Fine-Grained Privacy Control for Fitness and Health Applications Using the Privacy Management Platform

Christoph Stach

Institute for Parallel and Distributed Systems, University of Stuttgart,
Universitätsstraße 38, D-70569 Stuttgart, Germany
stachch@ipvs.uni-stuttgart.de

Abstract. Due to the Internet of Things, novel types of sensors are integrated into everyday objects. A domain that benefits most is the fitness and health domain. With the advent of the so-called *Smartbands*—i. e., bracelets or watches with built-in sensors such as heart rate sensors, location sensors, or even glucose meters—novel fitness and health application are made possible. That way a *quantified self* can be created. Despite all the advantages that such applications entail, new privacy concerns arise. These applications collect and process sensitive health data. Users are concerned by reports about privacy violations. These violations are enabled by inherent security vulnerabilities and deficiencies in the privacy systems of mobile platforms. As none of the existing privacy approaches is designed for the novel challenges arising from Smartband applications, we discuss, how the **P**rivacy **P**olicy **M**odel (*PPM*), a fine-grained and modular expandable permission model, can be applied to this application area. This model is implemented in the **P**rivacy **M**anagement **P**latform (*PMP*). Thus the outcomes of this work can be leveraged directly. Evaluation results underline the benefits of our work for Smartband applications.

Keywords: Smartbands · Health and Fitness Applications · Privacy Concerns · Bluetooth · Internet · Privacy Policy Model · Privacy Management Platform.

1 Introduction

The *Internet of Things* has significantly revolutionized our daily lives. As sensors, microprocessors, and memory became smaller, more powerful, and, above all, cheaper, this technology is increasingly integrated into everyday objects which we carry with us permanently. Examples for such *Smart Devices* are *Smartphones*, *Smart Watches*, or *Smart Bracelets*. These devices can run small third-party applications called *apps*. Perhaps the most important feature, however, is that these devices can be connected with each other. This way, the Smart Devices can provide their gathered sensor data to other devices and applications. Due to energy-efficient connection and transmission technologies such as Bluetooth

In: Mori P. et al. (Eds.) Information Systems Security and Privacy. ICISSP 2018. Communications in Computer and Information Science, vol 977. Springer, Cham, pp. 1–25, 2019.

LE, this interconnection has little to no impact on their battery life. As a result, novel application cases are constantly arising from different domains, which make use of these accumulated data stocks.

The consumer market is currently dominated by *Smartbands*. These hardware devices are equipped with GPS and a heartbeat sensor. Therefore, they are ideally suited for fitness apps. The Smartband is only used for data collection, while the actual fitness app is run on a connected Smartphone. This means that data processing (including data preparation, data analysis, and data presentation) is completely handled by the Smartphone. Since the smartphone carries many additional information about its user, the fitness data can also be linked and augmented with this data. As a result, a lot of further insights can be gained. As *Wearables* such Smartbands are small and comfortable to wear, they virtually disappear from our awareness [66]. This means that they can also be kept on while doing sports or even while sleeping.

Innovative apps take advantage of this persistent data capturing. For instance, a fitness app can analyze data from acceleration sensors and orientation sensors to identify movement patterns and determine the current activities of a user [32]. Location data can be used to determine the distance traveled by a jogger as well as his or her running speed and thus calculate his or her calorie consumption [67]. Heartbeat data can even be used to analyze the sleeping behavior of a user [43]. Such a comprehensive health profile is not only beneficial for the user, but also for his or her physician and many other stakeholders [30].

Yet, a *quantified self*, i. e., a comprehensive mapping of our lifestyle to quantifiable values to assess our daily routines, does not come without a price. This permanent, self-imposed monitoring poses a threat to our privacy. Smartbands and other Smart Devices collect so many personal data that a great deal of knowledge about the user can be derived from it. Many research activities are therefore concerned with the concrete threats posed by such innovative apps as well as their vulnerabilities [35] and what measures can be taken to provide security for these apps [38]. Particularly as the economic value of personal data increases [19], a completely new app business model has emerged. Users pay the usage of an app with their data, which is then sold to third parties, such as advertising clients [36]. Therefore, new control measures are needed to enable users to decide which personal information they are willing to disclose in return for what service [42].

To that end, we address the following five issues in our work:

(1) We introduce a real world mHealth use case for Smartband apps.
(2) We provide a comprehensive overview of state of the art concerning the protection of private data in the context of Smartband apps.
(3) We adapt a privacy policy model which enables users to control the data usage of Smartband apps in a fine-grained manner. Our approach is based on the **P**rivacy **M**anagement **P**latform (*PMP*) [59] and its **P**rivacy **P**olicy **M**odel (*PPM*) [58].
(4) We introduce a prototypical implementation of a privacy mechanism for Smartband apps using our privacy policy model.

(5) We evaluate our approach and demonstrate its applicability.

This paper is the extended and revised version of the paper entitled "Big Brother is Smart Watching You: Privacy Concerns about Health and Fitness Applications" [55] presented at the 4th International Conference on Information Systems Security and Privacy (ICISSP) 2018. This extended paper is more detailed on a technical level. The structure of a Smartband app is outlined, requirements towards a privacy system are derived from that app, and considerably more related approaches are discussed.

The remainder of the paper is structured as follows: Initially, a use case for Smartband apps is outlined in Sect. 2. In Sect. 3, current privacy control mechanisms for mobile platforms are discussed and the prevailing connection standard Bluetooth LE is characterized. Requirements towards a privacy system for Smartband apps is derived from this analysis in Sect. 4. Section 5 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 Sect. 6. Following this, our generic concept is implemented using the PMP in Sect. 7. Section 8 evaluates our approach and reveals whether it fulfills the requirements towards such a privacy control mechanism. Finally, Sect. 9 concludes this paper and glances at future work.

2 mHealth Use Case

Modern wearable Smart Devices such as Smartbands are equipped with multiple sensors and accordingly more and more data about their users can be acquired by them. While built-in GPS receivers are great to archive outdoor positioning, these sensors are virtually useless for indoor positioning. Therefore, there is a lot of research going on to archive indoor positioning with other standard sensors available in almost any Smart Device. Hsu et al. [25] introduce an approach using accelerometers and gyroscopes for that purpose. Another approach for indoor positioning is the use of the barometric pressure sensor [70]. Finally, the position can also be determined based on earth's magnetic field—i. e., via the compass sensor within the Smart Device [68]. However, these simple sensors can be used for more than just (indoor) positioning. In particular, wrist-based Smart Devices such as Smartbands enable to recognize activities of a user with high accuracy [27, 49]. Medical sensors are also increasingly being integrated in Smartbands. Whereas methods for monitoring the heart rate are already widely used [1], there are also attempts to collect medical data, such as blood sugar levels, via such devices [65].

Due to these features, it is little surprising that these devices are increasingly being used for medical apps [44]. So-called *mHealth* apps can facilitate patients' lives, relieve physicians, and reduce treatment costs [37]. Due to the versatility of Smartbands, there is an app for almost every health-related issue [50]. In the following, an app for children suffering from diabetes is outlined which is based on *Candy Castle* [20, 33, 54, 61].

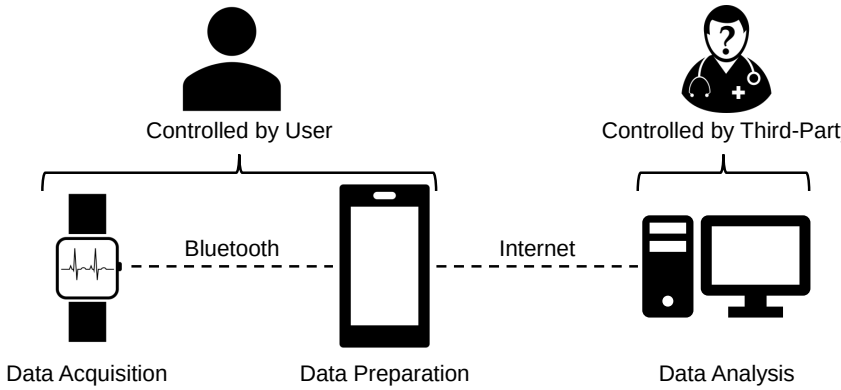


Fig. 1: Set-Up of a Smartband App.

