Health Monitors Under The Magnifying Glass: A Privacy And Security Study

Martin J. Krämer

Master of Science
Computer Science
School of Informatics
University of Edinburgh
2016

Abstract

The adoption of mHealth in well-being solutions and medical monitoring advances very quickly. Already mHealth data is not only used for doctor consultation or self-monitoring but also as evidence taken to the courtroom. At the same time security incidents and privacy issues happen regularly. Less often people are aware of related risks. Though past technology trends have shown that security and privacy issues in software and systems follow repeating pattern.

The starting point of this thesis is the postulation that given the assumption such issues exist, they can be revealed through thorough analysis. To that end an analysis framework for mHealth solutions, *mH-PriSe*, is proposed. The adequacy of this framework is validated through a comprehensive analysis of 8 different smart scale solutions which have been released since 2012.

The framework presented in this report allowed for the discovery of weaknesses affecting all solutions in scope. Among them the best performing have issues in about 45% of all applicable test steps only, whereas the worst solution fails in almost 90%. Solutions which perform well show strengths in security and privacy related aspects. An example for strong security is certificate pinning. For privacy aspects such solutions allow for fine grained privacy settings that control how data is shared with others and provide full control to the user over their data. In contrast, solutions performing worse exhibit many security and privacy issues. Privacy concerns arise from missing privacy policies, through data leakage or the absence of user control over the data. Security issues range from unencrypted traffic or improper certificate validation to un-salted password hashes which ultimately leak the user's password.

As many of these issues can be linked to prior research and entries in the Common Weakness Enumeration (CWE) database, they indeed relate to well-known problems. Hence the evidence found through experimentation is sufficient to confirm the hypotheses. Although newer solutions advance by addressing some of these issues and improvements are recognised, the findings and methodology is published for further use by the research community. In a more accessible way to consumers, the alarming results will also published on a web page after they have been disclosed with vendors.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Martin J. Krämer*)

Acknowledgements

My gratitude goes to my supervisor, Professor David Aspinall, for his supervision and guidance during the past months. My thankfulness is also his for encouraging me to pursue my academic interests and for supporting my applications for PhD positions.
This thesis marks the end of an incredibly enriching year, full of hard work and fun. While many acquaintances naturally line the way of master studies, being able to call some of them friends, I consider myself to be very fortunate. To these two exceptional yet for my (simple) mind unconventional characters who allowed me to learn for life beyond studies, a big thank you. It is my pleasure to have met you.
Also many thanks to my friends in Edinburgh who made this year special and more than just hard work. And of course thanks to my friends in other places who dedicated some of their time to review and discuss this work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement and Motivation

Recent studies show an increase in the adoption of mobile technology in the Health Sector [1, 2]. The reports show that patients start to make use of their Electronic Health Record (EHR) which includes information such as blood values or X-rays. Furthermore remarkable growth in the mHealth market and an increased user adoption and willingness to use wearable devices is pointed out [1, p. 24]. According to the study doctors and patients both recognize and appreciate the opportunity for patients to provide self-measured metrics [1, p. 22]. In fact in the US 21% of adults use some form of technology to track their health data [3].
Based on the findings of these studies patients seem to be willing to share health related data they have collected with their doctors. As a result patients nowadays take over tasks that otherwise would have been done by nurses or doctors themselves [4]. Both doctors and patients benefit from such a solution allowing doctors to focus on the actual expertise in consultation and patients to execute their medical routines independently and efficiently. Recent news show further usage of that data as it might even be considered as evidence in the courtroom [5]. With the willingness to share personal data with third parties privacy concerns inherently arise. Interestingly, while many people seem to be aware of privacy impacts when they publish personal information on social media [6] they rarely seem to understand the impacts on privacy of data being continuously collected by their mobile health (mHealth) solutions, such as activity trackers [7]. This is particularly remarkable given that just recently researchers discovered and exploited vulnerabilities in popular devices such as the FitBit activity trackers [8–10] or a Withings scale [11].

In addition many mHealth solutions typically synchronize collected data with an online database. Producers of mHealth solutions provide online social networks which often interact with Facebook, Twitter or other social platforms. Many data breaches have shown that where valuable information is collected, there will always be people with malicious intents interested in that data. There is sample evidence to support the claim that data breaches, harming user privacy, regularly happen [12, 13].

