Big Brother is Smart Watching You
Privacy Concerns about Health and Fitness Applications

Christoph Stach
[1]*Institute for Parallel and Distributed Systems, University of Stuttgart,*
Universitätsstraße 38, D-70569 Stuttgart, Germany
stachch@ipvs.uni-stuttgart.de

Keywords:
Smartbands, Health and Fitness Applications, Privacy Concerns, Privacy Management Platform.

Abstract:
Health and fitness applications for mobile devices are becoming more and more popular. Due to novel wearable metering devices, the so-called *Smartbands*, these applications are able to capture both health data (e. g., the heart rate) as well as personal information (e. g., location data) and create a *quantified self* for their users. However, many of these applications violate the user's privacy and misuse the collected data. It becomes apparent that this threat is inherent in the privacy systems implemented in mobile platforms. Therefore, we apply the **P**rivacy **P**olicy **M**odel (*PPM*) a fine-grained and modular expandable permission model to deals with this problem. We implement our adapted model in a prototype based on the **P**rivacy **M**anagement **P**latform (*PMP*). Subsequently, we evaluate our model with the help the prototype and demonstrate its applicability for any application using *Smartbands* for its data acquisition.

1 INTRODUCTION

With the *Internet of Things*, the technology landscape changed sustainably. New devices with various sensors and sufficient processing power to execute small applications, so-called *Smart Devices*, captured the market. Due to their capability to interconnect with each other via energy efficient technologies such as Bluetooth LE, they are able to share their sensor data with other devices over a long period of time. As a consequence of this accumulation of data from highly diverse domains (e. g., location data, health data, or activity data), new types of devices as well as novel use cases for pervasive applications emerge. In the consumer market, especially *Smartbands*, i. e., hardware devices equipped with GPS and a heartbeat sensor among others which are carried on the wrist, are currently very popular. Such devices are controlled via Smartphone applications and provide the recorded data to these applications. The applications analyze the data, augment it with additional knowledge about the user which is stored on the Smartphone, and gain further insights from it. Since the *Smartbands* are small and comfortable to wear, they do not interfere their users' activities in any way. So, they can be kept on even when doing sports or while sleeping.

That is why innovative fitness applications come up which make use of *Smartbands*. Such applications are able, e. g., to capture the user's movement patterns in order to determine his or her current activity (Knighten et al., 2015), track him or her to calculate the traveled distance, his or her speed as well as the nature of the route (this data can be used to calculate the calorie consumption) (Wijaya et al., 2014), and even analyze his or her sleeping behavior (Pombo and Garcia, 2016). This data is processed on the Smartphone and visualized in a user-friendly manner. However, in order to achieve a *quantified self*, i. e., a comprehensive mapping of our lifestyle to quantifiable values to assess our daily routines, this data is sent to a central storage where it is further analyzed and provided for other stakeholders, such as physicians (Khorakhun and Bhatti, 2015).

While this technology has the potential to radically modify the quality of human life as an unhealthy lifestyle or looming up diseases can be detect at an early stage, it also constitutes a threat towards the user's privacy. As *Smartbands* collect

a lot of sensitive data, an attacker could get insights into a user's daily routines or even his or her health status.

Thus, there is a lot research done concerning the vulnerability of the involved devices (*Smartband*, Smartphone, and back-end) or the data transfer channels in between (Lee et al., 2016). Due to thereby detected vulnerabilities, the Federal Trade Commission proposed a catalog of measures of how to provide security for these devices (Mayfield and Jagielski, 2015).

However, all of these efforts target attacks from the outside. Since there are a lot of stakeholders interested in this kind data, including insurance companies or the advertisement industry, the data becomes highly valuable (Funk, 2015) and a lot of "free" applications sell the collected data to third parties (Leontiadis et al., 2012). This brings up a completely different problem: How can the user be informed about the data usage of an application and how can s/he be enabled to control the data access privileges of an application as well as anonymize his or her data before providing it to an application (Patel, 2015)?

To this end, this paper yields the following contributions:

- We analyze the state of the art as well as research projects concerning the protection of private data in the context of *Smartband* applications.
- We adapt a privacy policy model which enables users to control the data usage of *Smartband* applications in a fine-grained manner. Our approach is based on the **P**rivacy **M**anagement **P**latform (*PMP*) and its **P**rivacy **P**olicy **M**odel (*PPM*) (Stach and Mitschang, 2013, Stach and Mitschang, 2014).
- We introduce a prototypical implementation of a privacy mechanism for *Smartband* applications using our privacy policy model.
- We evaluate our approach and demonstrate its applicability.

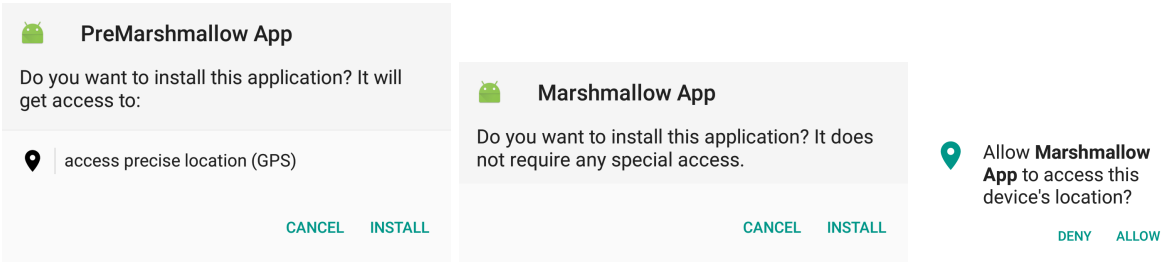
The remainder of the paper is structured as follows. In Section 2, the privacy control mechanisms of the currently prevailing mobile platforms (namely Apple's iOS and Google's Android) are discussed, and the prevailing connection standard Bluetooth LE is characterized. Section 3 looks at some related work, that is enhanced privacy control mechanisms for mobile platforms. Our approach for such a mechanism specifically for *Smartbands* and similar devices is introduced in Section 4. Following this, our generic concept is implemented using the *PMP* in Section 5. Section 6 evaluates our approach and reveals whether it fulfills the requirements towards such a privacy control mechanism. Finally, Section 7 concludes this paper and glances at future work.