The aim of Candy Castle is to motivate children suffering from diabetes to check their blood sugar level regularly and keep their diabetes diary. To that end, the Smartphone app turns the children into a virtual owners of a castle. This castle represents their health condition. For this reason, it is regularly attacked by *dark forces*—i. e., the diabetes disease—and the children have to defend it with their *magic device*—i. e., their Smartband. This act of defense means that they carry out a blood sugar measurement. Apart from the actual blood sugar level, the Smartband also captures the child's most recent activities (e. g., to determine whether s/he did sports or took insulin) and his or her current location—it is assumed that the location has great influence on the health condition [34]. All this data is sent to the Smartphone and processed there: The player gets a reward (the castle gets repaired and upgraded) and a new entry is automatically added to an electronic diabetes diary. At some point in time this diary is sent to the physician, e. g., by uploading it to a *Hospital Cloud*. This approach enables to carry out comprehensive analyses on the health data and provide physicians with all required information [8, 64].

Figure 1 shows the set-up of such an app. Most of the data acquisition is done on the Smartband. However, these devices are not powerful enough to process and combine all of this data. For this reason, they have to be sent to a Smartphone. This connection and data transfer is realized via Bluetooth. The data is then prepared on the Smartphone and provided to the Candy Castle. Even though Smartphone are becoming more and more powerful, they are not designed for comprehensive analyses. Therefore, data has to be sent to an external data processing unit via the Internet connection of the Smartphone. However, this implies that the user looses all control over his or her data. Especially, s/he cannot be sure about the identity of the recipient. Therefore, privacy actions have to be taken at the Smartphone.

3 State of the Art

Based on the aforementioned set-up, we explain, why especially the usage of apps for Smartbands and similar Smart Devices such as health or fitness apps 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. The Smart Device market is currently dominated by two operating system, namely Apple's iOS and Google's Android [31]. Both of those mobile platforms apply a permission-based system to protect sensitive data [16]. This means that each app must specify which data it will process. Each time data is accessed, the system checks whether the respective permission can be granted. A permission does not refer to a specific type of data, but to a sensor or a potentially dangerous system functionality [5]. However, both mobile platforms implement this concept differently.

An iOS app requires Apple's approval before it is released. Automated and manual verification methods check whether the permission requests are justified. When Apple grants the permissions, the app is signed and released. The user is only informed about permissions concerning his or her personal data (e. g., the contacts) [40].

In contrast, Google does not intervene at all in the permission process. If an Android app is installed, the user is notified of any requested permissions and must grant them all to proceed with the installation process [6]. *Runtime Permissions* are therefore introduced in Android 6.0. A Runtime Permission is not assigned at installation time, but it has to be granted for each access to data that is protected by the corresponding permission [29].

However, studies show that users cannot cope with the multitude of different permissions—especially since they cannot understand the consequences of granting a certain permission [17]. Therefore Google divides the permissions in Android since version 6.0 into two classes: *Normal Permissions* no longer require the user's consent. Apps only have to indicate the usage of a Normal Permission in their Manifest. Only *Dangerous Permissions* (which are a superset of the Runtime Permissions) have to be granted by the user explicitly. For instance, the ACCESS_FINE_LOCATION (access to the GPS) or BODY_SENSORS (access to heart rate data) permission belong to this category. However, the BLUETOOTH and INTERNET permission are classified as Normal Permissions. Table 1 gives a comprehensive overview of Normal and Dangerous Permissions.

Figure 2 shows the effects of this decision. An app that needs to access GPS data, discover, pair with, and connect to Bluetooth devices, and open network sockets must declare the following four permissions: ACCESS_FINE_LOCATION, BLUETOOTH, BLUETOOTH_ADMIN, and INTERNET. In pre-Marshmallow Android versions (< 6.0), users must grant all permissions during installation. However, the installation dialog only informs the user about the Dangerous Permissions (see Fig. 2a). On devices with a higher Android version, Runtime Permissions are no longer shown as they have to be granted need-based (see

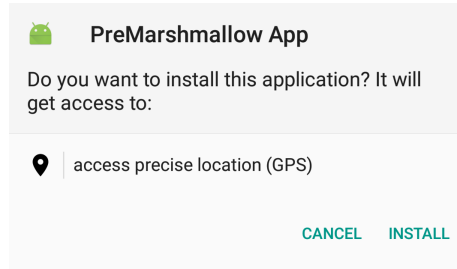
Table 1: Normal and Dangerous Android Permissions (excerpt) [based on 21].

Normal Android Permissions	*Dangerous Android Permissions*
ACCESS_NETWORK_STATE	READ_CALENDAR
ACCESS_WIFI_STATE	CAMERA
BLUETOOTH	READ_CONTACTS
BLUETOOTH_ADMIN	ACCESS_FINE_LOCATION
INSTALL_SHORTCUT	ACCESS_COARSE_LOCATION
INTERNET	RECORD_AUDIO
NFC	READ_PHONE_NUMBERS
REQUEST_INSTALL_PACKAGES	CALL_PHONE
SET_ALARM	ANSWER_PHONE_CALLS
TRANSMIT_IR	READ_CALL_LOG
UNINSTALL_SHORTCUT	BODY_SENSORS
USE_FINGERPRINT	SEND_SMS
VIBRATE	READ_SMS
WAKE_LOCK	READ_EXTERNAL_STORAGE

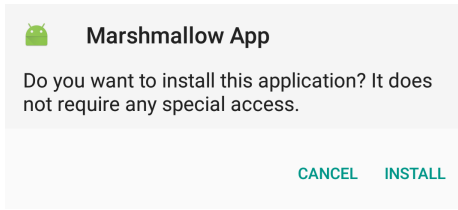
Fig. 2b). Thus, each time the app attempts to access GPS data, a permission request pops up (see Fig. 2c). In any case, the user is not aware that the app is also granted to transfer this data to any Bluetooth device or the Internet.

Transmission Standard of Smart Devices. Bluetooth LE has become today's connection standard for Smart Devices. It uses consumes less power than Classic Bluetooth and has a longer operating range than NFC. The device manufacturer defines UUIDs that other devices can use to request services provided by the device. For example, a service of a Smartband could allow access to a built-in heart rate sensor. The manufacturer also specifies how the data is encoded by the device. A mobile platform therefore cannot determine which type of data is transferred between two Smart Devices, since it cannot know which services are addressed by which UUID. Moreover, without knowledge about the applied encoding, the platform cannot look into the transferred data. Therefore, the permissions only control the Bluetooth connection itself and not which data is transferred via this connection. The same applies to the forwarding of data to a server. Again, an app only needs to indicate that it needs access to the Internet, but the user does not know what data the app is sending or where the data is being sent to.

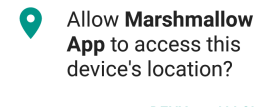
For instance, if a Smartband has a built-in GPS and heart rate sensor, then it can provide access to both, location and health data. An app only requires permission to discover, pair with, and connect to Bluetooth devices (BLUETOOTH and BLUETOOTH_ADMIN permission). However, both permissions belong to



(a) Android 5.1 Installation Dialog



(b) Android 6.0 Installation Dialog



(c) Request at Runtime

Fig. 2: Permission Requests in Different Android Versions [55].

the Normal Permissions category. That is, the system grants these permissions automatically and the user is not informed about it. If the same app would request the very same data directly from sensors which are built into a Smartphone, the ACCESS_FINE_LOCATION and BODY_SENSORS permissions are required. Both are Dangerous Permissions, i. e., the user must grant every access at runtime. As this kind of data is highly sensitive, that classification is reasonable. The use of a Smartband however completely override this protective measure. In addition, the app is able to share this information with any external sink without the user's knowledge. It only has to declare the INTERNET authorization in its Manifest—the INTERNET permission is also a Normal Permission. Therefore, a static, permission-based data privacy mechanism, as implemented in current mobile platforms, is not applicable to apps which access their data from Smart Devices such as Smartbands.

