPPA) and the California Online Privacy Protection Act of 2003 (OPPA). Canada has the Personal Information Protection and Electronic Documents Act (PIPEDA) concerning privacy.

The Privacy by Design (PbD) notion proposes to integrate privacy from the system design stage [3] to build privacy-respecting systems. PbD proves systems can embed privacy without sacrificing either security or functionality. Some PbD concepts are already included in European data legislation; the notion is considered to be enforced in European Data Protection Regulation, therefore, systems made with PbD are compliant with the law. An application of PbD notions is not only a benefit for users but also a legal obligation for developers.

The PbD concept was first presented by Dr Ann Cavoukian. She proposes seven key principles of PbD enabling the development of privacy-respecting systems. The system should be proactive, not reactive, embed the privacy feature from the design stage, integrate Privacy by Default, respect user privacy, data use should be transparent to the end user and the user should have access to the mechanism of control of his data. Full functionality and end-to-end security should be preserved without any sacrifice [3].

Mobile privacy was discussed in 'Opinion 02/2013 on apps on smart devices' by the Article 29 Data Protection Working Party [4], in which the opinion on mobile privacy and some general recommendations were given. The article states that both the Data Protection Directive and the ePrivacy Directive are applicable to mobile systems and to all applications made for EU users. Data Protection Regulation is also applicable to mobile systems. The article defines four main problems of mobile privacy: lack of transparency, lack of consent, poor security and disregard for purpose limitation.

Many reports such as [5] propose recommendations on mobile privacy improvement repeating basic privacy notions (e.g., data minimization, clear notices) but the exact patterns or technical solutions are missing.

Permission systems are embedded in mobile systems and are a crucial part of mobile security and privacy. Nowadays, permission systems do not follow Privacy by Design notions. Many works concentrate on analysing and modelling the current permission systems [6][7][8], on improving current permission systems to give more control to the user [9][10][11][12] or to add additional transfer permissions [13], on visual representation of permissions [14], on user perceptions of current permission systems [15][16][17], on data flow analyses (possible data leakage detection) [18][19][20] and on current permission enforcement and verification [21][22][23][24].

Few works try to redefine permission systems. In [25], authors propose an ontology where the right is given to an actor to take action on data. Rules are defined using SWRL language. Complex rules forbidding or allowing data access under a particular condition can be added to the policy. It is not clear what other action than the 'access' is supported by the ontology. A firewall was implemented on Java and ported to Android. The perception of users and the applicability to real applications is not disguised.

Authors of [26] propose a per-data permission system: more fine grained than Android default permissions. Access permissions are granted by data stored in SQLite databases and the access can be restricted by a piece of data - column (only phone number, only names from contacts) or by group - raws (only work contacts can be accessed) and mixed - cell. Privacy policy is expressed with subject having (1) or not having (0) access to an object. The permission system was implemented on Android and included used contact list data access permissions.

Current works are often limited by the data they can manage (use only geolocation data, SMS, address book, etc.), number of actions (most include only the 'access' action). Most works modify the functionality of current permission systems: allow permission to be revoked, return fake or empty data to the functionality. The perception of users and the applicability to real applications is often not disguised.

To our knowledge no work has been conducted on redefining the permission system to fit the Privacy by Design notion or on adding the purpose to permissions.

The remainder of the paper is organized as follows: Section II describes current permission systems of iOS and Android and points out problems regarding Privacy by Design. Section III introduces our proposal: the pattern of the privacy-respecting permission system. We show that it can cope with the transparency, consent and purpose-disregard problems and also improve security. Section IV shows the application of our novel permission system to the real mobile application. The paper ends with a conclusion and future works.

II. Existing Mobile Permission Systems

In this section, we present current iOS and Android permission systems and evaluate those systems regarding the PbD notion. We take into account the full functionality allowed by the permission system, privacy by default, transparency and the control notions.

- Full functionality: possibility to use all functionalities available on the platform.
- Privacy by Default: the default configurations of the system are privacy protective.
- Transparency: user should clearly understand what data is used, how and for what purpose.
- Control: user should have full control over his personal data use.

We consider the privacy policy to be very important for the proactive and transparent system, therefore, we present the state of application privacy policy in both systems.

A. Permissions

iOS and Android have different strategies in regard to access to the device data. The iOS platform gives non-native applications access only to the functionalities listed in privacy settings: location services, contacts, calendar, reminder, photos, microphone and Bluetooth (sensitive data, such as SMS and e-mails are not shared at all). Recently, the connection to Facebook, Twitter, Flickr and Vimeo was added to the platform (iOS7). Full functionality is given up for privacy reasons as applications cannot use the full power of the platform but only a limited number of functionalities.

An iOS application should have permission to access the information listed above. By default, an installed application has no permission granted. The application displays a pop-up explaining what sensitive data it needs before accessing it. The user can accept or decline permission. If permission is declined, the corresponding action is not executed. If the permission is accepted, the application obtains access to the corresponding data. The user is asked to grant permission only once, but he can enable or disable such permission for each application in privacy settings integrated by default into the iOS. iOS thereby maintains transparency, control and privacy by default.

The Android system remains on the sharing principle. Full functionality is preserved: applications have access to all native applications' data and can expose the data themselves. Applications need permission to access the data, but unlike iOS, users should accept the full list of permissions before installing an application. While all permissions are granted, an application has full access to the related data. Some Android permissions tagged as 'dangerous' can be prompted to the user every time the data is going to be accessed, but it is rarely the case. Users see the list of dangerous permissions on the screen before installing the application.

Android proposes more than 100 default permissions and developers can add supplementary permissions. Multiple works show that users do not understand many default permissions and fail to judge the application privacy and security correctly using the full permission list [15][17]. Permissions do not clearly show what data is used for and how. Moreover, some other studies show the abusive usage of Android permissions by developers [27].

Some users do not check the Android permission list because they need a service and they know that all permissions should be accepted to obtain it. Android permission lists look like a license agreement on a desktop application that everybody accepts but very few actually read [28].