1.2 Hypotheses, Objective and Research Questions

Hence to protect users security and privacy there is an urgent need to investigate available solutions to discover vulnerabilities, improve on development processes and create risk awareness. Therefore the key objective of this project is to understand how security and privacy are taken into consideration by popular mHealth solutions among different vendors. Although mHealth has become a active area for security research, approaches to find weaknesses and reveal vulnerabilities are barely based on sound scientific methodologies. In their paper "Security Testing for Android mHealth apps" Knorr et al. are the first to propose a testing methodology for mHealth solutions. Their work is limited to the investigation of mobile apps but excludes the examination of entire solutions. This project expands on their work to further close the gap in research. Therefore the following hypothesis are to be confirmed or refuted:

Hypothesis 1 *mHealth solutions exhibit security and privacy issues well-known from other areas such as web application development or enterprise software development, such as those listed in the Common-Weakness-Enumeration (CWE) or violating coding standards by CWE, CERT[1], MITRE[2], OWASP[3]*

Hypothesis 2 *Such issues can be detected using an analysis framework specifically suited for mHealth and allowing for methodological security testing*

In the following in order to examine the hypotheses this project will investigate the market of mHealth solutions horizontally by comparing solutions addressing the same need but offered by different vendors. The present thesis aims to analyse the impacts

[1]Carnegie Mellon Universities Cyber Emergency Response Team and related groups
[2]Coporation running Cyber Security and more research for the US Government
[3]The community run Open Web Application Security Project

of such solutions on security and privacy to raise risk awareness, provide recommendations and give guidelines. To achieve the goals this project aims to answer the following research questions:

Q 1 *Security – How secure is the design and implementation of the solution?*

- **Q 1.1** *Which commonly known weaknesses does the solution reveal?*
- **Q 1.2** *Which data does the solution collect and what are the characteristics of that data?*

The discovered security issues most certainly will have impact on the user's privacy. This impact is the first of two factors of privacy in focus. The second is based on vendor provided documentation on the solution, e.g. the privacy policy.

Q 2 *Privacy – Which privacy impacts can be assumed?*

- **Q 2.1** *Which privacy impacts can be assumed based on the collected data and detected risks?*
- **Q 2.2** *Does the privacy policy comply with the findings?*

The security and privacy properties that have been investigated serve as a basis for a comparison between the different solutions in scope. Different aspects for different purposes will be considered (see also 1.3.

Q 3 *Comparison – How compare different solutions?*

- **Q 3.1** *How do the investigated solutions compare in terms of privacy and security?*
- **Q 3.2** *Which general recommendations can be deduced from these findings?*

Ultimately the goal is to line out the differences between the vendors' claims on privacy and security in contrast to the actual behaviour of these solutions.

1.3 Research Process and Data Collection

In the following the research process and method employed to answer the two main research questions will be described. Based on a comprehensive data collection findings to answer the research questions will be retrieved.

Security – Research Question Q 1 To answer this question a comprehensive security test suite for mHealth solutions is designed and will expand on the work of Knorr et al. [14]. The test suite embraces static and dynamic analysis including data flow analysis as well as basic penetration testing. All tests will be set up with newly provisioned accounts to assure quality of results. Finally revealed weaknesses are grouped and per group a ranking is assigned.

Privacy – Research Question Q 2 Privacy policies are analysed and compared with the findings for Q 1. In addition privacy impacting findings are extracted and defined as additional criteria for further evaluation.

Comparison – Research Question Q 3 Results of this analysis build the foundation for a comparison between different solutions and recommendations to improve mHealth development towards more security and less privacy intrusiveness.

The nature of main the research questions results in the following research process: First, the security analysis is carried out using the mentioned toolset. Based on the data which is gathered during that process, the second step will be to analyse related risks and assess the impact on privacy of these security issues. This is the foundation for recommendations and guidelines which are given as a result of this study.