Since Android puts the user in charge of protecting his or her sensitive data, such a security vulnerability when dealing with data from Smartbands might have serious consequences. Therefore, this paper focuses on Android. However, the findings and concepts can be transferred to any other mobile platform.

4 Requirements Specification

Based on the identified deficiencies in current mobile platforms concerning privacy in Smartband apps, requirements for a privacy system can be derived. These requirements primarily focus on securing the two resources Bluetooth and Internet.

[R_1] Fine-Grained Privacy Rules. Although, Android provides a wide range of permissions, some of them are unnecessary (from a privacy point of

view) such as the VIBRATE permission and others are far too coarse-grained such as the BLUETOOTH permission. Therefore, a privacy system has to split these permissions or introduce new fine-grained permissions. Only by introducing fine-grained privacy rules, users are able to understand the meaning of permissions and express their privacy requirements.

[R_2] Extendable Permission Set. New and innovative developments are constantly emerging, especially in the area of Smart Devices. A privacy system must therefore be able to adapt to these technical innovations in future generations of Smart Devices. That is, a privacy system has to be extendable in order to support—i. e., provide data security for—new sensors and data processing techniques.

[R_3] Policy Changes at Runtime. The requirements of a user can vary at any time. For instance, s/he might want to execute a certain function which require a lot of permissions on rare occasions only. In such a case, it has to be possible that an app which provides this function only receives the corresponding permissions for a short amount of time. Similar to Android's Runtime Permissions the user therefore has to be able to change the privacy policy at runtime—yet, these runtime changes have to be available for any kind of permission.

[R_4] Context-Based Privacy Rules. External factors can also influence a user's privacy requirements. For instance, an app may be granted more permissions in case of an emergency. Therefore, a privacy system must decide based on context data which privacy rules apply in the current situation.

[R_5] Feedback. In order for a privacy system to be effective, i. e., to enable the user to protect his or her interests with regard to data security, it is crucial that s/he is fully involved in the permission process. For this reason, a privacy system has to be designed to provide the user with comprehensive information about his or her options for granting permissions and also to make him or her aware of the possible consequences of his or her settings.

5 Related Work

As the prevailing privacy mechanisms applied in the current mobile platforms do not comply these requirements, 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 apps.

Apex [41] enables the user to add contextual conditions to each Android permission. These conditions specify situations in which a permission is granted. E. g., the user can set a timeframe in which an app 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 app crashes. Furthermore, as Apex is based on the existing Android permissions, it is too coarse-grained for the Smartband use case.

AppFence [24] analyzes the internal dataflow of apps. 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 app is started. However, AppFence does not knows which data an apps reads from a Bluetooth source. Thereby, it cannot differentiate whether an apps 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.

AppGuard [3] introduces a data protection system that integrates a monitoring component into apps which supervises apps from within. It consists of three components: (1) a pre-configured set of rules which are directly mapped to Android permissions, (2) an app converter that injects the monitoring component and the rule set into existing apps, and (3) a GUI, via which further, user-defined rules can be added. AppGuard also enables to describe how the control flow of an app should be modified if it violates any of the rules. However, as the rules are mapped to existing Android permissions, AppGuard has the same shortcomings regarding Smartbands. Moreover, the usage of the app converter violates copyright law [2].

AUDACIOUS [45] addresses this issue by introducing a program library via which experts are able to perform static and dynamic analyses on apps. As this library has to be integrated by the app developers themselves, this approach does not violate copyright law. The analyses reveal which data is used by an app and how it is processes. If any conspicuous data usage is detected, the app is stopped by AUDACIOUS. However, the rules are not defined by the user, but by experts who determine on their own which data usage is permissible.

Aurasium [69] introduces an additional sandbox which is injected into every app. This has to be done before the app is installed. The sandbox monitors its embedded app 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 app 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 app is costly and violates copyright and related rights.

CRêPE [11] is a context-based privacy system for Android. CRêPE uses a powerful situation recognition system to draw conclusions about the current activities of a user [12]. Via this technique, higher-level contexts can be described instead of simply linking single sensor values. Each privacy rule consists of a subject-object-permission triple. The subject is either a user or another app, the object represents any kind of data source, and the permission defines whether the given subject may access the object. A context is added to each triple and the rule is only active under that specific context. The access control is ensured via *XACML* [39]. However, CRêPE is not designed for end-users and the privacy rule creation is far too complex for common users.

Data-Sluice [47] 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 apps 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 app from any other source, except for the Internet.

Dr. Android & Mr. Hide [26] addresses the problem that many developers are unable to handle the Android permissions and therefore unintentionally give too many permissions to their apps [28]. To enable developers to assign permissions in a better way, Jeon et al. split the existing permissions into fine-grained permissions. The novel permissions are based on the most common activities of apps which require private data. The permissions are divide into four classes. For each class, the user can apply different anonymization techniques. To enforce the permission settings, the program library *hidelib* is provided which reimplements the APIs of the Android app framework based on the new permissions. However, Dr. Android & Mr. Hide manipulates the bytecode of each controlled app, whereby it also violates copyright law.

IacDroid [71] does not directly address the issue of granting permissions, but a related topic. As Android provides a wide range of possibilities for interprocess communication (IPC), apps often exchange data and even permissions in an uncontrolled manner [22, 48]. Therefore, Zhang et al. introduce two components that monitor and regulate data and permission exchange at runtime. All IPCs are monitored for this purpose. This enables IacDroid to infer a sequence of process calls. The user can then assign special permissions for each call. However, this does not tackle the underlying issue of data collection in Smartband apps.

MockDroid [7] provides additional privacy settings for specific data sources (including location data or the contacts) and system functions (including access to the Internet or writing SMS messages). Via these settings, apps can reduce the required permissions, e. g., by requesting Internet access to a specific address, only. In addition, users can decide, whether apps have access to actual data or whether MockDroid should provide them with fake data instead. However, all of these settings are available for supported data sources and system functions, only. While Internet access is protected by MockDroid, access to Bluetooth devices is still unsecured.

Privacy Protector (No root) [23] is an Android app which promises a simple privacy protection. However, the Privacy Protector only considers location data and Internet access as safety hazards. Therefore, the user can specify which apps should have access to it. Privacy Protector permanently monitors which apps are currently running and if any of the regulated apps are among them, the Internet or respectively the GPS tracking functions are deactivated system-wide. This has an effect on all running apps. Moreover, since Android 5.0 the `getRunningTasks` method has been severely restricted to prevent apps from spying on user behavior. This also reduces the functionality of Privacy Protector sustainably.

I-ARM Droid [14] 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* [13] assigns this task to a security expert who creates a configuration according to the user's demands. Thereby frequent changes of the privacy rules are not possible—not to mention 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.

SEAF [4] considers that the order in which permissions are requested might affect the risk potential of an app. For instance, an app that first gets access to the Internet and then accesses confidential data can do less damage than an app that performs these operations in reverse order. Banuri et al. therefore identify several operation sequences that indicate a potential data misuse and define the order in which permissions have to be requested to execute each of these operation sequences. SEAF monitors apps for such sequences of permission requests. If such a sequence is detected, SEAF informs the user and s/he decides whether the operations should be executed or whether the required permissions should be denied. However, SEAF is based on the coarse-grained Android permissions, and therefore it is not suitable for Smartband apps.