Android user does not have any control over permissions once the application is installed: permissions cannot be revoked. Android does not include an iOS-like system permission manager (privacy settings) by default, therefore, the user has to enable or disable the entire functionality to disable access to related data (e.g., Wi-Fi or 3G for Internet connection; GPS for geolocation) or to use additional privacy enhancing applications.

TABLE I
ANDROID AND IOS PERMISSION SYSTEMS COMPARISON

	Full Functionality	Default Settings	Transparency	Control
Android	+	-	-	-
iOS	-	+	-/+	+

Both iOS and Android default permission systems mostly inform about data access, but not about any other action that can be completed with the data. For example, no permission is needed to transfer the data. Android and iOS include permissions for functionalities that can be related to personal data transfer, such as Bluetooth and Internet. Permissions can be harmless to users, but there is no indication of whether personal data is involved in a transaction. This decreases the transparency of both platforms.

Android and iOS permissions do not include purpose explanation. An iOS application helps to understand the purpose by asking permission while in use, but if an application has a granted permission once for one functionality it could use it again for a different purpose without informing the user. Android users can only guess what permission is used for and whether the use is legitimate.

Table I shows the system differences regarding four main privacy notions: full functionality, transparency, control and privacy by default. One can see that the current Android permission system is lacking in transparency, control and default privacy; iOS sacrifices functionality and also lack of transparency. Permissions are often functionality-related and users fail to understand and to judge them. Personal data use is unclear and the purpose is missing.

B. Privacy Policy

Users should choose applications they can trust. Apple ensures that applications available on the market are potentially harmless, although Android users should judge the application for themselves with the help of information available on the market. The AppStore and Google Play provide similar information: name, description, screenshots, rating and user reviews.

The transparency and the proactivity of the system can be improved by including the privacy policy in the store. A user can be informed about the information collected and stored before he downloads the application. Without any privacy policy, the user can hardly evaluate the security and privacy of the application, only the functionality and stability of the system. In their feedback, users often evaluate the functionalities and user interfaces and report bugs, but they rarely indicate privacy and security problems.

iOS does not require developers to include the privacy policy in the application but only in applications directed at children younger than 13 years old. Apple encourages the use of privacy policy in the App Store Review Guidelines and iOS Developer Program License Agreement. Apple specifies that developers should ensure the application is compliant with all laws of the country the application is distributed in. On viewing the App Store Review Guidelines one can see that all Privacy by Design fundamental principles and data violation possibilities are covered by Apple verification. However, the exact evaluation process used by Apple remains secret and some privacy-intrusive applications may appear in the store. Until recently, Apple authorized the use of device identification. This identification number was not considered private. Many applications used this number to uniquely identify their users, therefore, many applications were considered privacy intrusive [29].

Google Play Terms of Service do not require any privacy policy to be added to the Android applications. Google provides an option to include the privacy policy but does not verify or enforce it. Google Developers Documentation provides recommendations and warns that the developer has a responsibility to ensure the application is compliant with the laws of the countries in which the application is distributed.

Some developers include a license agreement and privacy policy. According to [30] only 48% of the top 25 Android paid applications, 76% of the top 25 Android free applications, 64% of iOS paid applications and 84% of iOS free applications have included the privacy policy. Android includes the permission list in the store and this can be considered a privacy policy, but, as previously discussed, the list is unclear to the final user.

III. PRIVACY-RESPECTING PERMISSION SYSTEM

Mobile phones have significant data flow: information can be received, stored, accessed and sent by the application. Data can be entered by the user, retrieved from the system sensors or applications, come from another mobile application, arrive from servers or from other devices. Data can be shared on the phone with another application, with servers or other device.

The permission system is integrated into mobile operating systems; well designed, it makes a proactive privacy-respecting tool embedded in the system.

We propose to focus permissions on data and the action that can be carried out on this data, rather than on the technology used. The definition of purpose of the data use is included in our permission system.

Privacy Policy should be short and clear. Users should have a global vision of the data use and functionalities before they install an application. Users rarely read long involved policies, especially when they want a service and feel they have no choice but to accept all permissions. Our permission system enables a simple policy to be generated with a list of permissions.

A. Permission definition

We model our permission system with an access control model. We choose discretionary access control where only data owner can grant access. The user should be able to control the data, therefore, we consider the user is a unique owner of all information related to him.

R_{app} is a set of $rules$ assigned to the application. We define a $rule$ as an assignment of the $\mathcal{R}ight$ over an $\mathcal{O}bject$ to a $\mathcal{S}ubject$. The $rule$ triplet is defined as follows:

$$\forall rule \in \mathcal{R}_{app},\ rule = (s,\ r,\ o) \tag{1}$$

where $s = \mathcal{S}ubject,\ r \in \mathcal{R}ight,\ o \in \mathcal{O}bjects$

We define a mobile application as a $\mathcal{S}ubject$. Each mobile application is associated with the unique identification number can be used as a $\mathcal{S}ubject$.

$$\mathcal{S}ubject = Mobile\, Application\, ID \tag{2}$$

$\mathcal{O}bjects$ are the user-related data, such as e-mail, contact list, name and surname, phone number, address, social networks friend list, etc. As the permission system is data-centred, the definition of data should be as precise as possible.

$$\mathcal{O}bjects = \{Phone\#,\ Name,\ Contacts,\ \cdots\} \tag{3}$$

Each application needs to use a piece of personal data to perform a particular action and with a specific goal. Users give the $\mathcal{R}ight$ to the application according to this action and this goal. To define $\mathcal{R}ight$ we have to introduce $\mathcal{A}ction$ and $\mathcal{P}urpose$.

Each action is one of all the actions, denoted $\mathcal{A}ctions$, that can be carried out on user private data by the application: load, read, modify, store and transfer. We define the $\mathcal{A}ctions$ as follows:

$$\mathcal{A}ctions = \{Read, Modify, Load, Store, Transfer\} \tag{4}$$

where

$\mathcal{R}ead$ is a read-only access to the data that is already stored on the phone.

$\mathcal{M}odify$ is an action permitting the replacement or update of a piece of personal data already stored in the system.