2 STATE OF THE ART

In the following, we explain, why especially the usage of applications for *Smartbands* and similar *Smart Devices* such as health or fitness applications constitutes a real threat to privacy. To this end, it is necessary to look at the privacy mechanisms implemented in mobile platforms as well as the modus operandi of how to connect a *Smartband* with a Smartphone.

Privacy Mechanisms in Current Mobile Platforms. Every relevant mobile platform applies some kind of permission system to protect sensitive data (Felt et al., 2012a). This means in effect, that every application has to declare which data is processed by it. The system validates for each data access whether the permission can be granted. A permission refers not to a certain dataset but to a sensor or a potentially dangerous system functionality (Barrera et al., 2010). This concept is implemented divergently in every mobile platform. An iOS application requires Apple's approval prior to its release. In this process, automated and manual verification methods check whether the permission requests are justified. If the permissions are granted by Apple, the application is signed and released. The user is only informed about permissions which affect his or her personal information (e. g., the contacts). On the contrary, Google does not engage in the permission process at all (Enck et al., 2009). When an Android application is installed, the user is informed about every requested permission and has to grant all of them in order to be able to proceed with the installation (Barrera and Van Oorschot, 2011). With Android 6.0 *Runtime Permissions* are introduced. A *Runtime Permission* is not requested at installation time, but each time data is accessed, which is protected by the respective permission.

However, studies prove, that users are unable to cope with the huge amount of different permissions—especially since they are not able to understand which consequences the granting of a certain permission has (Felt et al., 2012b). This is why Google divide Android's permissions into



(a) Android 5.1 Installation Dialog (b) Android 6.0 Installation Dialog (c) Request at Runtime

Figure 1: Permission Requests in Different Android Versions.

two classes since Android 6.0: *Normal Permissions* require no longer the user's approval. Only *Dangerous Permissions* (which are a superset of the *Runtime Permissions*) have to be granted. For instance, the ACCESS_FINE_LOCATION (access to the GPS) or BODY_SENSORS (access to heart rate data) permission belong to this category. Yet, the BLUETOOTH and INTERNET permission are classified as *Normal Permissions*. Figure 1 depicts the consequences of these changes. An application which needs to access GPS data, discover, pair, and connect to Bluetooth devices, as well as open network sockets has to declare the following four permissions in its Manifest: ACCESS_FINE_LOCATION, BLUETOOTH, BLUETOOTH_ADMIN, and INTERNET. On devices running a pre-Marshmallow Android version (< 6.0), the user has to grant all permissions at installation time. The installation dialog however informs him or her about the *Dangerous Permissions*, only (see Figure 1a). On devices with a higher Android version, even the *Runtime Permissions* are hidden (see Figure 1b). Instead, s/he has to grant the permission every time the application tries to get access to GPS data (see Figure 1c). In any case, the user is not aware of the fact, that the application is also able to send this data to any Bluetooth device or the Internet.

Transmission Standard of Smart Devices. Most of the current *Smart Devices* use Bluetooth LE to connect to each other as it has a lower power consumption than Classic Bluetooth and a higher connectivity range than NFC. The device vendor defines UUIDs via which other devices are able to request the device's services. For instance, a service of a *Smartband* could provide access to the heart rate. The vendor also specifies how the data is encoded by his or her device. As a consequence, a mobile platform cannot determine what data is transferred between a Smartphone and an other *Smart Device*, since it does not know the mapping of the UUIDs to services and also cannot look into the transmitted data. This is why the permissions only refers to the general usage of Bluetooth connections and not to the type of data which is transmitted. The same holds for the onward transmission of the data to a server. Here too, an application only has to indicate that it needs access to the Internet, but the user neither is aware of what kind of data the applications sends out nor where the data is sent to.

Assuming that a *Smartband* has a built-in GPS and heart rate senor. Therefore, it is able to provide both, location and health data to applications. The application only needs the permission to discover, pair, and connect to Bluetooth devices (BLUETOOTH and BLUETOOTH_ADMIN) to that end. Yet, both permissions belong to the *Normal Permissions* category, i. e., the system automatically grants the permission and the user is not informed about it. If the application wants to access the very same data from the Smartphone directly, the permissions ACCESS_FINE_LOCATION and BODY_SENSORS are required. Both of them belong to the *Dangerous Permissions* category, i. e., the user has to grant every access at runtime. This classification is reasonable as the covered data reveals a lot of private information about the user. The usage of a *Smartband* bypasses this protective measure completely. Moreover, the application is able to share this information with any external sink without the user's knowledge. It only has to declare the INTERNET permission in its Manifest—also a *Normal Permissions*. Therefore, a static permission-based privacy mechanism as implemented in current mobile platforms is inapplicable for health or fitness applications using *Smartbands*.

As Android assigns the responsibility over sensitive data to the user, a security vulnerability such as the careless handling of data interchanges with *Smartbands*, causes the most serious conse-

quences. Thus, the remainder of this paper focuses on Android. However, the insights and concepts are applicable to any other mobile platform.

3 RELATED WORK

As the prevailing mobile platforms provide no adequate protection for sensitive data, there are a lot of research projects dealing with better privacy mechanisms for these platforms. In the following, we present a representative sample of these approaches and determine to what extent they are applicable for *Smartband* applications.