1.4 Scope and Contribution

This project focusses on investigating security and privacy issues of mHealth solutions which are related to either a mobile application, a corresponding web server or the data transmission between them. The analysis includes static and dynamic analysis of the mobile application, the data transmission between web server and mobile application as well as penetration testing for aforementioned parts and the web server itself. Technical analysis such as firmware analysis, of the mHealth device itself, the tracker or sensor, is not within the scope for this project. The traffic analysis focusses on WiFi only, excluding Bluetooth communication. Meanwhile most mHealth solutions support iOS and Andorid as mobile operating systems this project focusses on Android applications only.
To better understand the scope of this project the following perspectives can be assumed. They represent different interests in studying mHealth solutions.

Security Analyst Security analysts are typically interested in weaknesses and vulnerabilities of products and solutions. To detect them they wish to use an automated test suite, which reveals such issues to some level of detail. Hence this report includes a detailed description of the test set up and execution for future reuse.

Software Developer Many security issues occur due to faulty software development. This is not necessarily due to the software developer providing bad quality in coding, but can also be blamed on too complex security protocols and mechanisms which are often hard to implement. To mitigate the risk this report provides a comparative analysis, recommendations and guidelines for developers.

End User The end user of mHealth solutions, that is the average customer, who typically has no specific knowledge about IT security, depends on guidelines and recommendations as well. Furthermore customers depend on researchers to reveal security and privacy issues of solutions they intend to buy or and use. To this end the project contributes to public interests.

1.5 Structure and Methodology

There is no general approach to formalise security research, but rigorous experimental research is accepted as the standard method within the community [15]. "It is the lack of substantial theories that hampers progress in our field" [15]. Hence cyber security research hardly fits into any common methodology framework such as [16]. Consequently the methodology chosen for this project is although common in security research, less defined in literature. Peisert et al. defined basic criteria such as *objectivity*, *falsifiability*, *controllability* and *reproducibility* [17] (see also [18, p. 8]). In other words research has to follow objective experiments considering all data without a bias towards confirming the hypothesis, a postulated hypothesis must be refutable, experiments are required to change one parameter at a time and others must be able to reproduce results through rerunning experiments. Hence a major part of this project is dedicated to software assurance analysis by proposing *mH-PriSe*, a framework for methodological privacy and security analysis. Described in more detail in Chapter 3 this project follows a combination of static analysis, dynamic analysis and penetration testing [18, ch. 4].
This thesis follows a structure similar to "The Scientific Method" mentioned in [17, 18]. The rest of this report is structured as follows: Chapter 2 explains and discusses relevant background information that is related to mHealth solutions with pointers given to more extensive information. In Chapter 3 the comprehensive security test-

ing methodology, *mH-PriSe*, and its setup used to investigate the mHealth solutions is defined and described. Chapter 4 is a documentation of case studies which have been carried out using the security testing methodology defined in Chapter 3. In Chapter 5 a discussion and measurement against the postulated hypothesis and research questions follows and the report concludes with Chapter 6 summarizing the work done and outlining further research questions and topics that should be explored further.

1.6 Summary of Results

The *mH-PriSe* framework, proposed as part of this project, allowed for detection of security and privacy issues in all solutions. The alarming results show that out of 9 solutions in scope, all showed at least minor security and/or privacy issues.

- Out of 9 solutions, where a solution consists of a smart scale, paired with a companion app and synchronizing data to a vendor supplied web page, 7 solutions are associated with severe security and privacy issues
- The remaining 2 solutions, the iChoice S1 with SwissMed mobile application and the Withings Body Cardio with the Withings application, pass the tests and investigations with some caveats
- Some of the main issues are known from other areas such as web application development and include no SSL usage, improper certificate validation, weak password policies or improper cryptography usage
- Other issues are related to the mobile nature of such solutions, including *over-pivileged* mobile applications or leakage of device/user identifying data (e.g. IMEI or MAC)
- Privacy policies are measured against OECD privacy principles and although are in general of a good standard, can be further improved in all cases
- In general, vendors can improve on privacy preserving functionality, i.e. user data management (account deactivation or deletion), location of data storage and information on data usage