Sorbet [18] enables app developers to use IPC in a secure way. Sometimes it is required that apps exchange permissions via IPC. However, as the validity of Android permission is neither restricted in time nor in functionality, this leads to an almost unlimited and uncontrollable situation. Sorbet therefore enables controlled delegation of permissions and data. For this purpose, Sorbet records where the data or permissions originated from and to whom they were passed on. Each of these records is tagged with an expiration date which is specified by the permission originator, i. e., the app that received the permission in the first place. However, this does not solve the privacy problems of Smartband apps.

YAASE [46] introduces a new fine-grained permission model in order to reduce uncontrolled information passing between apps. In this model, the user has the option of tagging his or her data and thereby defining at a data level to which destination it may be sent to. Other apps as well as external recipients (e.g. Internet servers) can be used as targets. To monitor the inter-application information flow, *TaintDroid* [15] is used. In order to be able to monitor communication with Internet servers as well, YAASE also modifies the methods to establish Internet connections at a kernel level. This way, YAASE is always informed about the destination of any connection. As soon as a data transfer to an app via IPC or to an Internet server is detected that violates the privacy requirements of a user, any transferred data is concealed. However, YAASE provides no protection especially focused on Bluetooth devices.

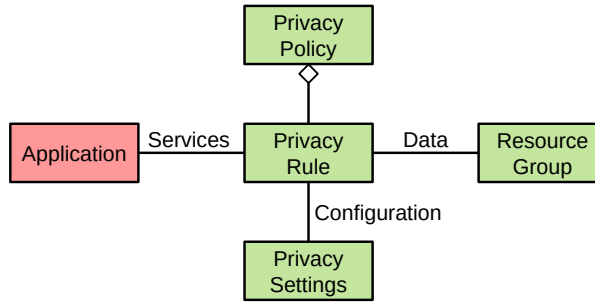


Fig. 3: Simplified Representation of the Privacy Policy Model (untrusted components are shaded red and trusted components are shaded green) [based on 55, 58].

6 A Permission Model for Smartbands

None of the analyzed related work is applicable to restrict access to data from Smartbands as their permissions are too coarse. Moreover, they lack modular expandability. As a result, they quickly become obsolete as they cannot adapt to the privacy challenges originating from new device or data types. The **P**rivacy **M**anagement **P**latform (*PMP*) [53, 58, 59] provides these features. In addition, the PMP provides support for the connection of Smart Devices to Smartphones [63].

To this end, we add two components to the PMP: the *Smartband Resource Group* and the *Internet Resource Group*. These components enable the PMP to provide data gathered by Smartbands to apps in a privacy-aware manner and also restrict the spreading of sensitive data via the Internet. In the following, we introduce the **P**rivacy **P**olicy **M**odel (*PPM*), which forms the core of the PMP, and describe how we adapt it to the smart band setting (see Sect. 6.1). Then, we outline the mode of operation of the PMP (see Sect. 6.2). Finally, we present the concept of our two extensions (see Sect. 6.3 and Sect. 6.4).

6.1 The Privacy Policy Model

The PPM associates apps with data sources or system functions (labeled as *Resource Groups*). In the PPM, an app is subdivided in its *Features*. For each Feature must be specified, which data or system functions are accessed by it. An interface through which an app can interact with a Resource Group—i. e., access its data or functions—has to be defined for each group. In the *Privacy Rules*, the user specifies which Features of an app should be deactivated to reduce the usage of data or system functions. S/he can also refine any Privacy Rule by adding *Privacy Settings*, e. g., to reduce the accuracy of a Resource Group's data. The set of all Privacy Rules constitutes the *Privacy Policy*. The PPM assumes that apps are untrusted components, while Resource Groups are provided by trusted parties. The simplified model is shown in Fig. 3 as a UML-like class diagram. Further information on PPM can be found in the respective literature [58].

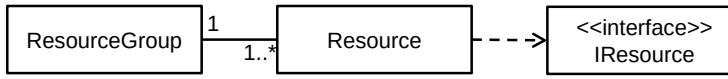


Fig. 4: Architecture of Resource Groups [55].

Only Resource Groups are of interest for the reminder of this work. Figure 4 provides insight into the architecture of a Resource Group. Each Resource Group defines an interface (`IResource`) and descriptors, how the provided data can be protected. The actual implementation of the interface is given in so-called *Resources*. Similar Resources can be bundled in a mutual Resource Group. This way, many alternative implementation variants for the interface can be provided. For instance, a *Location* Resource Group might provide a single method to retrieve the current location of the user. This method is implemented in two different ways, once via the GPS and once via the Cell-ID. Depending on the available hardware, user settings, and so forth, the Resource Group selects the appropriate Resource, when an app requests the data. Moreover, that Resource Group could define an *Accuracy* Privacy Setting that allows the user to define how accurate the location data is, i. e., up to how many meters the actual location should deviate from his or her current location. Of course, s/he can also completely prohibit access to the Resource Group for a certain app.

6.2 The Privacy Management Platform

The PMP is a privacy system that implements the PPM. Due to the structure of the PPM, the PMP has two characteristics that are very advantageous for work:

(*a*) On the one hand, the PMP is **modularly expandable**. This means that additional Resource Groups as well as Resources can be added at runtime. Therefore even the latest device models (by adding Resources) and completely new types of devices or sensors (by adding Resource Groups) are supported automatically by the PMP.

(*b*) On the other hand, the PMP supports **fine-grained access control**. Each Resource Group specifies its own Privacy Settings. These settings correspond to the requirements of the respective device. This allows users not only to turn a device or sensor on and off to protect their private data, but also to add numerical or textual restrictions. For example, a Location Resource Group may have a numeric Privacy Setting that can be used to reduce the accuracy of location data. Another example is an Internet Resource Group which provides a textual Privacy Setting to specify to which addresses an app is allowed to send data to.

To attain these properties, the PMP is an intermediate layer between the application layer and the actual application platform. For the sake of simplicity, the PMP can be seen as an interface to the application platform itself. Figure 5 shows the implementation model of the PMP in a condensed representation. First, an app requests access to data sources or system functions—i. e., to a Resource

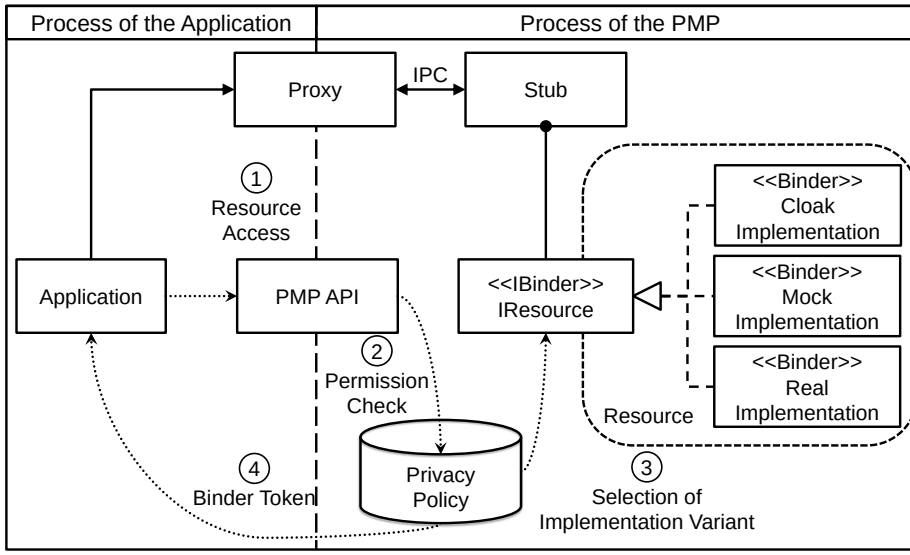


Fig. 5: Simplified Implementation Model of the Privacy Management Platform [55].