Apex (Nauman et al., 2010) enables the user to add contextual conditions to each Android permission. These conditions specify situations in which a permission is granted. For instance, the user can set a timeframe in which an application gets access to private data or define a maximum number of times a certain data access is allowed. If the condition is not kept, a `SecurityException` is raised and the application crashes. Furthermore, as *Apex* is based on the existing Android permissions, it is too coarse-grained for the *Smartband* use case.

AppFence (Hornyack et al., 2011) analyzes the internal dataflow of applications. When data from a privacy critical source (e. g., the camera or the microphone) is sent to the Internet, the user gets informed. S/he is then able to alter the data before it is sent out or s/he can enable the flight mode whenever the affected application is started. However, *AppFence* does not knows which data an applications reads from a Bluetooth source. Thereby, it cannot differentiate whether an applications accesses trivial data from headphones (e. g., the name of the manufacturer) or private data from a *Smartband* (e. g., health data). Moreover, *AppFence* cannot identify to which address the data is sent to.

Aurasium (Xu et al., 2012) introduces an additional sandbox which is injected into every application. This has to be done before the application is installed. The sandbox monitors its embedded application and intercepts each access to system functions. Thereby, *Aurasium* is not limited to the permissions predefined by Android. Especially for the access to the Internet, *Aurasium* introduces fine-grained configuration options, e. g., to specify to which servers the application may send data to. For every other permission, the user can simply decide whether s/he wants to grant or deny it. Moreover, *Aurasium* is not extensible. That is, it cannot react to new access modes as introduced by *Smartbands* where several data types can be accessed with the same permission. Also, the bytecode injection which is required for every application is costly and violates copyright and related rights.

Data-Sluice (Saracino et al., 2016) considers solely the problem of uncontrolled data transfer to external sinks. Therefore, *Data-Sluice* monitors the any kind of network activities. As soon as an applications attempts to open a network socket, the user is informed and s/he can decide whether the access should be allowed or denied. Additionally, *Data-Sluice* logs every network access and is able to blacklist certain addresses. However, the user is neither informed about which data is sent to the network nor is s/he able to limit the data access of an application from any other source, except for the Internet.

I-ARM Droid (Davis et al., 2012) is the most comprehensive approach. The user defines critical code blocks (i. e., a sequence of commands that accesses or processes private data) and specifies rewriting rules for each of them. A generic converter realizes the rewriting at bytecode level. However, this approach is much too complex for common users. As a consequence, its derivative *RetroSkeleton* (Davis and Chen, 2013) assigns this task to a security expert who creates a configuration according to the user's demands. Because of this, frequent changes of the privacy rules are not possible—not to mention rule adjustments at runtime. Additionally, the expert has to know each conceivable code block that could violate the user's privacy. In other words s/he has to know every available *Smartband*, as each vendor defines a specific communication protocol.

4 A PERMISSION MODEL FOR SMARTBANDS

None of the privacy mechanisms mentioned in the foregoing section is applicable for health or fitness applications using *Smartbands* due to too coarse-grained permissions and missing modular expandability in order to support novel device or data types. The **P**rivacy **M**anagement **P**latform (*PMP*) (Stach and Mitschang, 2013, Stach and Mitschang, 2014, Stach, 2015) provides these features. Furthermore, the *PMP* facilitates the connection of *Smart Devices* to Smartphones (Stach et al., 2017b).

Therefore, we extend the *PMP* by two additional components, the *Smartband Resource Group*

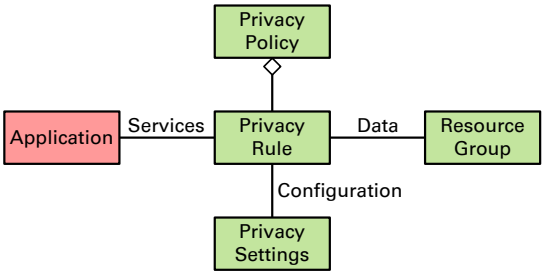


Figure 2: Simplified Representation of the *Privacy Policy Model*; Untrusted Components are Shaded Red and Trusted Components are Shaded Green (cf. (Stach and Mitschang, 2013)).

and the *Internet Resource Group*. With these two components the *PMP* enables users to provide the data from *Smartbands* to applications in a privacy-aware manner and also restricts the transmission of sensitive data to the Internet.

For that purpose, we characterize the **P**rivacy **P**olicy **M**odel (*PPM*), which is the foundation of the *PMP*, and describe how we adapt it to the *Smartband* setting (Section 4.1). Then we outline the operating principle of the *PMP* (Section 4.2). Lastly we introduce the concept of our two extensions (Section 4.3 and Section 4.4).

4.1 The Privacy Policy Model

The *PPM* interrelates applications with data sources or system functions (labeled as *Resource Groups*). An application describes its *Features* and specifies which data or system functions are required for their execution. A *Resource Group* defines an interface via which an application can access its data or execute certain system functions. The user declares in *Privacy Rules* which of an application's *Features* should be deactivated in order to reduce its access to data or system functions. Moreover, s/he can refine each *Privacy Rule* by adding *Privacy Settings*, e. g., to downgrade the accuracy of a *Resource Group's* data. The set of all *Privacy Rules* forms the *Privacy Policy*. The model assumes that applications are untrusted components, while *Resource Groups* are provided by trustworthy parties. The *PPM* is shown in Figure 2 as a simplified UML-like class diagram. For more information on the *PPM*, please refer to the literature (Stach and Mitschang, 2013).