$\mathcal{L}oad$ represents an action bringing new information to the phone from a distinct server, Internet or mobile sensor such as GPS, etc.

$\mathcal{S}tore$ action indicates a new piece of private data will be saved on the device.

$\mathcal{T}ransfer$ action indicates some private data is transmitted from the device to the server or another device.

$\mathcal{P}urpose$ is assigned by the application developer and depends on the service. For example, purpose could be 'retrieve forgotten password', 'display on the screen', 'calculate the score', 'send news', 'retrieve nearest restaurant' and 'attach to the message'.

$$\mathcal{P}urpose = \{Retrieve\, forgotten\, password,\ \cdots\} \tag{5}$$

We define a permission right, denoted $\mathcal{R}ight$, for all actions except the $Store$ action as a combination of one element of $\mathcal{A}ctions$ and one element of $\mathcal{P}urposes$.

To respect the minimisation principle, any personal data should be stored on a mobile device only the period of time that is necessary for the functionality. We define the set of rights, denoted $\mathcal{R}ight$, with the action equal to $Store$ having an additional parameter, $time$, informing about the time storage. We define the period $[0, T]$ as an application lifetime.

$$\forall r \in \mathcal{R}ight,\ r = \begin{cases} (action,\ purpose) & if\,(r*) \\ (action,\ purpose,\ time) & if\,(r**) \end{cases} \tag{6}$$

(r*) $action \in \mathcal{A}ctions - \{Store\}$

(r**) $action = Store$

where $purpose \in \mathcal{P}urposes$, $time \in [0, T]$.

The time storage can indicate the number of days, hours or months data is stored or the time regarding the application lifecycle: until the application is closed, until the application is stopped, until the application is uninstalled. All personal data available during the deinstallation of the application is deleted regardless of the defined period, as it cannot exceed the application lifetime.

B. Object: private data

Using existing mobile permissions and mobile forensic techniques on iOS and Android phones we identified some private data that can be accessed on the smartphone by an application. Below is an exhaustive list of personal information that can be found on a mobile phone.

Contact list

1) Type (phone, Facebook, Twitter, Skype) or associated service names
2) Name
3) Surname
4) Nickname
5) E-mail
6) Picture
7) Address
8) Website
9) Company
10) Birthday
11) Job title
12) Significant other

Calendar

1) Calendar name
2) Appointment subject
3) Appointment location
4) Appointment starting date
5) Appointment ending date
6) Appointment starting time
7) Appointment ending time
8) Appointment status
9) Appointment notes
10) Appointment attendees' full names

SMS

1) SMS type (incoming, outgoing)
2) Time
3) Sender's Name
4) Sender's phone number
5) Text content
6) Multimedia content (small images)

MMS

1) MMS type (incoming, outgoing)
2) Time
3) Sender's Name
4) Sender's phone number
5) Text content

6) Multimedia content (images, photos, video, contact information, etc.)

Call logs

1) Call type (incoming, outgoing, missed)
2) Caller's Name
3) Caller's phone number
4) Voice messages

Location

1) Exact location: latitude and longitude
2) Approximate location: latitude and longitude
3) Address
4) Street
5) City
6) Country

Stored Multimedia

1) Type (image, audio, video)
2) Multimedia content
3) META data (date, geolocation)

Other

1) Sensors multimedia (newly created image, audio, video)
2) Mobile phone usage statistics (last used applications, configurations)
3) Device unique id, SIM id
4) Web browser history
5) User accounts information (login, password, tokens)
6) Documents from external or internal storages
7) Currently displayed screen (screen shots)
8) Push messages
9) Bank account information
10) Biometric information
11) Medical records
12) Social networks and other mobile applications' data

We identified 26 Android permissions giving access to personal data including location, accounts, SMS, MMS, camera, audio and video content, mobile user activities, calls, contacts, calendar, saved documents, external storage data and screen shots. Android group Personal info contains only 16 read and write permissions where only 5 permissions give read access to personal data. Other permissions are distributed between Location, Messages and Hardware control groups. One can see that very few permissions exist on Android to protect personal data, compared to the large amount of personal data that can be available on a mobile device.

C. Permission use restrictions

To reduce the flexibility of permission usage by an application and to give more control to the user, we propose to add to each rule several simple restrictions. Each permission is associated with permission restrictions under which the user accepts the permission. Restrictions should be simple for the user to set up.

Each $\mathcal{R}estriction$ contains an action type from the set of action types denoted $\mathcal{A}ctionTypes$. We define two action types: automatic action, denoted $\mathcal{A}utomatic$, and an action that will be explicitly launched by the user, denoted $\mathcal{U}serEvent$

$$\mathcal{A}ctionTypes = \{Automatic, UserEvent\} \quad (7)$$

$\mathcal{U}serEvent$ action is launched via the user interface by the user triggering a particular event. $\mathcal{A}utomatic$ action is launched by an application and can include regular access, update and transfer of the data or automatic insertion of a complementary piece of data (e.g., automatic attachment of the geolocation data to the message, automatically fill in the form, automatically synchronise the data with the server, etc.). Automatic action can be triggered without any action by the user.

$\mathcal{U}serEvent$ restrictions attach the permission to a particular user event. We define $\mathcal{R}estriction$ for the $\mathcal{U}serEvent$ action type as follows:

$$\forall res \in \mathcal{R}estriction,$$
$$res = (rule, action - type, user - trigger - event) \quad (8)$$

where $action - type = UserEvent$, $rule \in \mathcal{R}_{app}$, $action - type = UserEvent$ and $user - trigger - event$ is a concrete user event intercepted by an application.

$\mathcal{A}utomatic$ action can be triggered by the system with a certain frequency or can be associated with a trigger event (e.g., send message, create new message, etc.) or both. We define $\mathcal{R}estriction$ for the $\mathcal{A}utomatic$ action type as follows:

$$\forall res \in \mathcal{R}estriction,$$
$$res = (rule, action - type, frequency, event) \quad (9)$$

where $action - type = Automatic$, $rule \in \mathcal{R}_{app}$, $frequency$ represent the number of times per day/week/month the action can be performed by an application; $event$ is an event associated to the action launch: application is opened, screen is shown, message is sent (button is clicked), form is filled in, etc.