Chapter 2

Mobile Health: System and Applications

2.1 Related Work

The work of Knorr et al. is closely related to this project as this project was designed to expand on their work [14, 19]. They investigated mHealth mobile applications comprehensively with regards to security, safety and privacy. Their study comprises four parts: (i) dynamic and (ii) static analysis of mobile applications, (iii) web server connection analysis and (iv) privacy policy inspection. However they were focussing on the mobile application only without actually connecting a sensor. This project adds to the study of Knorr et al. by investigating the market of mHealth solutions horizontally. This implies that as opposed to Knorr et al. working mHealth solutions including sensor device, mobile application respectively mobile device and web server will be investigated. Whereas Knorr et al. focus on mobile applications [14, 19].
There have been similar studies on mHealth application security which are also mentioned by Knorr et al. such as [20–23] all of which are looking into security issues with mHealth apps. They find that SSL encryption is broken [20, 21, 23] and privacy aspects are often not prioritized during development. The work of Mense et al. provides even more detail on the behaviour of mHealth application with respect to privacy [24] especially on which data is being sent. Huckvale et al. similarly analysed mHealths apps within a time frame of 6 months from 2013 - 2014 [25]. Baig et al. recently explored the research area of mHealth applications by reviewing system design and the identification of challenges and issues. Among those the biggest according to Baig et al. are security, privacy and safety [26].

Other research provides more comprehensive studies on privacy and security of single mHealth solutions but limit their research and evaluation to one or just a few selected solutions [10, 27]. They explore the Fitbit solution [28] which has been investigated by several other researchers as well but with different focus [8, 9, 29]. Similar research on mHealth solutions covers different areas such as over-the-air-attacks on fitness devices [10, 27, 30] and health devices [31, 32] or sophisticated reverse engineering on firmware and protocols [11, 33]. From this perspective closest to this project is the work of Clausing et al. and Hilts et al. who analyse and compare different fitness tracker [30, 34]. Hence this study is the first to propose a structured approach for future research and to apply this methodology to a horizontal study of mHealth by investigating 8 smart scales.

2.2 Mobile Health Solution

2.2.1 Mobile Health and Relevant Definitions

Mobile Health (mHealth) is loosely defined as "medical and public health practice supported by mobile devices, such as mobile phones, patient monitoring devices, personal digital assistants (PDAs), and other wireless devices" [35]. Use cases in mHealth are of any one of the two broad categories (i) Electronic Health Record (EHR) or (ii) Personal Health Record (PHR). The latter is managed by the patient or consumer whereas EHR are typically maintained by healthcare providers. Within PHR a subset of use cases can be defined as "vendor-supplied PHR". These typically offer access to collected data via an application specific website that is hosted on the vendor's server [21, 36]. The present thesis is concerned with such use cases, precisely solutions from the categories of medical or fitness & wellness are investigated [37]. Examples for the fitness & wellness are activity tracker such as the Fitbit flex [28]. Medical solutions include many different devices for example scales [38–40], devices for blood value measurement [41] or thermometer [42]. Probably the very first connected health device was the Nike+ SportKit which worked with the Apple iPod as a running kit. Already within a year after it's release researchers found vulnerabilities [43].
For the reminder of this report a few terms have to be defined. This report follows the definitions of the US National Committee on Vital and Health Statistics (NCVHS) [44]. According to them health information privacy is defined "an individual's right to control the acquisition, uses, or disclosures of his or her identifiable health data" [44, p.3].

They understand confidentiality as "the obligations of those who receive information to respect the privacy interests of those to whom the data relate" (ibid.) and security as "physical, technological, or administrative safeguards or tools used to protect identifiable health data from unwarranted access or disclosure" (ibid.). In contrast, safety refers to the physical health and well-being of patients [45]. Although closely related it is not further considered for this work.