In the context of this work, only the *Resource Groups* are of interest. Figure 3 gives an insight into the architecture of a *Resource Group*. Each *Resource Group* defines an interface (`IResource`) and descriptors, which *Privacy Settings* can be

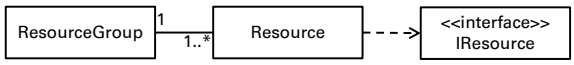


Figure 3: Architecture of *Resource Groups*.

applied to it. The actual implementation of the defined functions is given in *Resources*. A single *Resource Group* can bundle many *Resources*, i. e., many alternative implementation variants for the interface. For instance, a *Resource Group Location* could provide a single method to query the users current location. This method is implemented in two varying kinds, once using the GPS and once using the Cell-ID. Depending on the available hardware, the user's settings, etc., the *Resource Group* selects the proper *Resource* when an application requests the data. In addition to it, the *Location Resource Group* could define a *Privacy Settings Accuracy* via which the user defines how accurate the location data is, that is up to how many meters the actual location should differ from the provided location in order to restrict the application's data access. Naturally, s/he is also able to prohibit the access to the *Resource Group* completely for a certain application.

4.2 The Privacy Management Platform

The *PMP* is a privacy mechanism which implements the *PPM*. Due to the model's features mentioned above, the *PMP* has two characteristics which are highly beneficial for the work at hand: (a) On the one hand, the *PMP* is **expandable by modules**. That is, further *Resource Groups* as well as *Resources* can be added at runtime. That way recent device models (by adding *Resources*) and even completely new kinds of devices or sensors (by adding *Resource Groups*) can be supported. (b) On the other hand, the *PMP* supports a **fine-grained access control**. Each *Resource Group* defines its own *Privacy Settings*. These settings are meeting the demands of the corresponding device. So, a user is not just able to turn a device or sensor on and off in order to protect his or her private data, but s/he can also add numeric or textual restrictions. For instance, a *Resource Group* for location information can have a numeric *Privacy Setting* via which the accuracy of the location data can be reduced, or an Internet *Resource Group* can use a textual *Privacy Setting* to specify to which addresses an application is allowed to send data to.

To accomplish these objectives, the *PMP* is an intermediate layer between the application layer

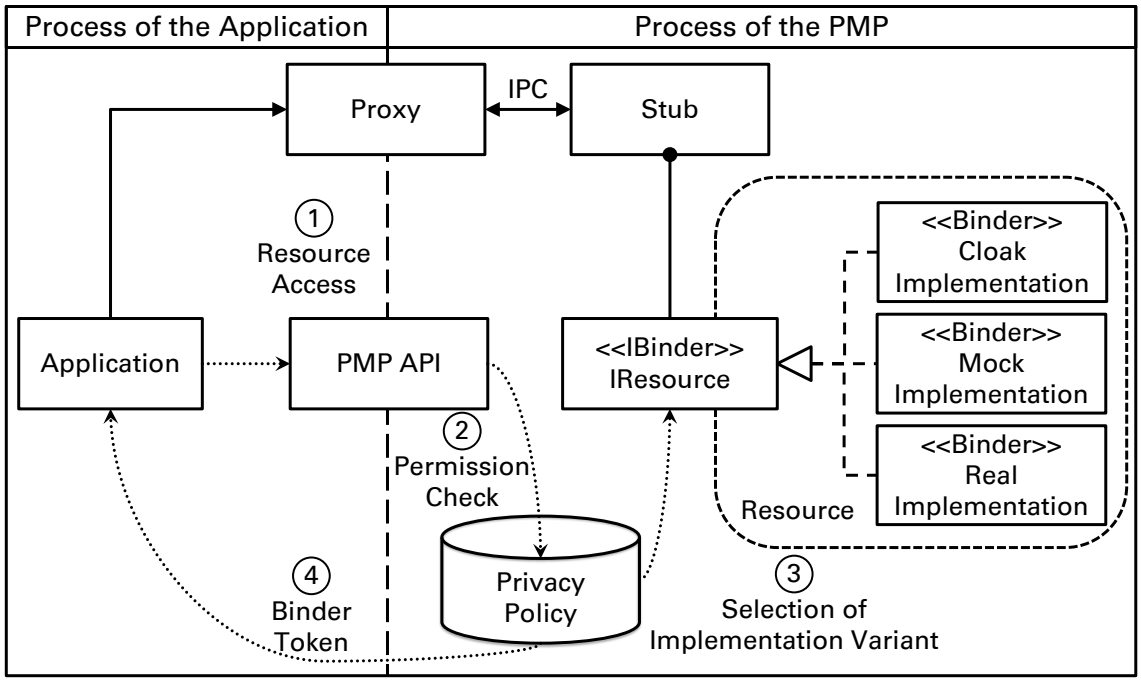


Figure 4: Simplified Implementation Model of the *Privacy Management Platform*.

and the actual application platform. For simplification purposes, in the context of this work the *PMP* can be seen as an interface to the application platform. Figure 4 shows the implementation model of the *PMP* in a simplified representation. Initially, an application requests access to data sources or system functions—i.e., to a *Resource Group*—via the *PMP API* (1). The *PMP* checks, whether this request complies with the *Privacy Rules* in the *Privacy Policy* (2). These rules also specify which restrictions (*Privacy Setting*) apply for the respective application. If the access is granted, a fitting *Resource* within the requested *Resource Group* is selected (3). For each *Resource*, the *PMP* also offers two fake implementation (*Cloak Implementation* and *Mock Implementation*) with highly anonymized or completely randomized data. The proper implementation of the selected *Resource* is then bond as a *Binder*[1] to the *IBinder* interface and the *PMP* forwards the corresponding *Binder Token* to the inquiring application (4).