$Frequency$ is not a mandatory parameter. One permission can have several restrictions: it can be associated with several different user or application events.

D. Permission state

Each $rule$ should be explicitly asked of the user to be assigned. Thus, each $rule$ has a $\mathcal{S}tate$: granted or revoked. The rule is granted with corresponding restrictions, the rule is revoked entirely. To respect the Privacy by Default notion the default $\mathcal{S}tate$ of the permission in installed applications is $Revoked$. Only the user can modify the $\mathcal{S}tate$ of permission with an explicit action via the user interface.

We propose to define the $\mathcal{S}tate$ as follows:

$$\mathcal{S}tate(rule, time) = \begin{cases} Granted, & user\,accepts\,the\,rule \\ Revoked, & user\,declines\,the\,rule \end{cases}$$
$$\mathcal{S}tate(rule, 0) = Revoked \quad (10)$$

where $rule \in \mathcal{R}_{app}, time \in]0, T]$.

The $State$ of a rule $r1 \in \mathcal{R}_{app}$ changes over the application lifetime. The diagram in Figure 1 shows an example of state modification.

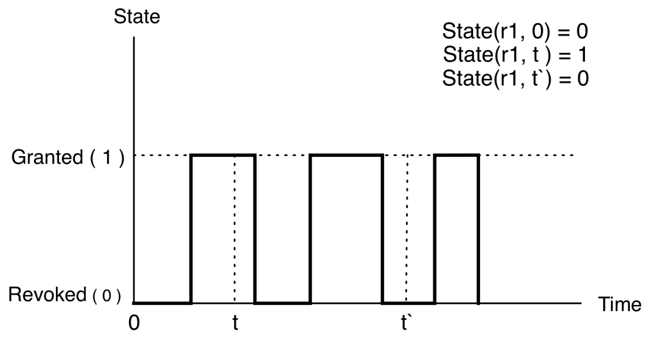

Figure 1. Example of state modification diagram for a given permission

E. User control

User should have a choice of granting the permission permanently, of revoking the permission permanently or of confirming the permission usage with the user each time. We introduce the $rule\ Check$.

$$\mathcal{C}heck(rule, time) = \begin{cases} True, & confirmation\, required \\ False, & no\, confirmation \end{cases}$$
$$\mathcal{C}heck(rule, 0) = True \tag{11}$$

where $rule \in \mathcal{R}_{app}$, $time \in]0, T]$

If the $\mathcal{C}heck$ parameter is set to $True$, than the $\mathcal{S}tate$ passes to $\mathcal{R}evoked$ and is ignored by the system. The permission is granted or revoked each time by the user via the user interface allowing execution of the functionality, thereby the $\mathcal{R}estriction$ of the permissions is also ignored.

If the $\mathcal{C}heck$ parameter is set to $False$, the system verifies the permission $\mathcal{S}tate$ and $\mathcal{R}estriction$ in order to execute the functionality.

To respect the 'Privacy by Default' notion we set the default $\mathcal{C}heck$ parameter to $True$.

F. Permissions interconnection

Each permission is associated with the purpose, thereby each permission is associated with one particular functionality. Several permissions may be needed to assure one functionality and the developer can give the user a choice of using one permission or another. Several permissions can be grouped by functionality in two ways: all permissions are needed to assure the functionality.

We define the $\mathcal{G}roupType$ parameter as follows:

$$GroupType = \{ALL, ONE\} \tag{12}$$

where $\mathcal{A}LL$ shows all permissions are necessary to assure the functionality. $\mathcal{O}NE$ shows only one of the listed permissions is necessary to achieve the functionality.

The ALL parameter can be expressed as a single permission that should be accepted or declined by the user or by one activation button grouping all permissions. To respect the minimisation principle, all permissions linked to a particular functionality should be Revoked if at least one permission was declined by the final user.

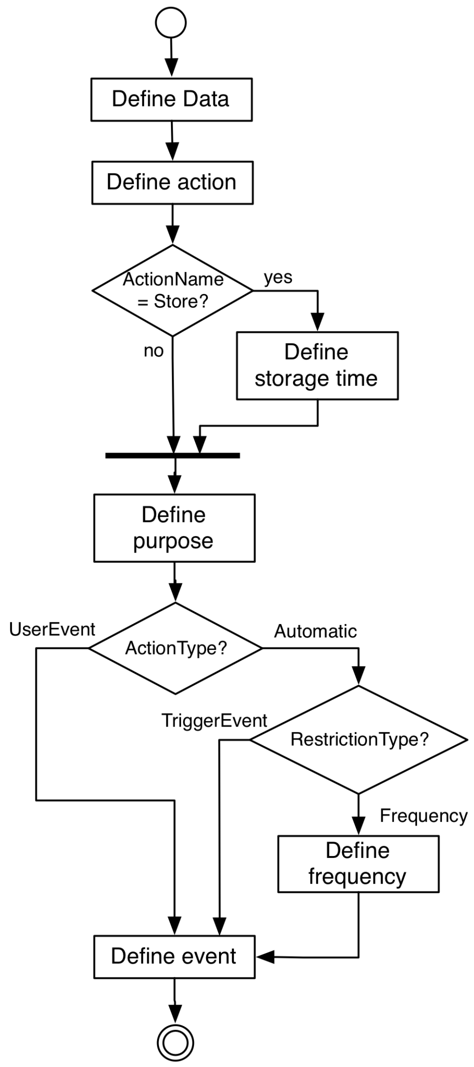

Figure 2. Activity diagram for the rule definition

The ONE parameter can be expressed by the user interface as a group of radio buttons or as one permission with a drop-down list(s) on parameters that differ from permission to permission (e.g., data, usage frequency). If at least one permission is given by the user, the corresponding functionality will be assured and other permissions will be Revoked. For example, an application can assure the service with different types of geolocation data: latitude and longitude, city and the street name and the city only. Developers can propose that the user choose one of the types of geolocation data.