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 stipulate the restrictions (Privacy Setting) which apply to the respective app. When access is granted, a suitable implementation (i. e., Resource) is selected within the requested Resource Group (3) . For each Resource, the PMP also has two fake implementations (*Cloak Implementation* and *Mock Implementation*) which provide only anonymized or fully randomized data. The proper implementation of the selected Resource is then linked to the *IBinder* interface as a *Binder*[1]. The PMP forwards a *Binder Token* to the requesting app (4) .

Android's *Binder Framework* manages the actual access to a Resource: The IBinder interface of a Resource is materialized as a so-called *Stub*. *Proxy* components realize the interprocess communication (*IPC*) via which an app can pull data from these Stubs. Without the corresponding Binder Token, an app cannot communicate with a Stub. This ensures that any data request must be made via the PMP. So, the PMP is able to verify that each request complies with the Privacy Policy. Since all Resource Groups are implemented as subpackages of the PMP and run in the same process, they are executed in a mutual sandbox. In this way, the PMP can interact directly with Resource Groups.

These features qualify the PMP for our approach towards a privacy mechanism for Smartband apps. To achieve this goal, two novel Resource Groups are required, (*a*) a Resource Group for Smartbands that restricts access to the various data types of these devices (see Sect. 6.3), and (*b*) a Resource Group that restricts the

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

data transfer of Smartband apps to the Internet (see Sect. 6.4). The specifications for these Resource groups are listed below.

6.3 Smartband Resource Group

The Smartband Resource Group must provide a unified interface to any Smartband model, including Smart Watches and related devices. For this reason, the interface is designed as a superset of data access operations supported by most of these devices. This includes access to personal data (e. g., age or name), health-related data (e. g., heart rate or blood sugar level), activity data (e. g., acceleration or orientation), and location data. Besides these receiving operations, most Smartbands have a small display for displaying short messages as well. The Smartband Resource Group also defines a send operation to display messages on the Smartband. However, not every Smartband model supports each of these operations. This has to be handled by the Resource implementing these functions for the respective Smartband model. The `UnsupportedOperationException` is introduced for this purpose. This exception is automatically caught and handled by the PMP, e. g., by passing mock data to the app.

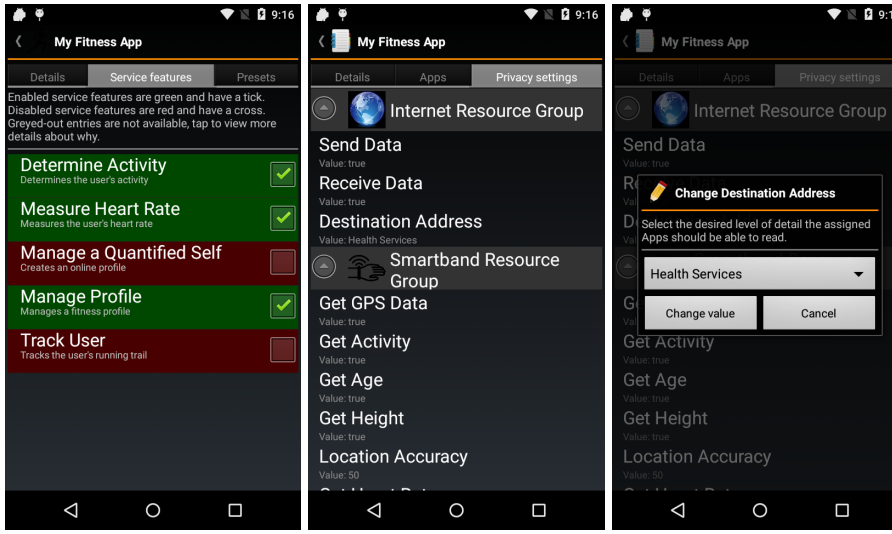
The Smartband Resource Group defines several fine-grained Privacy Settings to restrict access to the data provided by a Smartband. Basically, there is a bivalent Privacy Setting for each type of data, via which the respective data access must be granted or denied. That way, the user can decide which app is allowed to access which data from the Smartband.

As already mentioned, this feature alone is a significant advance over the state of the art because Android supports only one Bluetooth permission for all types of devices and data—not to mention the fact that users cannot see whether an app needs this permission at all. Moreover, the Smartband Resource Group provides additional Privacy Settings for certain types of data. For example, the accuracy of location data can be reduced. In addition, any data source in the Smartband Resource Group can be replaced by a mocked implementation. All mock values are within a realistic range, so apps can't tell the difference.

Furthermore, Smartbands that provide location data can be integrated as additional Resources into the existing *Location Resource Group* as introduced in our previous work [51]. This allows the PMP to switch between the available resources if required (e. g., if the location data of the Smartband is more accurate than the location provided by the Smartphone).

6.4 Internet Resource Group

The Internet Resource Group provides a simplified interface for sending data to and receiving data from a network resource (e. g., a back-end server). Both functions essentially have two parameters, a destination address and the actual payload. The payload parameter is also used to store the response of the network resource. In the context of Smartband apps, such a simplified interface is sufficient. In order to support apps that require a lot of interactions with network resources,



(a) Feature Selection **(b)** Privacy Settings **(c)** Internet Restrictions

Fig. 6: PMP-based Permission Configuration [55].

this interface can be extended by further generic I / O functions (e. g., to support several network protocols).

Similar to the Smartband Resource Group, the Internet Resource Group also defines bivalent Privacy Settings for both I / O functions. The user can decide for each app separately whether s/he wants to allow this app to send data to and / or receive data from the Internet. In addition, the permitted destination addresses can also be restricted. Theoretically, it is possible to do this via a textual Privacy Setting which indicates addresses to trusted network resources. However, the user's attention is limited and such a fine-grained address selection overburdens him or her [9]. For this reason, the Internet Resource Group categorizes addresses into different domains, such as the health domain or a domain for location-based services. There is also a category "public" which does not restrict the permitted destination addresses at all. In this manner, the user can see which domain a certain type of app should have access to. For experts however, the Internet Resource Group can still provide such a textual Privacy Setting described above to fine-adjust the permissible address space.

7 Prototypical Implementation

To verify the applicability of our approach, we have implemented a basic fitness app in addition to the two Resource Groups as described in Sect. 6.3 and Sect. 6.4. The fitness app creates a local user profile, including age, height, and weight. Workout data is collected by the Smartband's motion sensors (e. g., to determine current activities) as well as health data (e. g., the heart rate). These data are

supplemented by location data from the Smartband to detect popular 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.

The fitness app defines five Features that can be individually deactivated by the PMP. Once the app is installed, the PMP displays all these Features and the user can make an initial selection (see Fig. 6a). For example, a user wants to use the fitness app to record his or her workout progress in a local profile, only. However, the app should not track his or her locations in this process and by no means any data should be leaked to the Internet. This selection predefines which service quality the user can expect from the app. To find out what permissions are required for each Feature, the PMP can display additional information.

This interface via which apps are able to interact with the respective Resource Groups is described in the Android Interface Definition Language (*AIDL*). Listing 1 shows such an interface definition for the Smartband Resource Group in excerpts[2].

```
1  interface SmartbandResource {
2
3      // access to personal data
4      int getAge();
5      ...
6      // access to workout data
7      int getHeartRate();
8      ...
9      // access to location data
10     Location getLocation();
11     ...
12
13 }
```

List. 1: Interface Definition for the Smartband Resource Group in AIDL (excerpt) [55].