The actual access to a *Resource* is realized by the Android *Binder Framework*. *Proxy* components specified therein use interprocess communication (*IPC*) to interchange data with the *Stub* bond to the implementation of the *Resource*. An application cannot access a *Resource* directly without the appropriate *Binder Token*. Thereby it is assured that every data request has to be made via the *PMP* and as a consequence the *PMP* is able to verify for every request whether it complies to the *Privacy Policy*. As all *Resource Groups* are implemented as subpackages of the *PMP* and run in the same process, they are executed in a shared sandbox. That way, the *PMP* is able to interact with *Resource Groups* directly.

Due to these features, we are able to use the *PMP* for our approach of a privacy mechanism for *Smartband* applications. Essentially, two additional *Resource Groups* are required to achieve this goal: A *Resource Group* for *Smartbands* which restricts the access to the diverse data types of these devices and a second *Resource Group* which restricts the data transfer of *Smartband* applications to the Internet. The specifications for these *Resource Groups* are given in the following.

4.3 Smartband Resource Group

The *Smartband Resource Group* has to provide a uniform interface to any *Smartband* model (including *Smart Watches* and related devices). Therefore, the interface is composed as a superset of data access operations which are supported by most of such devices. This includes access to personal data (e. g., age or name), health-related data

[1]see https://developer.android.com/reference/android/os/Binder.html

(e. g., heart rate or blood sugar level), activity-related data (e. g., acceleration or orientation), and location data. In addition to these reading operations, most *Smartband* also have a small display to show short notifications. So, the *Smartband Resource Group* also defines a writing operation to send messages to this display. However, not every *Smartband* model supports each of these operations. The *Resources* implementing the operations for the particular *Smartbands* have to deal with this problem. They throw an `UnsupportedOperationException` which is caught and handled by the *PMP* (e. g., by passing mock data to the application).

To restrict access to data provided by *Smartbands*, the *Smartband Resource Group* defines several fine-grained *Privacy Settings*. Basically, there is a two-valued *Privacy Setting* for each data type via which the particular data access has to be granted or denied. In this way, the user is able to decide which application is allowed to access which data from his or her *Smartband*. As mentioned above, this is already a major advance in comparison with state of the art, since Android supports only a single Bluetooth permission for any kind of device and data—let alone the fact that users cannot see whether an application requests this permission at all! Furthermore, the *Smartband Resource Group* supports for certain data types additional *Privacy Settings* (e. g., the accuracy of location data can be reduced). Moreover, each data source in the *Smartband Resource Group* can be replaced by a mock implementation. All mock values are within a realistic range so that applications cannot observe a difference.

Moreover, *Smartbands* providing access to location data can be integrated into the existing *Location Resource Group* (see (Stach, 2013)) as additional *Resources*. So, the *PMP* is able to switch between the available *Resources* when needed (e. g., in case the *Smartband's* location data is more precise than the one from the Smartphone).

4.4 Internet Resource Group

The *Internet Resource Group* provides a simplified interface to send data to and receive data from a network resource. Essentially, both functions have two parameters, an address of the destination device and the actual payload. The payload is also used to store the response from the network resource. This simplification of the interface is adequate in the context of *Smartband* applications. In order to support applications requiring extensive interactions with network resources, this interface can be extended by more generic I / O functions (e. g., to support several network protocols).

Analogously to the *Smartband Resource Group*, also the *Internet Resource Group* defines two-valued *Privacy Settings* for both I / O functions. Thereby the user is able to specify for each application separately whether it is allowed to send data to and / or receive data from the Internet. In addition to it, also the admissible destination addresses is restrictable. In theory, it is possible to realize this by a textual *Privacy Setting* via which the user is able to declare admissible addresses. However, the user's attention is a finite resource and such a fine-grained address-wise restriction overstrains him or her (Böhme and Grossklags, 2011). On this account, the *Internet Resource Group* categorizes addresses by various domains, such as the health domain or a domain for location-related information. The category "public" does not restrict the admissible destination addresses at all. In this way, the user is able to comprehend which domain is reasonable for a certain kind of application. However, the *Resource Group* can also be extended by such textual *Privacy Settings* described above if required—e. g., expert users might ask for a more fine-grained access control.

5 PROTOTYPICAL IMPLEMENTATION

In order to verify the applicability of our approach, we implemented a simple fitness application in addition to the two *Resource Groups*. The fitness application creates a local user profile, including inter alia his or her age, height, and weight. Whenever the user works out, data from the *Smartband's* motion sensors (e. g., to determine his or her activity) and health data (e. g., his or her heart rate) is captured. This data is augmented by location data from the *Smartband* to track the user's favorite workout locations. To share this data with others (e. g., with an insurance company to document a healthy lifestyle) or to create a *quantified self*, this data can be uploaded to an online account.

To this end, the fitness application defines five *Features* which can be individually deactivated by the *PMP*. When the application is installed, the *PMP* lists all of these *Features* and the user can make a selection (see Figure 5a). For instance, a user might want to use the application to capture his or her workout progresses in a local profile, but the application should not track the workout

```
interface SmartbandResource {
   // access to personal data
   int getAge();
   ...
   // access to workout data
   int getHeartRate();
   ...
   // access to location data
   Location getLocation();
   ...
}
```

Listing 1: Interface Definition for the *Smartband Resource Group* in AIDL (excerpt).

locations or upload the profile to a server. This selection characterizes which service quality the user expects from the application. In order to find out which permissions are required for each *Feature*, the *PMP* can show additional information.

Applications access data via the interface of the respective *Resource Group*. This interface is described in the Android Interface Definition Language (AIDL). Listing 1 shows such a definition for the *Smartband Resource Group* in excerpts[2].