Several permissions can be grouped to add dependencies and an acceptance rule. We define the $\mathcal{G}roup$ parameter as follows:

$$\forall group \in \mathcal{G}roup$$
$$group = (group - type, \{rule_x, rule_y, \cdots\}) \tag{13}$$

where $group - type \in \mathcal{G}roupType$; $rule_x \in \mathcal{R}_{app}$; $rule_y \in \mathcal{R}_{app}$

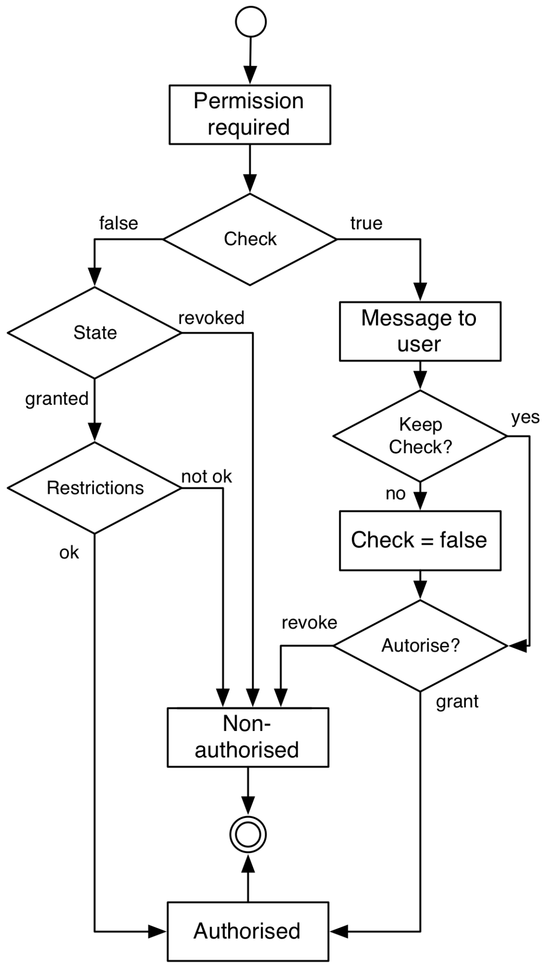

Figure 3. Activity diagram: permission management.

G. Permission in action

The developer should define the permission for all personal data ($\mathcal{O}bject$) used in the application ($\mathcal{S}ubject$) before making the application available on the market. Permission restrictions and permission groupes should also be defined. Figure 2 shows the recapitulative schema of the permission definition.

The permission ($rule$) is stored inside the application with its current $State$ and $Check$ parameters. The default $State$ is $Revoked$. The default $Check$ is $True$. Developers should verify that the permission is fully $displayed$ with the corresponding object, action, purpose and restrictions and $requested$ at least once and that the user is able to grant or revoke this permission. Finally, the user should stipulate the settings with all $rule \in \mathcal{R}_{app}$ to be able to $Grant$ or to $Revoke$ individual permissions in later use.

The simple privacy policy can be generated from the list of defined rules and added to the store.

The activity diagram in Figure 3 shows the permission management cycle from the permission request to the permission usage authorisation/non-authorisation.

When one permission is required by the application, systems first verifies the parameter $Check$. If $Check$ is set to $True$ the system generates the message including all information about the required permission. Users can accept

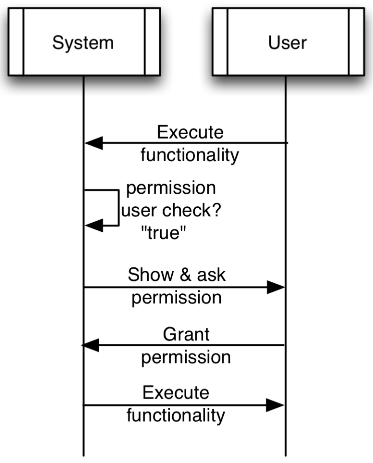

Figure 4. Sequence diagram: first use of one permission or a use or permission in 'user check' mode.

or to decline the permission as well as switch permission management to automatic (Set $Check$ to $False$). According to the user answer the permission is authorised or non-authorised. $Check$ is set to $False$ means an automatic permission management is enabled. System verifies the permission's $Status$. If permission $Status$ is $Revoked$, then the permission is non-authorised. If permission $Status$ is $Granted$, then the systems check corresponding $Restrictions$. If $Restrictions$ are respected, the permission is authorised, otherwise the permission is non-authorised.

The sequence diagram in Figure 4 shows the one case of permission management when the permission is used for the first time or the $Check$ parameter is set to $True$. For example, the user wants to invite a friend to play a game together, the application needs permission to access the list of contacts and full name with emails to send an invitation mail. System verifies the parameter $Check$ that is set to 'true'. System generates the message for the user explaining the permission needed, the user accepts the permission. Now user can choose the friend he wants to invite.

The sequence diagram in Figure 5 shows the patterns in

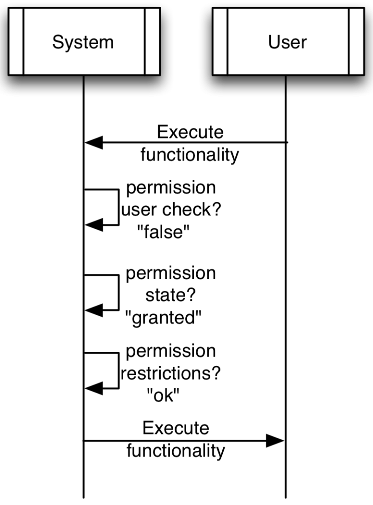
System
User
Execute functionality
permission user check? "false"
permission state? "granted"
permission restrictions? "ok"
Execute functionality

Figure 5. Sequence diagram: use of one permission without user confirmation.