2.2.2 Solution Architecture

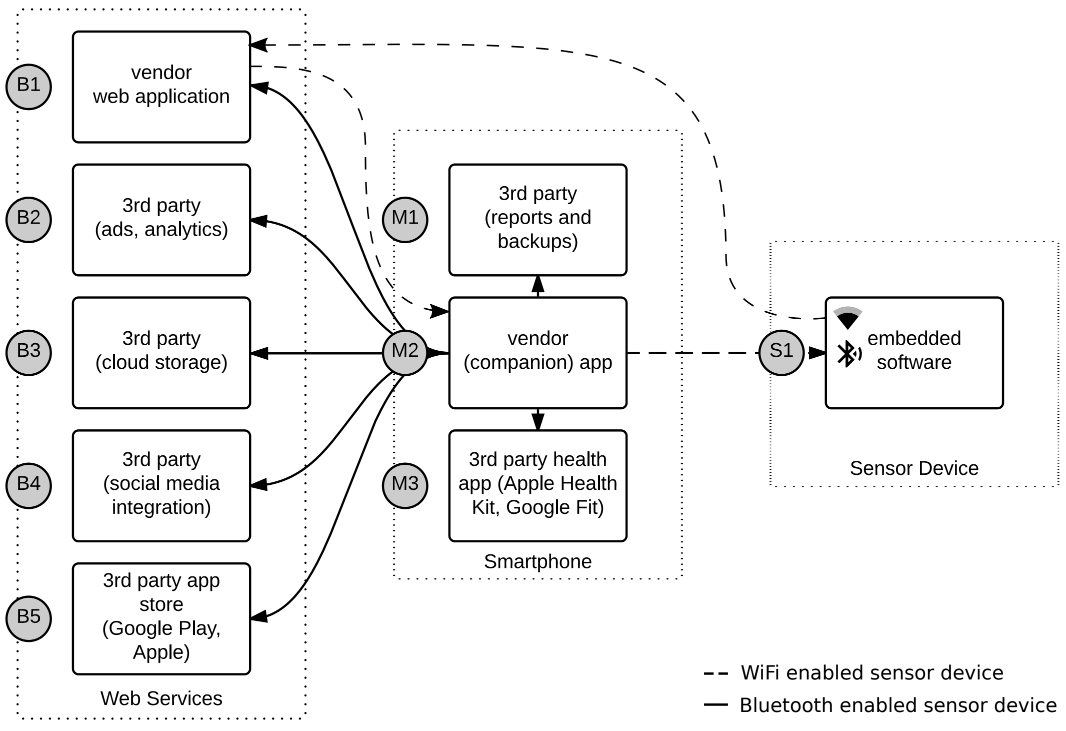

B1
vendor web application
B2
3rd party (ads, analytics)
B3
3rd party (cloud storage)
B4
3rd party (social media integration)
B5
3rd party app store (Google Play, Apple)
Web Services
M1
3rd party (reports and backups)
M2
vendor (companion) app
M3
3rd party health app (Apple Health Kit, Google Fit)
Smartphone
S1
embedded software
Sensor Device
-- WiFi enabled sensor device
— Bluetooth enabled sensor device

Figure 2.1: mHealth logical architecture following [14]

Most mHealth solutions follow a similar architecture. From a vendor's point of view, Fig. 2.1 depicts a logical view on a typical mHealth architecture including a sensor device to gather data, a mobile application to store and aggregate data, as well as a web based application. In addition the figure shows 3rd parties which may be involved with the solution. Labels (B1-B5) reference internet based services connected to the solution. Services reach from (B2) advertisement over (B3) cloud storage (backups, data transfer) to (B4) social media integration (sharing, login services) [20, 24]. (M1-M3) reference components with respect to the mobile device: (M1) reports or backups might be exported, (M2) a vendor provided mobile companion app to run the solution

and (M3) OS based health apps integrate via an API [14, 23]. The arrows symbolize data flows. Often not all of these data flows are obvious to a user, nor are they mentioned in privacy policies [20]. The majority of mHealth sensor devices support communication over Bluetooth only. However a few devices offer both, Bluetooth and WiFi connectivity, and others offer only WiFi connectivity. The first mentioned design is the most common variant among solutions in scope. Other architectures are discussed in *"Models of self-tracking systems"* [23] but not relevant for this work.
Typically data is synchronised from the sensor device to a mobile application and then from the mobile application to a web server (3-tier architecture). Though, there are arguments in favour of other architectures. Worth mentioning is the case of FitBit and their popular activity trackers. When Fitbit was released they drew the attention of research [5, 23, 27, 29, 33, 46–48] and industry [49]. After Rahman et al. discovered several security vulnerabilities related to their architecture [8], Fitbit decided to change their architecture from synchronising data between wrist band and mobile device, to transmitting all data from wrist band to their web application [33] (see Fig. 2.2). The communication still runs through Fitbit's mobile app as a proxy. Presumably this change was made in favour of an increased security since all communication could be encrypted on the wrist band and only decrypted by their web application. However the mobile application is required to communicate with the web application extensively for data analytic purposes. Other comparable solutions store data on mobile phones [30]. Contrary to this, there is also an argument for storing data offline on mobile devices. This is in favour of preserving data privacy with the data not being transmitted to any of the internet based services (B1-B5) and hence the user would be in full control of the data [48]. In any case, however, as will be discussed in 2.3 one attack surface is the mobile phone.