Beyond that, the user is also able to adjust *Privacy Rules* from a *Resource Group's* point of view. To that end, all *Resource Groups* requested by the particular application are listed together with the therein defined *Privacy Settings* (see Figure 5b). Two-valued *Privacy Settings* such as "Send Data" can be directly turned on and off simply by clicking on them. For textual and numerical *Privacy Settings* such as "Location Accuracy" the user can enter new values in an input mask with a text box. Enumeration *Privacy Settings* such as "Admissible Destination Address" open an input mask with a selection box (see Figure 5c). If the selected *Privacy Settings* are too restrictive for a certain *Feature*, the *PMP* deactivates the *Feature* and informs the user.

The *Privacy Settings* are defined within a so-called *Resource Group Information Set* (*RGIS*) in XML. Similar to Android's *App Manifest* this file contains metadata required by the *PMP* about the *Resource Group*. Listing 2 shows an excerpt of the *Privacy Settings* definition in the *RGIS* for the *Internet Resource Group*. As seen in the listing, each *Privacy Setting* mainly consists of a unique identifier, a valid value range, and a human-readable description. The *PMP* derives its configuration dialogs such as the *Privacy Settings* dialog (see Figure 5b) from the *RGIS*.

[2]The data type `Location` is not supported by AIDL. Additional type definition files are required.

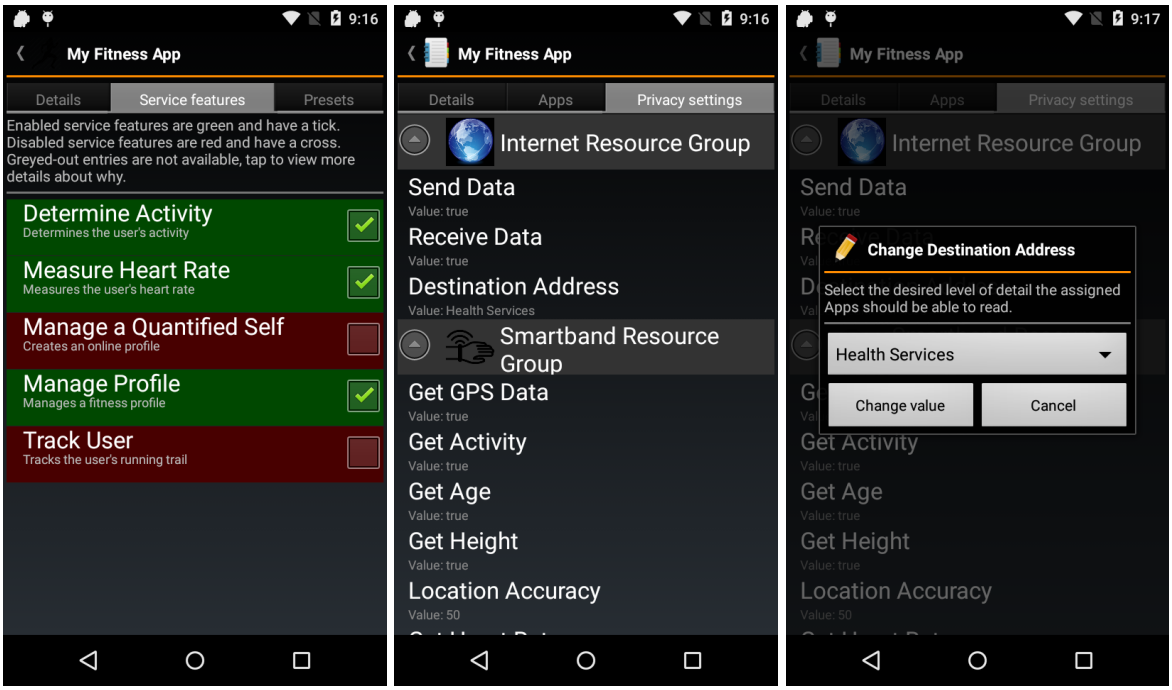
```
<?xml version="1.0" encoding="UTF-8"?>

<resourceGroupInformationSet>
  <resourceGroupInformation
   ↪ identifier="internet">
    <name>Internet</name>
    <description>Manages any network
     ↪ connections.</description>
  </resourceGroupInformation>

  <privacySettings>
    <privacySetting identifier="sendData"
     ↪ validValueDescription="'true',
     ↪ 'false'">
      <name>Send Data</name>
      <description>Allows apps to send data
       ↪ to the Internet.</description>
    </privacySetting>
    <privacySetting
     ↪ identifier="destinationAddress"
     ↪ validValueDescription="'PRIVATE',
     ↪ 'HEALTH', 'LOCATION', 'PUBLIC'">
      <name>Destination Address</name>
      <description>Restricts the admissible
       ↪ destination
       ↪ addresses.</description>
    </privacySetting>
    ...
  </privacySettings>
</resourceGroupInformationSet>
```

Listing 2: *Resource Group Information Set* for the *Internet Resource Group* (excerpt).

While the *Feature* selection is more adequate for normal users, the direct configuration of the *Privacy Settings* is meant for fine tuning by expert users. According to the activated *Features* and the configuration of the *Privacy Settings*, the *PMP* adapts the application's program flow, binds the required *Resources*, and performs the requested anonymization operations. The user can adjusted all settings at runtime, e. g., to activate additional *Features*. Neither applications nor *Resource Groups* have to deal with any of these data or program flow changes.

6 ASSESSMENT

As shown by prevailing studies, mobile platforms have to face novel challenges concerning the privacy-aware processing of data from *Smartbands* (Funk, 2015, Patel, 2015). Since Android permissions are based on technical functions of a Smartphone, there is only a single generic BLUETOOTH permission restricting access to any kind



(a) *Feature* Selection (b) *Privacy Settings* (c) Internet Access Restriction

Figure 5: *PMP*-based Permission Configuration.

of Bluetooth devices including headphones, *Smartbands*, and even medical devices.