In addition, the user is able to adapt the Privacy Rules from a Resource Group's point of view as well. To do this, all Resource Groups requested by a respective app are listed together with the Privacy Settings defined by them (see Fig. 6b). Bivalent Privacy Settings such as "Send Data" can be switched on and off directly by simply 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 field. Enumeration Privacy Settings such as "Admissible Destination Address" open an input mask with a selection box (see Fig. 6c). If the selected Privacy Settings are too restrictive for a particular Feature, the PMP informs the user that this Feature had to be deactivated due to conflicting Privacy Settings.

[2] The data type `Location` is not supported by AIDL. Additional type definitions are required to compile this interface definition.

The PMP introduces the so-called *Resource Group Information Set* (*RGIS*) to define Privacy Settings for Resource Groups. Like the Android *App Manifest*, this file contains the metadata required by the PMP about a Resource Group. Listing 2 shows an excerpt of the RGIS Privacy Settings definition for the Internet Resource Group. As can be seen in that listing, each Privacy Setting consists mainly of a unique identifier, a valid range of values, and a human-readable description. The PMP reads these XML files to compile the configuration dialogs for each Resource Group (see Fig. 6b).

```
<?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 out data.</description>
    </privacySetting>
    <privacySetting
          identifier="destinationAddress"
          validValues="'PRIVATE', 'HEALTH', 'LOCATION', 'PUBLIC'">
      <name>Destination Address</name>
      <description>Restricts destination address.</description>
    </privacySetting>
    ...
  </privacySettings>
</resourceGroupInformationSet>
```

List. 2: Resource Group Information Set for the Internet Resource Group (excerpt) [55].

While the Feature selection is more suitable for normal users, the direct configuration of Privacy Settings is intended for fine-tuning by experienced users. According to the selected Features and the configuration of the Privacy Settings, the PMP adapts the program flow of an app, binds the required Resources, and carries out the configured anonymization operations. The user can adjust all settings at runtime, e. g., to activate additional Features. Neither apps nor Resource Groups need to deal with these data or program flow changes.

8 Assessment

As shown by prevailing studies, mobile platforms have to face novel challenges concerning the privacy-aware processing of data from Smartbands [19, 42]. Since

Table 2: Comparison of Privacy Systems for Mobile Platforms [based on 51, 52, 58].

	Feature				
System	**[R_1]**	**[R_2]**	**[R_3]**	**[R_4]**	**[R_5]**
Android	✗	(✓)	✗	✗	✗
Apex	✗	✗	✓	✓	✗
AppFence	(✓)	✗	✓	✗	✗
AppGuard	✗	✗	✓	✓	✗
AUDACIOUS	✓	✓	✗	✓	✗
Aurasium	✓	✓	✓	✗	✓
CRêPE	✓	✓	✓	✓	✗
Data-Sluice	✗	✗	✓	✗	✓
Dr. Android & Mr. Hide	✓	✓	✗	✗	✗
IacDroid	✗	✓	✓	✗	✓
MockDroid	✗	✗	✓	✗	✗
Privacy Protector	✗	✗	✗	✗	✗
RetroSkeleton	(✓)	✓	✗	(✓)	(✓)
SEAF	✗	✗	✓	(✓)	✓
Sorbet	✓	✗	✗	✗	✗
YAASE	✓	✗	✓	✗	✗
PMP	✓	✓	✓	✓	✓

Android permissions are based on technical functions of a Smartphone, there is only a single generic BLUETOOTH permission restricting access to any kind 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 app 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 ([$\mathbf{R_1}$]). In addition, our model is **extendable** ([$\mathbf{R_2}$]). That is, new devices can be added at runtime as Resources and are immediately available for any app. In conclusion, due to these three key features

our approach solves the privacy challenges of Smartband apps. Any privacy rule can be **changed at runtime** ([$\mathbf{R_3}$]). Moreover, due to the PPM, a *context can be added to each rule* in order to define a scope of app ([$\mathbf{R_4}$]). Finally, the user is included in the configuration of the PMP all the time and s/he receives **feedback** so that s/he is able to express his or her privacy requirements in the PPM ([$\mathbf{R_5}$]). A side-by-side comparison of PMP and the related work introduced Sect. 5 is shown in Table 2.

In addition, our approach also provides a solution for another big challenge in the context of Smartband apps: The **interoperability of devices** from different vendors is low. This means in effect, that each device uses its proprietary data format for the data interchange with an app [10]. So, each app supports a limited number of Smartbands, only. With our Smartband Resource Group, an app developer has to program against its given unified interface and the PMP selects the appropriate Resource which handles the data interchange.

Therefore, the usage of the PMP is particularly useful in an health context [60, 62], as early prototypes of health apps have shown [20, 54]. 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 apps fall back on online services for data processing [8, 64], it is part of future work to deal with this problem. In the following section, we give a brief outlook on a possible solution for this problem.

9 Conclusion and Future Work

The improvements in the area of the Internet of Things in recent years have been tremendous. Especially concerning wearable Smart Devices such as Smartbands, there are numerous innovations. An increasing number of sensors are integrated in Smartbands, enabling them to accurately capture the user's context. In addition to capture the user's location, these devices are also able to recognize activities as well as monitor health data. This makes innovative fitness and health apps possible by gather and analyzing all of these data in order to create a quantified self. As the processed data are highly sensitive, these apps require novel privacy mechanisms adapted to the latest innovations in the area of Smartbands.

As neither the prevailing privacy mechanisms applied in the current mobile platforms nor the latest research prototypes fully comply these special requirements, we come up with two extensions for the Privacy Management Platform (PMP) dealing especially with Smartband apps. One of these extensions does not only secure but also facilitate the connection to and data transmission from Bluetooth devices. The other one makes date transmissions to the Internet privacy-aware. This gives users full control over the access to and processing of private data by Smartband apps, as evaluation results show.

However, Smartband apps often do a lot of data processing and analyzing not directly on the user-controlled Smart Devices. Rather, most of the computation takes place at external computing clusters hosted by mainly unknown third-parties. These data stream processing systems have access to a large number of

data sources and resources. Due to this huge amount of data and computing power, they can derive much knowledge about the users. Local privacy settings on the Smart Devices of the users restrict the knowledge extraction of these systems only slightly. Therefore, in addition to an effective privacy system for Smart Devices such as the PMP, an affiliated privacy system for stream processing systems is required. As *PATRON* [56, 57] is highly effective in this area, future works has to investigate how privacy rules for the PMP can be deployed to PATRON. An initial step in this direction is the ACCESSORS permission model [60].

Acknowledgments 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

1. Albaghli, R., Anderson, K.M.: A Vision for Heart Rate Health Through Wearables. In: Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct. pp. 1101–1105. UbiComp '16 (2016)
2. Alpers, S., Pieper, M., Wagner, M.: Herausforderungen bei der Entwicklung von Anwendungen zum Selbstdatenschutz. In: Informatik 2017: Digitale Kulturen, Tagungsband der 47. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 25.9-29.9.2017, Chemnitz. LNI, vol. 275, pp. 1061–1072 (2017), *(in German)*
3. Backes, M., Gerling, S., Hammer, C., Maffei, M., von Styp-Rekowsky, P.: AppGuard: Enforcing User Requirements on Android Apps. In: Proceedings of the 19th International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 543–548. TACAS '13 (2013)
4. Banuri, H., Alam, M., Khan, S., Manzoor, J., Ali, B., Khan, Y., Yaseen, M., Tahir, M.N., Ali, T., Alam, Q., Zhang, X.: An Android Runtime Security Policy Enforcement Framework. Personal and Ubiquitous Computing **16**(6), 631–641 (2012)
5. Barrera, D., Kayacik, H.G., van Oorschot, P.C., Somayaji, A.: 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. pp. 73–84. CCS '10 (2010)
6. Barrera, D., Van Oorschot, P.: Secure Software Installation on Smartphones. IEEE Security and Privacy **9**(3), 42–48 (2011)
7. Beresford, A.R., Rice, A., Skehin, N., Sohan, R.: MockDroid: Trading Privacy for Application Functionality on Smartphones. In: Proceedings of the 12th Workshop on Mobile Computing Systems and Applications. pp. 49–54. HotMobile '11 (2011)
8. Bitsaki, M., Koutras, C., Koutras, G., Leymann, F., Mitschang, B., Nikolaou, C., Siafakas, N., Strauch, S., Tzanakis, N., Wieland, M.: An Integrated mHealth Solution for Enhancing Patients' Health Online. In: Proceedings of the 6th European Conference of the International Federation for Medical and Biological Engineering. pp. 695–698. MBEC '14 (2014)
9. Böhme, R., Grossklags, J.: The Security Cost of Cheap User Interaction. In: Proceedings of the 2011 New Security Paradigms Workshop. pp. 67–82. NSPW '11 (2011)
10. Chan, M., Estève, D., Fourniols, J.Y., Escriba, C., Campo, E.: Smart Wearable Systems: Current Status and Future Challenges. Artificial Intelligence in Medicine **56**(3), 137–156 (2012)