2.2.3 Technical Infrastructure

As a part of the Internet-of-Things mHealth can be expected to be based on many different technologies and protocols [50]. It turns out that technologies and protocols used in the area of mHealth mostly rely on Bluetooth (Low Energy) [51, 52] for data transmission, some additionally support Wireless Lan [53] and others the rather proprietary ANT+ protocol [54] (see Figure 2.2). Nevertheless researchers are investigating mHealth device networks with the aim to define new standards. The following publications are good starting points for more information on networks: wireless sensor

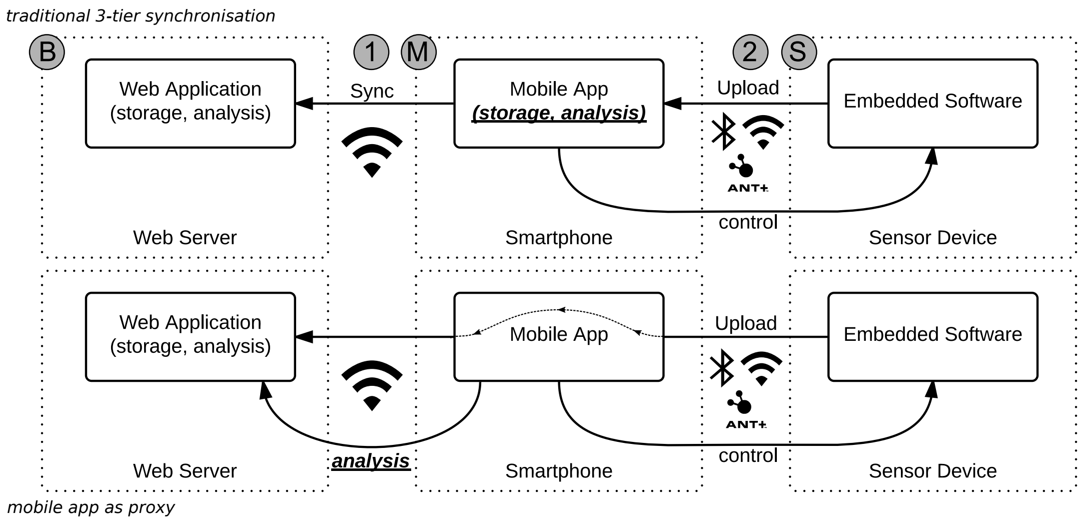


Figure 2.2: mHealth physical architecture following [33]

networks [55], Wireless Media Sensor Networks (WSM) [56], Wireless Body Area Network (WBAN) [57, 58], ANT+ based networks [59].
In the subsequent paragraph an overview of security or privacy related publications on each component of the architecture depicted in mHealth physical architecture. As there is extensive information available this overview is by no means complete, but highlights relevant literature.

(B) Web Application & Web Server Web applications are traditionally subject to extensive exploration through researchers and hackers. Hence there are well established guidelines to facilitate secure development and tool sets to test for security issues. The OWASP web testing projects such as their Zad Attack Proxy Project are exemplary [60].

(M) Mobile Application & Mobile Device Android OS and its applications available through Google Play Store have been researched extensively with respect to security. Sufatrio et al. published a survey which provides a very good overview [61]. Many research tools have been published which can be used for Android application reverse engineering and security analysis [14] – also see *3 mHealth PriSe*. A more detailed description is given by Gupta et al. and Drake et al. in their books on how to learn penetration testing Android devices [62, 63]. Misra provide an overview on Android security in general [64].

(S) Embedded Software & Sensor Device Although this part is out of scope for the project, the following information represents a good starting point for further exploration. A paper by Kim et al. and blog posts by Collado provide information on tools and methodology [65, 66]. More tangible examples of studies show how wearable devices can be exploited by firmware replacement through update channel attacks [11, 33, 67].