On the contrary, our approach introduces a more **data-oriented permission model**. In this way the user is able to select specifically which data or function of a *Smartband* an application should have access to. Moreover, the *PPM*, which is the basis of our model, supports not only two-valued permission settings (grant and deny) but also numerical or textual constraints. Thereby, it enables a **fine-grained access control**, which is essentially for devices such as *Smartbands* dealing with a lot of different sensitive data. Lastly, our model is **extensible**. That is, new devices can be added at runtime as *Resources* and are immediately available for any application. In conclusion, due to these three key features our approach solves the privacy challenges of *Smartband* applications.

In addition, our approach also provides a solution for another big challenge in the context of *Smartband* applications: The **interoperability of devices** is low. This means in effect, that each device uses its proprietary data format for the data interchange with an application (Chan et al., 2012). So, each application supports a limited number of *Smartbands*, only. With our *Smarband Resource Group*, an application developer has to program against its given unified interface and the *PMP* selects the appropriate *Resource* which handles the data interchange.

Evaluation results of other *Resource Groups* show that the *PMP* produces an acceptable overhead concerning the overall runtime of an application, the average CPU usage, the total battery drain, and the memory usage (Stach and Mitschang, 2016). Likewise, the usability of the *PMP* satisfies the users' expectations (Stach and Mitschang, 2014). Both properties apply also to our approach since it is based on the *PMP*. Therefore, the usage of the *PMP* is particularly useful in a health context (Stach et al., 2018), as early prototypes of health applications have shown (Stach, 2016).

However, our approach is only able to protect the user's privacy as long as his or her data is processed on the Smartphone. Once the data is sent out, the user is no longer in control. Since many applications fall back on online services for data processing (such as (Wieland et al., 2016) or (Steimle et al., 2017)), it is part of future work to deal with this problem. In the following, we give an outlook on a solution to this problem.

7 CONCLUSION AND OUTLOOK

Since the *Internet of Things* gains in importance, new devices with various sensors come into the market. That way, diverse measuring instruments for home use are available for the end-user. Especially *Smartbands* and *Smart Watches*, i. e., hardware devices equipped with GPS and a heartbeat sensor among others which are carried on the wrist, are becoming more and more popular. Due to their capability to interconnect with other devices, Smartphone applications are able to make use of the captured data (e. g., innovative fitness applications are able to create a *quantified self* by analyzing this data). However, without an appropriate protection mechanism, such applications constitute a threat towards the user's privacy, as *Smartbands* have access to a lot of sensitive data.

Since the privacy mechanisms in current mobile platforms—namely Android and iOS—constitute no protection at all as they are not tailored for the usage of *Smartbands*, we introduce a novel privacy mechanism specially designed for this use case. Our approach is based on the **P**rivacy **P**olicy **M**odel (*PPM*) and implemented for the **P**rivacy **M**anagement **P**latform (*PMP*) (Stach and Mitschang, 2013, Stach and Mitschang, 2014). In this way we are able to provide a fine-grained access control to each of a *Smartband's* data type. Moreover, the user is able to restrict the network access in terms of selecting valid addresses with which an application is allowed to establish a connection.

Evaluation results show, that our approach meets the requirements of such a privacy mechanism. However, this is just a first protective measure. As modern Smartphone applications commonly serve as data sources for comprehensive stream processing systems realizing the actual computation. These systems have access to a wide range of sources and therefore they are able to derive a lot of knowledge. Even if a user restricts access to a certain type of data on his or her device, a stream processing system could be able to retrieve this data from another source. Therefore the privacy rules of each application also have to be applied to affiliated services which process the application's data.

Future work has to investigate, how the *PPM*-based rules can be applied to a privacy mechanism for stream processing systems such as the *PATRON* research project[3] (Stach et al., 2017a).

[3]see http://patronresearch.de/

ACKNOWLEDGEMENTS

This paper is part of the PATRON research project which is commissioned by the Baden-Württemberg Stiftung gGmbH. The authors would like to thank the BW-Stiftung for the funding of this research.

REFERENCES

Barrera, D., Kayacik, H. G., van Oorschot, P. C., and Somayaji, A. (2010). A Methodology for Empirical Analysis of Permission-based Security Models and Its Application to Android. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 73–84.

Barrera, D. and Van Oorschot, P. (2011). Secure Software Installation on Smartphones. *IEEE Security and Privacy*, 9(3):42–48.

Böhme, R. and Grossklags, J. (2011). The Security Cost of Cheap User Interaction. In *Proceedings of the 2011 New Security Paradigms Workshop*, NSPW '11, pages 67–82.

Chan, M., Estève, D., Fourniols, J.-Y., Escriba, C., and Campo, E. (2012). Smart Wearable Systems: Current Status and Future Challenges. *Artificial Intelligence in Medicine*, 56(3):137–156.

Davis, B. and Chen, H. (2013). RetroSkeleton: Retrofitting Android Apps. In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '13, pages 181–192.

Davis, B., Sanders, B., Khodaverdian, A., and Chen, H. (2012). I-ARM-Droid: A Rewriting Framework for In-App Reference Monitors for Android Applications. In *Proceedings of the 2012 IEEE Conference on Mobile Security Technologies*, MoST '12, pages 28:1–28:9.

Enck, W., Ongtang, M., and McDaniel, P. (2009). Understanding Android Security. *IEEE Security and Privacy*, 7(1):50–57.

Felt, A. P., Egelman, S., Finifter, M., Akhawe, D., and Wagner, D. (2012a). How to Ask for Permission. In *Proceedings of the 7th USENIX Conference on Hot Topics in Security*, HotSec '12, pages 1–6.

Felt, A. P., Ha, E., Egelman, S., Haney, A., Chin, E., and Wagner, D. (2012b). Android Permissions: User Attention, Comprehension, and Behavior. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, SOUPS '12, pages 3:1–3:14.