11. Conti, M., Nguyen, V.T.N., Crispo, B.: CRêPE: Context-related Policy Enforcement for Android. In: Proceedings of the 13th International Conference on Information Security. pp. 331–345. ISC '10 (2010)
12. Conti, M., Zachia-Zlatea, I., Crispo, B.: Mind How You Answer Me!: Transparently Authenticating the User of a Smartphone when Answering or Placing a Call. In: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security. pp. 249–259. ASIACCS '11 (2011)
13. Davis, B., Chen, H.: RetroSkeleton: Retrofitting Android Apps. In: Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services. pp. 181–192. MobiSys '13 (2013)
14. Davis, B., Sanders, B., Khodaverdian, A., Chen, H.: 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. pp. 28:1–28:9. MoST '12 (2012)
15. Enck, W., Gilbert, P., Chun, B.G., Cox, L.P., Jung, J., McDaniel, P., Sheth, A.N.: TaintDroid: An Information-flow Tracking System for Realtime Privacy Monitoring on Smartphones. In: Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation. pp. 393–407. OSDI '10 (2010)
16. Felt, A.P., Egelman, S., Finifter, M., Akhawe, D., Wagner, D.: How to Ask for Permission. In: Proceedings of the 7th USENIX Conference on Hot Topics in Security. pp. 1–6. HotSec '12 (2012)
17. Felt, A.P., Ha, E., Egelman, S., Haney, A., Chin, E., Wagner, D.: Android Permissions: User Attention, Comprehension, and Behavior. In: Proceedings of the Eighth Symposium on Usable Privacy and Security. pp. 3:1–3:14. SOUPS '12 (2012)
18. Fragkaki, E., Bauer, L., Jia, L., Swasey, D.: Modeling and Enhancing Android's Permission System. In: Foresti, S., Yung, M., Martinelli, F. (eds.) Computer Security – ESORICS 2012: 17th European Symposium on Research in Computer Security, Pisa, Italy, September 10-12, 2012. Proceedings, pp. 1–18. Springer (2012)
19. Funk, C.: IoT Research - Smartbands. Tech. rep., Kaspersky Lab (Mar 2015), https://securelist.com/analysis/publications/69412/iot-research-smartbands/
20. Giebler, C., Stach, C.: Datenschutzmechanismen für Gesundheitsspiele am Beispiel von Secure Candy Castle. In: Tagungsband der 15. GI-Fachtagung Datenbanksysteme für Business, Technologie und Web. pp. 311–320. BTW '17 (2017), *(in German)*
21. Google Inc.: Permissions Overview. https://developer.android.com/guide/topics/permissions (May 2018)
22. Grace, M., Zhou, Y., Wang, Z., Jiang, X.: Systematic Detection of Capability Leaks in Stock Android Smartphones. In: Proceedings of the 2012 Network and Distributed System Security Symposium. pp. 7/5:1–7/5:15. NDSS '12 (2012)
23. Guo, H.: Privacy Protector (No root). https://play.google.com/store/apps/details?id=net.houzuo.a (Feb 2012)
24. Hornyack, P., Han, S., Jung, J., Schechter, S., Wetherall, D.: 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. pp. 639–652. CCS '11 (2011)
25. Hsu, H.H., Peng, W.J., Shih, T.K., Pai, T.W., Man, K.L.: Smartphone Indoor Localization with Accelerometer and Gyroscope. In: Proceedings of the 2014 17th International Conference on Network-Based Information Systems. pp. 465–469. NBiS '14 (2014)
26. Jeon, J., Micinski, K.K., Vaughan, J.A., Fogel, A., Reddy, N., Foster, J.S., Millstein, T.: Dr. Android and Mr. Hide: Fine-grained Permissions in Android Applications. In:

Proceedings of the Second ACM Workshop on Security and Privacy in Smartphones and Mobile Devices. pp. 3–14. SPSM '12 (2012)
27. Jiang, W., Yin, Z.: Human Activity Recognition Using Wearable Sensors by Deep Convolutional Neural Networks. In: Proceedings of the 23rd ACM International Conference on Multimedia. pp. 1307–1310. MM '15 (2015)
28. Kang, J., Kim, D., Kim, H., Huh, J.H.: Analyzing Unnecessary Permissions Requested by Android Apps Based on Users' Opinions. In: Proceedings of the 2014 International Workshop on Information Security Applications. pp. 68–79. WISA '14 (2014)
29. Khatoon, A., Corcoran, P.: Android permission system and user privacy — A review of concept and approaches. In: Proceedings of the 2017 IEEE 7th International Conference on Consumer Electronics – Berlin. pp. 153–158. ICCE-Berlin '17 (2017)
30. Khorakhun, C., Bhatti, S.N.: mHealth through quantified-self: A user study. In: Proceedings of the 2015 17th International Conference on E-health Networking, Application & Services. pp. 329–335. HealthCom '15 (2015)
31. Kitagawa, M., Glenn, D., Maita, K., Escherich, M., Jump, A., Tay, L., Cozza, R., Atwal, R., Nguyen, T.H., Lakehal, B., Tsai, T., Zimmermann, A., Gupta, A., Lu, C., Wang, A.: Market Share: Final PCs, Ultramobiles and Mobile Phones, All Countries, 1Q18 Update. Tech. rep., Gartner, Inc. (May 2018)
32. Knighten, J., McMillan, S., Chambers, T., Payton, J.: Recognizing Social Gestures with a Wrist-Worn SmartBand. In: Proceedings of the 2015 IEEE International Conference on Pervasive Computing and Communication Workshops. pp. 544–549. WristSense '15 (2015)
33. Knöll, M.: "On the Top of High Towers . . . " Discussing Locations in a Mobile Health Game for Diabetics. In: Proceedings of the 2010 IADIS International Conference Game and Entertainment Technologies. pp. 61–68. MCCSIS '10 (2010)
34. Knöll, M., Moar, M.: On the importance of locations in therapeutic serious games: Review on current health games and how they make use of the urban landscape. In: Proceedings of the 2011 5th International Conference on Pervasive Computing Technologies for Healthcare and Workshops. pp. 538–545. PervasiveHealth '11 (2011)
35. Lee, M., Lee, K., Shim, J., Cho, S.j., Choi, J.: Security Threat on Wearable Services: Empirical Study using a Commercial Smartband. In: Proceedings of the IEEE International Conference on Consumer Electronics-Asia. pp. 1–5. ICCE-Asia '16 (2016)
36. Leontiadis, I., Efstratiou, C., Picone, M., Mascolo, C.: 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. pp. 2:1–2:6. HotMobile '12 (2012)
37. Martin, D., Vicente, O., Vicente, S., Ballesteros, J., Maynar, M.: I Will Prescribe You an App. In: Proceedings of the 2014 Summer Simulation Multiconference. pp. 58:1–58:8. SummerSim '14 (2014)
38. Mayfield, J., Jagielski, K.: FTC Report on Internet of Things Urges Companies to Adopt Best Practices to Address Consumer Privacy and Security Risks. Tech. rep., Federal Trade Commission (Jan 2015), https://www.ftc.gov/news-events/press-releases/2015/01/ftc-report-internet-things-urges-companies-adopt-best-practices
39. Mazzoleni, P., Crispo, B., Sivasubramanian, S., Bertino, E.: XACML Policy Integration Algorithms. ACM Transactions on Information and System Security **11**(1), 4:1–4:29 (2008)
40. Mohamed, I., Patel, D.: Android vs iOS Security: A Comparative Study. In: Proceedings of the 2015 12th International Conference on Information Technology – New Generations. pp. 725–730. ITNG '15 (2015)