(1) Network The communication between mobile application and web application – the vendor's web server or 3rd party server – is exposed to different kinds of over-the-air (ota) attacks. The communication protocol is likely to be Hyper Text Transfer Protocol (HTTP) and should be encrypted using Secure Socket Layer (SSL) [68]. However researchers have found out that SSL is often badly implemented [69] and that particularly regarding Android applications, bad implementation causes serious security vulnerabilities [70, 71]. These vulnerabilities enable so-called Man In The Middle Attack (MITMA) (see *2.2.4 Eavesdropping on Communication*) which can result in "credential stealing or arbitrary code execution" [72].

(2) IEEE 802.11, Bluetooth Low Energy (BLE), ANT+ Sensor device and mobile application communicate using one of the mentioned protocols. If the devices communicate using Wireless Lan (WLAN), the aforementioned applies. Research on BLE revealed weaknesses such as incomplete implementations in several products [73, 74]. With its history there are many tools available to attack Bluetooth devices. Dunning provides a good overview on Bluetooth threats and related tools [75]. For ANT+ Rahman et al. describe a possible attack [8].

2.2.4 Eavesdropping on Communication

The concept of eavesdropping is at the centre of this project and hence will be explained in more detail. A communication between two parties `A` and `B` is said to be subject to Man In The Middle Attack (MITMA) if a third party `C` manages to successfully intercept their communication and is able to learn any information from the data he captured [76, ch. 6.2]. This access to the communication stream is also referred to as "eavesdropping" on the connection.

If data between parties `A` and `B` is unencrypted eavesdropping is easily possible for an attacker that controls a network. In fact, as a friendly use case for MITMA, it is common for public hotspots to intercept the communication of their users, e.g. to present a login page or to enforce the blocking of specific sites. This applies to wired as well as wireless networks. Wireless networks, however, enable attackers to eavesdrop on data streams even when not connected to the network, by switching their (wireless) network cards to promiscuous mode and hence listening to all traffic send – the same works for wired networks but is limited to the subnet the attacker is located in. Assuming an attacker is in full control of the network, SSL/TLS (referred to as SSL hereafter) is the means of choice to defeat MITMA, if it is properly used.

SSL, however, has been known to be vulnerable to various security attacks such as the famous Heartbleed Bug [77], the FREAK attack [78] and many more. In the case of Android researchers have shown issues with proper certificate validation (CWE-295) [70, 79]. For Android these issues originate mostly from flawed SSL implementations where applications, intentionally or unintentionally, trust all certificates, miss to validate the certificate chain or skip certificate validation entirely. While the details are not relevant for this work the paper of Fahl et al. discusses them in more detail [70]. These vulnerabilities allow attackers to successfully forge certificates, that is to intercept a SSL handshake and establish themselves as a Man-In-The-Middle between the two communicating parties. Broadly spoken an attacker simply pretends to `A` to be `B` and to `B` to be `A`. If the parties correctly validated certificates this type of attack would fail, but if they do not, eavesdropping is possible. An example for such an attack is discussed later in this report (see *4.2.2 Getting SSL Right*).

2.3 Mobile Health Threat Model

2.3.1 Assets and Agents

To understand risks and impacts arising from threats and vulnerabilities, it is common practice to define a threat model. As a part of that model assets have to be defined. They can be separated into categories as follows [80].

AS 1 *PII* – Personal Identifiable Information (PII) is a term used in US law (personal data in EU law) to describe any information that can be used to reveal an individuals identity [81]. In general this refers to identification and contact information such as address or phone number.

AS 2 *Health data* – Any measurements and information send from mHealth solutions such as activity times, weight, blood glucose levels and more.