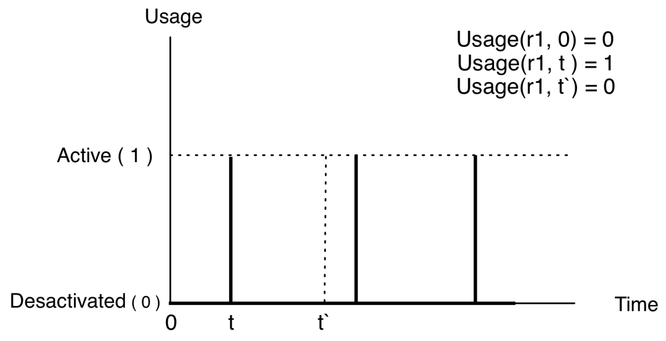

Figure 6. Example of usage modification diagram for a given permission

action when the permission is used automatically without user confirmation: $Check$ parameter is set to $False$. Permission that does not require user confirmation can be used by an application if, and only, the permission status is $\mathcal{G}ranted$ and all $\mathcal{R}estrictions$ are respected. For example, the user wants to automatically attach his geolocation (city) to the first message it sends per day. The permission to load the city from the GPS is needed to attach it to the message. Permission is restricted with a particular event - create new message - and a frequency, once per day. When the user creates a new message, the system verifies the $Status$ of the described permission. As $Status$ is granted, the system verifies the $Restriction$ - permission was not used yet today, therefore, the permission use is authorised.

One can see that the time in which permission can be used by an application is shorter than the time in which permission is Granted. $UserEvent$ action-type permissions can only be used following a particular user event, therefore, permission becomes active only punctually. $Automatic$ permissions are limited by the defined frequency or an event. The usage diagram is depicted in Figure 6

IV. Application

In this section, we propose an example of permission system made for the application of trust evaluation of friends on social networks named Socializer 1.0 [31]. We choose this application because its service is based on private information and cannot be anonymous, the PbD notion should be integrated into this application.

This application needs user friend lists of different social networks (Facebook, Twitter, LinkedIn) and the contact list to view friends and mutual friends to calculate the overlap of friends in different social networks and contact list and to evaluate the trust of Facebook friends.

The following private data can be used by the application:

1) List of contacts from mobile address book: name, surname
2) Facebook friends list: name, surname
3) List of mutual Facebook friends for each Facebook friend: name, surname
4) Twitter friends list: Name, Surname
5) Daily Facebook messages for each Facebook friend
6) Calculated trust score

Contact list is found on the smartphone, therefore, the application needs an $Access$ right.

$$r_1 = (s,\ (Read, p_1),\ ContactList)$$

where s is a $Subject$ defined as the application Socializer 1.0; p_{r1} = calculate the trust scores.

The application will load the user contact list on user event: onClick on the button "load contact list".

$$res_1 = (r_1, UserEvent,\ e_1)$$

where e_1 is a user event: onClick on the button 'load contact list';

Social networking friends lists should usually be retrieved from the server of a given social network, therefore, the load and store actions should be defined. The Facebook friends list with the contact list are essential to assure the overlap and trust functionality.

$$r_2 = (s,\ (Load, p_1),\ FacebookFriendList)$$
$$r_3 = (s,\ (Store, p_1, t_1),\ FacebookFriendList)$$
$$res_2 = (r_2, UserEvent,\ e_2)$$
$$res_3 = (r_3, UserEvent,\ e_2)$$

where t_1 is a storage time defined as: while the application is installed; e_2 is a user event: onClick on the button "load Facebook friends list";

For each Facebook friend, the list of mutual friends with the user is necessary for trust calculation.

$$r_4 = (s,\ p_1),\ FacebookMutualFriendLists)$$
$$res_4 = (r_4, UserEvent,\ e_3)$$

where e_3 is a user event: onItemClick on the user name in the list of users for trust calculation;

A list of friends from other social networks improves the scores of overlap and trust.

$$r_5 = (s,\ (Load, p_2),\ TwitterFriendList)$$
$$r_6 = (s,\ (Store, p_2, t_1),\ TwitterFriendList)$$
$$res_5 = (r_5, UserEvent,\ e_4)$$
$$res_6 = (r_6, UserEvent,\ e_4)$$

where e_4 is a user event: onClick on the button "load Twitter friends list"; p_2 = improve the trust score;

$$r_7 = (s,\ (Load, p_2),\ LinkedInFriendList)$$
$$r_8 = (s,\ (Store, p_2, t_1),\ LinkedInFriendList)$$
$$res_7 = (r_7, UserEvent,\ e_5)$$
$$res_8 = (r_8, UserEvent,\ e_5)$$

where e_5 is a user event: onClick on the button "load LinkedIn friends list";

The second functionality of the application is to evaluate the behaviour of Facebook and Twitter friends to indicate potentially dangerous contacts. The behaviour evaluation is calculated by analysing the messages published by the given friend over time. The application needs a permission to load messages.

$$r_9 = (s,\,(Load, p_2),\,TwitterFriendMessages)$$

$$res_9 = (r_9, Automatic,\,\emptyset,\,e_6)$$

$$res_9 = (r_9, Automatic,\,f_1,\,e_7)$$

where p_3 is a purpose defined as 'calculate the Twitter friends behavior'; f_1 is an action frequency: ones per day; e_6 is a user event: onSlideDown on the list of messages; e_7 is an application event: onApplicationStarted.

$$r_{10} = (s,\,(Load, p_3),\,NewFacebookFriendMessages)$$

$$res_{10} = (r_{10}, Automatic, \emptyset, e_6)$$

$$res_{10'} = (r_{10}, Automatic,\,f_1\,e_7)$$

where p_3 is a purpose defined as 'calculate the Facebook friends behaviour'.