Funk, C. (2015). IoT Research - Smartbands. Technical report, Kaspersky Lab.

Hornyack, P., Han, S., Jung, J., Schechter, S., and Wetherall, D. (2011). 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*, CCS '11, pages 639–652.

Khorakhun, C. and Bhatti, S. N. (2015). mHealth through quantified-self: A user study. In *Proceedings of the 2015 17th International Conference on E-health Networking, Application & Services*, HealthCom '15, pages 329–335.

Knighten, J., McMillan, S., Chambers, T., and Payton, J. (2015). Recognizing Social Gestures with a Wrist-Worn SmartBand. In *Proceedings of the 2015 IEEE International Conference on Pervasive Computing and Communication Workshops*, PerCom Workshops '15, pages 544–549.

Lee, M., Lee, K., Shim, J., Cho, S.-j., and Choi, J. (2016). Security Threat on Wearable Services: Empirical Study using a Commercial Smartband. In *Proceedings of the IEEE International Conference on Consumer Electronics-Asia*, ICCE-Asia '16, pages 1–5.

Leontiadis, I., Efstratiou, C., Picone, M., and Mascolo, C. (2012). Don't kill my ads!: Balancing Privacy in an Ad-Supported Mobile Application Market. In *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*, HotMobile '12, pages 2:1–2:6.

Mayfield, J. and Jagielski, K. (2015). FTC Report on Internet of Things Urges Companies to Adopt Best Practices to Address Consumer Privacy and Security Risks. Technical report, Federal Trade Commission.

Nauman, M., Khan, S., and Zhang, X. (2010). Apex: Extending Android Permission Model and Enforcement with User-defined Runtime Constraints. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '10, pages 328–332.

Patel, M. (2015). The Security and Privacy of Wearable Health and Fitness Devices. Technical report, IBM SecurityIntelligence.

Pombo, N. and Garcia, N. M. (2016). ubiSleep: An Ubiquitous Sensor System for Sleep Monitoring. In *Proceedings of the 2016 IEEE 12th International Conference on Wireless and Mobile Computing, Networking and Communications*, WiMob '16, pages 1–4.

Saracino, A., Martinelli, F., Alboreto, G., and Dini, G. (2016). Data-Sluice: Fine-grained traffic control for Android application. In *Proceedings of the 2016 IEEE Symposium on Computers and Communication*, ISCC '16, pages 702–709.

Stach, C. (2013). How to Assure Privacy on Android Phones and Devices? In *Proceedings of the 2013 IEEE 14th International Conference on Mobile Data Management*, MDM '13, pages 350–352.

Stach, C. (2015). How to Deal with Third Party Apps in a Privacy System — The PMP Gatekeeper. In *Proceedings of the 2015 IEEE 16th International Conference on Mobile Data Management*, MDM '15, pages 167–172.

Stach, C. (2016). Secure Candy Castle — A Prototype for Privacy-Aware mHealth Apps. In *Proceedings of the 2016 IEEE 17th International Conference on Mobile Data Management*, MDM '16, pages 361–364.

Stach, C., Dürr, F., Mindermann, K., Palanisamy, S. M., Tariq, M. A., Mitschang, B., and Wagner, S. (2017a). PATRON — Datenschutz in Datenstromverarbeitungssystemen. In *Informatik 2017: Digitale Kulturen, Tagungsband der 47. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 25.9-29.9.2017, Chemnitz*, volume 275 of *LNI*, pages 1085–1096. *(in German).*

Stach, C. and Mitschang, B. (2013). Privacy Management for Mobile Platforms – A Review of Concepts and Approaches. In *Proceedings of the 2013 IEEE 14th International Conference on Mobile Data Management*, MDM '13, pages 305–313.

Stach, C. and Mitschang, B. (2014). Design and Implementation of the Privacy Management Platform. In *Proceedings of the 2014 IEEE 15th International Conference on Mobile Data Management*, MDM '14, pages 69–72.

Stach, C. and Mitschang, B. (2016). The Secure Data Container: An Approach to Harmonize Data Sharing with Information Security. In *Proceedings of the 2016 IEEE 17th International Conference on Mobile Data Management*, MDM '16, pages 292–297.

Stach, C., Steimle, F., and Franco da Silva, A. C. (2017b). TIROL: The Extensible Interconnectivity Layer for mHealth Applications. In *Proceedings of the 23nd International Conference on Information and Software Technologies*, ICIST '17, pages 190–202.

Stach, C., Steimle, F., and Mitschang, B. (2018). The Privacy Management Platform: An Enabler for Device Interoperability and Information Security in mHealth Applications. In *Proceedings of the 11th International Conference on Health Informatics*, HEALTHINF '18. *(to appear).*

Steimle, F., Wieland, M., Mitschang, B., Wagner, S., and Leymann, F. (2017). Extended Provisioning, Security and Analysis Techniques for the ECHO Health Data Management System. *Computing*, 99(2):183–201.

Wieland, M., Hirmer, P., Steimle, F., Gröger, C., Mitschang, B., Rehder, E., Lucke, D., Abdul Rahman, O., and Bauernhansl, T. (2016). Towards a Rule-based Manufacturing Integration Assistant. *Procedia CIRP*, 57(Supplement C):213–218.

Wijaya, R., Setijadi, A., Mengko, T. L., and Mengko, R. K. L. (2014). Heart Rate Data Collecting Using Smart Watch. In *Proceedings of the 2014 IEEE 4th International Conference on System Engineering and Technology*, ICSET '14, pages 1–3.

Xu, R., Saïdi, H., and Anderson, R. (2012). Aurasium: Practical Policy Enforcement for Android Applications. In *Proceedings of the 21st USENIX Security Symposium*, Security '12, pages 539–552.