41. Nauman, M., Khan, S., Zhang, X.: 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. pp. 328–332. ASIACCS '10 (2010)
42. Patel, M.: The Security and Privacy of Wearable Health and Fitness Devices. Tech. rep., IBM SecurityIntelligence (Sep 2015), https://securityintelligence.com/the-security-and-privacy-of-wearable-health-and-fitness-devices/
43. Pombo, N., Garcia, N.M.: 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. pp. 1–4. WiMob '16 (2016)
44. Reeder, B., David, A.: Health at hand: A systematic review of smart watch uses for health and wellness. Journal of Biomedical Informatics **63**, 269–276 (2016)
45. Ringer, T., Grossman, D., Roesner, F.: AUDACIOUS: User-Driven Access Control with Unmodified Operating Systems. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 204–216. CCS '16 (2016)
46. Russello, G., Crispo, B., Fernandes, E., Zhauniarovich, Y.: YAASE: Yet Another Android Security Extension. In: Proceedings of the 2011 IEEE Third International Conference on Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third International Conference on Social Computing (SocialCom). pp. 1033–1040. PASSAT '11 (2011)
47. Saracino, A., Martinelli, F., Alboreto, G., Dini, G.: Data-Sluice: Fine-grained traffic control for Android application. In: Proceedings of the 2016 IEEE Symposium on Computers and Communication. pp. 702–709. ISCC '16 (2016)
48. Sbîrlea, D., Burke, M.G., Guarnieri, S., Pistoia, M., Sarkar, V.: Automatic Detection of Inter-application Permission Leaks in Android Applications. IBM Journal of Research and Development **57**(6), 10:1–10:12 (2013)
49. Shahmohammadi, F., Hosseini, A., King, C.E., Sarrafzadeh, M.: Smartwatch Based Activity Recognition Using Active Learning. In: Proceedings of the Second IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies. pp. 321–329. CHASE '17 (2017)
50. Siewiorek, D.: Generation Smartphone. IEEE Spectrum **49**(9), 54–58 (2012)
51. Stach, C.: How to Assure Privacy on Android Phones and Devices? In: Proceedings of the 2013 IEEE 14th International Conference on Mobile Data Management. pp. 350–352. MDM '13 (2013)
52. Stach, C.: Wie funktioniert Datenschutz auf Mobilplattformen? In: Informatik 2013: Informatik angepasst an Mensch, Organisation und Umwelt, Tagungsband der 43. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 16.9-20.9.2013, Koblenz. LNI, vol. 220, pp. 2072–2086 (2013), *(in German)*
53. Stach, C.: 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. pp. 167–172. MDM '15 (2015)
54. Stach, C.: Secure Candy Castle — A Prototype for Privacy-Aware mHealth Apps. In: Proceedings of the 2016 IEEE 17th International Conference on Mobile Data Management. pp. 361–364. MDM '16 (2016)
55. Stach, C.: Big Brother is Smart Watching You: Privacy Concerns about Health and Fitness Applications. In: Proceedings of the 4th International Conference on Information Systems Security and Privacy. pp. 13–23. ICISSP '18 (2018)
56. Stach, C., Dürr, F., Mindermann, K., Palanisamy, S.M., Tariq, M.A., Mitschang, B., Wagner, S.: 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. LNI, vol. 275, pp. 1085–1096 (2017), *(in German)*
57. Stach, C., Dürr, F., Mindermann, K., Palanisamy, S.M., Wagner, S.: How a Pattern-based Privacy System Contributes to Improve Context Recognition. In: Proceedings of the 2018 IEEE International Conference on Pervasive Computing and Communications Workshops. pp. 238–243. CoMoRea '18 (2018)
58. Stach, C., Mitschang, B.: Privacy Management for Mobile Platforms – A Review of Concepts and Approaches. In: Proceedings of the 2013 IEEE 14th International Conference on Mobile Data Management. pp. 305–313. MDM '13 (2013)
59. Stach, C., Mitschang, B.: Design and Implementation of the Privacy Management Platform. In: Proceedings of the 2014 IEEE 15th International Conference on Mobile Data Management. pp. 69–72. MDM '14 (2014)
60. Stach, C., Mitschang, B.: ACCESSORS: A Data-Centric Permission Model for the Internet of Things. In: Proceedings of the 4th International Conference on Information Systems Security and Privacy. pp. 30–40. ICISSP '18 (2018)
61. Stach, C., Schlindwein, L.F.M.: Candy Castle — A Prototype for Pervasive Health Games. In: Proceedings of the 2012 IEEE International Conference on Pervasive Computing and Communications Workshops. pp. 501–503. PerCom '12 (2012)
62. Stach, C., Steimle, F., Mitschang, B.: 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. pp. 27–38. HEALTHINF '18 (2018)
63. Stach, C., Steimle, F., Franco da Silva, A.C.: TIROL: The Extensible Interconnectivity Layer for mHealth Applications. In: Damaševičius, R., Mikašytė, V. (eds.) Information and Software Technologies: 23nd International Conference, ICIST 2017, Druskininkai, Lithuania, October 12-14, 2017, Proceedings, pp. 190–202. Springer (2017)
64. Steimle, F., Wieland, M., Mitschang, B., Wagner, S., Leymann, F.: Extended Provisioning, Security and Analysis Techniques for the ECHO Health Data Management System. Computing **99**(2), 183–201 (2017)
65. Wakabayashi, D.: Freed From the iPhone, the Apple Watch Finds a Medical Purpose. The New York Times **12**(27), B1 (2017)
66. Weiser, M.: The Computer for the 21st Century. Scientific American **265**(3), 94–105 (1991)
67. Wijaya, R., Setijadi, A., Mengko, T.L., Mengko, R.K.L.: Heart Rate Data Collecting Using Smart Watch. In: Proceedings of the 2014 IEEE 4th International Conference on System Engineering and Technology. pp. 1–3. ICSET '14 (2014)
68. Xie, H., Gu, T., Tao, X., Lu, J.: A Reliability-Augmented Particle Filter for Magnetic Fingerprinting Based Indoor Localization on Smartphone. IEEE Transactions on Mobile Computing **15**(8), 1877–1892 (2016)
69. Xu, R., Saïdi, H., Anderson, R.: Aurasium: Practical Policy Enforcement for Android Applications. In: Proceedings of the 21st USENIX Security Symposium. pp. 539–552 (2012)
70. Ye, H., Gu, T., Tao, X., Lu, J.: Scalable Floor Localization Using Barometer on Smartphone. Wireless Communications & Mobile Computing **16**(16), 2557–2571 (2016)
71. Zhang, D., Wang, R., Lin, Z., Guo, D., Cao, X.: IacDroid: Preventing Inter-App Communication capability leaks in Android. In: Proceedings of the 2016 IEEE Symposium on Computers and Communication. pp. 443–449. ISCC '16 (2016)

All links were last followed on June 1, 2018.