The third functionality proposes to view today Facebook and Twitter messages on the screen for the user.

$$r_{11} = (s,\,(Store, p_4, t_2),\, TodayTwitterFriendMessages)$$

$$res_{11} = (r_{11}, Automatic,\,f_1\,e_7)$$

$$res_{11'} = (r_{11}, Automatic, \emptyset, e_6)$$

where p_4 is a purpose defined as 'view today Twitter messages'; t_2 is a storage time defined as: one day.

$$r_{12} = (s,\,(Store, p_5, t_2),\, TodayFacebookFriendMessages)$$

$$res_{12} = (r_{12}, Automatic,\,f_1,\,e_7)$$

$$res_{12'} = (r_{12}, Automatic, \emptyset, e_6)$$

where p_5 is a purpose defined as 'view today Facebook messages';

The user has the option of sharing the scores by posting new messages on Facebook and Twitter. The user can also contribute to the research by sending the anonymized trust and behaviour statistics to the developer. Those actions should be taken with the user's express consent.

$$r_{13} = (s,\,(Transfer, p_6),\, FacebookFriendTrustScore)$$

$$res_{13} = (r_{13}, UserEvent,\,e_7)$$

where p_6 is a purpose defined as 'share results on Facebook'; e_7 is a user event: onClick on the button 'share'.

$$r_{14} = (s,\,(Transfer, p_7),\, FacebookFriendTrustScore)$$

$$res_{14} = (r_{14}, UserEvent,\,e_7)$$

TABLE II
TABLE RECAPITULATING PERMISSIONS NEEDED FOR THE APPLICATION (LAST COLUMN IS A PERMISSION GROUP NUMBER)

Object	Action	Purpose	#
Contacts list	Read	Calculate Trust	1
Facebook friends list	Load; Store		
Facebook mutual friends	Load		
Twitter friends list	Load; Store	Improve Trust	2
LinkedIn friends list			3
Twitter messages	Load	Tw. friends behaviour	4
Facebook messages		Fb. friends behaviour	5
Today Tw. messages	Store; (1 day)	View Tw. messages	6
Today Fb. messages		View Fb. messages	7
Trust score	Transfer	Publish to Twitter	8
		Publish to Facebook	9
Trust and behaviour	Transfer	Contribute to research	10

where p_7 is a purpose defined as 'share results on Twitter'.

$$r_{15} = (s,\,(Transfer, p_8),\,AnonymizedTrust)$$

$$r_{16} = (s,\,(Transfer, p_8),\,AnonymizedBehavior)$$

$$res_{16} = (r_{16}, UserEvent,\,e_8)$$

where p_8 is a purpose defined as 'contribute to the improvement of the methodology'; e_8 is a user event: onClick on the button 'help research'.

The final application has 16 rules required by the application for full functionality.

$$\mathcal{R}_{app} = \{r_1, r_2, r_3, \cdots, r_{15}, r_{16}\}$$

Those rules can be combined into groups. The rules r_1, r_2, r_3 and r_4 have a common purpose, all rules should be accepted to achieve the functionality mentioned in the purpose: 'calculate the trust'.

$$group_1 = (ALL,\,\{r1, r2, r3, r4\})$$

Similarly, r_5 should be grouped with r_6 and r_7 with r_8.

$$group_2 = (ALL,\,\{r5, r6\})$$

$$group_3 = (ALL,\,\{r7, r8\})$$

The rules from r_9 to r_{16} should be accepted one by one to achieve the aforementioned purpose (to achieve the functionality). Finally, we obtain 10 permissions to be added to the application to propose control to the user. Table II recapitulates permissions.

To compare with actual permission systems, (a) iOS requires contact list, Facebook and Twitter access permissions. (b) Android requires 'internet', 'read_contacts' and 'get_accounts' access permissions. Facebook and Twitter connections are managed with APIs that require permissions to be declared on the platform, but the permission management will not be available for users in the mobile application by default. iOS permissions give a certain transparency to the user but Android permissions are vague.

We obtained more fine-grained control of the application and the data including permissions to all necessary personal data, actions carried out on this data and corresponding purposes. The recapitulation table (Table II) clearly shows what data are used for what purpose. This kind of table can be added to the privacy policy to improve transparency. Restrictions should also be added to the table. We did not add restrictions to the table due to the limited space.

V. Conclusion and Future Work

We modelled a permission system for a mobile application including Privacy by Design. This permission system is data-oriented, thus, the final user can easily understand what personal data is involved. We include actions that are missing from current iOS and Android permission systems, such as load and transfer, that improve transparency of the application. We also included simple restrictions to better control data use.

The novelty is to include the purpose of the data use in the permission system. The clear purpose will help users to understand better why the data is used and to judge whether this permission is needed. Purpose in permission also forces developers to apply the minimization principle: a developer cannot use the data if he cannot define the clear purpose of usage. The compulsory purpose definition should help guard against the abusive permission declaration 'in case'. Finally, purpose gives the user more fine-grained control, as the same data can be allowed to be used for one functionality but not for another. It is important for our system to integrate clear purpose and not a vague explanation (e.g., 'measure the frequency of application utilization' instead of 'improve user experience').

PbD states that the user should have control over his data and be Privacy by Default, therefore, permissions used in the application are revoked by default. Users should be clearly informed and asked to grant permission. Moreover, users should keep control of permissions during all the application use time, therefore, the permission setting must be available.

Our permission system helps developers to be compliant with the law; it defines what permissions the developer should add to the application, but in the current state it cannot ensure that all necessary permissions are really added. Our pattern indicates to the developer what should be added to the application to be more transparent, but if he decides to transfer data without asking permission, the pattern allows this (even if it is against European law). The privacy policy generated can give the first indication permitting evaluation if the data use is reasonable and the purpose is clear. Manual verification of an application can show the anomaly in permission system use.

We aim to build a framework for the automatic management of a new permission system to simplify the developers' work. We target the Android system first as it is more crucial due to the more open communication and data sharing, and the vagueness of the current permission system.

The impact of new privacy-respective permission systems on users and developers could be measured by conducting real-life experiments. We aim to measure the impact of integration of the new permission system on design and development time, as well as particular situations and difficulties in applying the pattern. We have an additional hypothesis that the explicative application with high transparency improves user experience and leads to more positive perception of the same application, therefore the use of our permission system benefits the application owner.

References

[1] K. Sokolova, M. Lemercier, and J. B. Boisseau, "Privacy by Design Permission System for Mobile Applications," in PATTERNS 2014, The Sixth International Conferences on Pervasive Patterns and Applications, 2014, pp. 89–95.

[2] TRUSTe. Consumer Mobile Privacy Insights Report. [retrieved: Apr., 2011]

[3] A. Cavoukian, "Privacy by design: The 7 foundational principles," 2009.

[4] E. data protection regulators, "Opinion 02/2013 on apps on smart devices," EU, Tech. Rep., Feb. 2013.

[5] K. D. Harris, "Privacy on the go," California Department of Justice, Jan. 2013, pp. 1–27.

[6] W. Shin, S. Kiyomoto, K. Fukushima, and T. Tanaka, "Towards Formal Analysis of the Permission-Based Security Model for Android," in Wireless and Mobile Communications, 2009. ICWMC '09. Fifth International Conference on. IEEE Computer Society, 2009, pp. 87–92.

[7] K. W. Y. Au, Y. F. Zhou, Z. Huang, P. Gill, and D. Lie, "Short paper: a look at smartphone permission models," in SPSM '11 Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices. ACM, Oct. 2011, pp. 63–67.

[8] R. Stevens, J. Ganz, V. Filkov, P. T. Devanbu, and H. Chen, "Asking for (and about) permissions used by Android apps," MSR, 2013, pp. 31–40.

[9] A. R. Beresford, A. Rice, N. Skehin, and R. Sohan, "MockDroid: trading privacy for application functionality on smartphones," in HotMobile '11: Proceedings of the 12th Workshop on Mobile Computing Systems and Applications, ser. HotMobile '11. ACM, Mar. 2011, pp. 49–54.

[10] P. Hornyack, S. Han, J. Jung, S. Schechter, and D. Wetherall, "These aren't the droids you're looking for: retrofitting android to protect data from imperious applications," in Proceedings of the 18th ACM conference on Computer and communications security, ser. CCS '11. ACM, Oct. 2011, pp. 639–652.

[11] M. Nauman, S. Khan, and X. Zhang, "Apex: extending Android permission model and enforcement with user-defined runtime constraints," in ASIACCS '10: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security. ACM, Apr. 2010, pp. 328–332.

[12] Y. Zhou, X. Zhang, X. Jiang, and V. W. Freeh, "Taming Information-Stealing Smartphone Applications (on Android)," in Trust and Trustworthy Computing. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 93–107.

[13] S. Holavanalli et al., "Flow Permissions for Android," in Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on, 2013, pp. 652–657.

[14] J. Tam, R. W. Reeder, and S. Schechter, "Disclosing the authority applications demand of users as a condition of installation," Microsoft Research, 2010.

[15] S. Egelman, A. P. Felt, and D. Wagner, "Choice Architecture and Smartphone Privacy: There's a Price for That," in Proceedings of the 11th Annual Workshop on the Economics of Information Security (WEIS), 2012.

[16] M. Lane, "Does the android permission system provide adequate information privacy protection for end-users of mobile apps?" in 10th Australian Information Security Management Conference, Dec. 2012, pp. 65–73.

[17] P. G. Kelley et al., "A conundrum of permissions: Installing applications on an android smartphone," in Proceedings of the 16th International Conference on Financial Cryptography and Data Security, ser. FC'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 68–79.

[18] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, "Android permissions demystified," in CCS '11: Proceedings of the 18th ACM conference on Computer and communications security. ACM, Oct. 2011, pp. 627–638.

[19] M. Egele, C. Kruegel, E. Kirda, and G. Vigna, "PiOS: Detecting Privacy Leaks in iOS Applications." 2011.

[20] C. Gibler, J. Crussell, J. Erickson, and H. Chen, "AndroidLeaks: Automatically Detecting Potential Privacy Leaks in Android Applications on a Large Scale," in Trust and Trustworthy Computing. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 291–307.

[21] T. Vidas, N. Christin, and L. Cranor, "Curbing android permission creep," in In W2SP, 2011.

[22] Y. Zhang et al., "Vetting undesirable behaviors in android apps with permission use analysis," in CCS '13: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. ACM, Nov. 2013, pp. 611–622.

[23] W. Xu, F. Zhang, and S. Zhu, "Permlyzer: Analyzing permission usage in Android applications," Software Reliability Engineering (ISSRE), 2013 IEEE 24th International Symposium on, 2013, pp. 400–410.

[24] W. Luo, S. Xu, and X. Jiang, "Real-time detection and prevention of android SMS permission abuses," in SESP '13: Proceedings of the first international workshop on Security in embedded systems and smartphones. ACM, May 2013, pp. 11–18.

[25] J. Vincent, C. Porquet, M. Borsali, and H. Leboulanger, "Privacy Protection for Smartphones: An Ontology-Based Firewall," in Information Security Theory and Practice. Security and Privacy of Mobile Devices in Wireless Communication. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 371–380.

[26] S. Bugiel, S. Heuser, and A.-R. Sadeghi, "mytunes: Semantically linked and user-centric fine-grained privacy control on android," Center for Advanced Security Research Darmstadt, Tech. Rep. TUD-CS-2012-0226, Nov. 2012.

[27] A. P. Felt, K. Greenwood, and D. Wagner, "The effectiveness of application permissions," in WebApps'11: Proceedings of the 2nd USENIX conference on Web application development. USENIX Association, Jun. 2011, pp. 7–7.

[28] R. Böhme and S. Köpsell, "Trained to accept?: a field experiment on consent dialogs," in CHI '10: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, Apr. 2010, pp. 2403–2406.

[29] E. Smith, "iPhone applications and privacy issues: An analysis of application transmission of iPhone unique device identifiers UDIDs," October 2010.

[30] "Mobile Apps Study," Future of Privacy Forum (FPF), pp. 1–16, Jun. 2012.

[31] C. Perez, B. Birregah, and M. Lemercier, "A smartphone-based online social network trust evaluation system," Social Network Analysis and Mining, vol. 3, no. 4, 2013, pp. 1293–1310.