AS 3 *Technical* – Technical information on phone or sensor device such as

IMEI, MAC, location data and more (all of which is data that will allow to track a user remotely over short or longer distances)

De et al. define more categories than the aforementioned. For the purpose of this project the three mentioned categories are sufficient. As a next step in defining the threat model potential attackers posing threats on assets are identified. Threats to an asset can be understood to have the objective to harm any one or more of availability, confidentiality, integrity, accountability and authentication. The action of harming a user's assets can be executed by interception, modification, interruption or fabrication [76, p. 34 ff.].
The following list is composed of threat agents mentioned in different papers. Most importantly the work of Knorr et al. and Dhanjani build the basis [14, 74]. Moreover Kotz distinguishes between insiders and outsiders. Insiders are defined as individuals with authorized access to data and information. In contrast, outsiders have to gain unauthorized access [36]. All agents mentioned in this list can be classified as outsider with the exception of the user himself. Outsiders are excluded form this list as investigation of threats they might pose and weaknesses they might exploit is not feasible without access to a vendors internal systems and business processes.

AG 1	*Investigators* – they all aim to harm a user's privacy by learning any kind of information [14, 74]
AG 2	*Hacktivists/Cyberbullies* – with a malicious intent they aim to disclose information about the user [14, 74]
AG 3	*National State Agencies* – data intelligence companies want to make profit based on information they learn about individuals [14, 74]
AG 4	*Employers* – want to gather information on employees or applicants [14, 74]
AG 5	*Health Insurance Companies* – aim to analyse data to retrieve more information about their customers health conditions [14]
AG 6	*Terrorists* – are also worth considering as attacks [82, 83] on mHealth devices might maximise the impact of attacks [74]
AG 7	*User* – have interest in their data to be accurate, but might also be willing to manipulate data for personal profit [33]

AG 8 *Social Environment* – family, close friends, co-workers or acquaintances aiming to humiliate or even harm the victim

2.3.2 Weaknesses

Following the definition of Gollmann weaknesses become vulnerabilities through intentional or accidental exploitation [84, p. 24]. In the context of this work only weaknesses in mHealth are discussed. Weaknesses relevant to this project can be grouped into security and privacy weaknesses, although they are often closely related. This means that whenever a security weakness is discovered one can assume privacy impacts and vice versa. Kotz as well as Avancha et al. provide an comprehensive literature survey and define a threat taxonomy for privacy threats [36, 85]. Furthermore weaknesses from a security point of view are mentioned in numerous research papers. The following listing provides an overview and is based on [8, 14, 36, 56, 85].

W 1 *Privacy Weaknesses*

W 1.1 *Identity* – Authentication might be harmed in cases where credentials are stolen to commit fraud or identity theft; anonymity might be harmed if researchers manage to link anonymised data back to individuals

W 1.2 *Access* – Should be limited to some reasonable extend: user consent to use and share data, access control, audit and data integrity assurance; in case access is not regulated carefully information might be used for humiliation and embarrassment, or the manipulation of records for financial gain.

W 1.3 *Disclosure* – Of information (unwanted) where information is transmitted via an insecure way; user want to hide existence of devices as they might allow to guess a disease, or a device might be stolen or compromised by Malware

W 2 *Security Weaknesses*

W 2.1 *Tampering* – Through MITMA data can be manipulated or replayed by an attacker

W 2.2 *(Invalid Reporting)* – For completeness although more related to safety: an attacker may manage to report wrong data manually or through physical attacks to devices; e.g. attaching an activity tracker to a car wheel and driving around to record activity [8].

W 2.3 *Location Tracking* – Based on GPS tracking data or other context information the user's location might be revealed

W 2.4 *Denial-of-Service (DoS)* – Live sustaining systems might be subject to DoS attacks

W 2.5 *Eavesdropping* – Through MITMA on vital signs and other data (learning of information)

2.3.3 Attack Surface & Vectors

The attack surface of a mHealth solution includes every involved component and the communication between those components. Figure 2.3 depicts the full landscape and highlights relevant attack vectors. The following list is based on the work of Knorr et al. [14] and He et al. [21]. Additional work is mentioned where relevant.

AV 1 *Data Transmission* Communication over public channels (wireless or wired) may be subject to eavesdropping and tampering; this applies for any wireless communication including Bluetooth and others but is also the case for wired networks where an attacker might control the entire network

AV 2 *Mobile Device* Pyhsical access (unencrypted database, logfiles [20]), other apps (e.g. Malware) through missing application policy for private phones [86], side channel attacks on physical layer communication to infer knowledge through pattern recognition

AV 3 *Mobile App* Various vulnerabilities are known (see 2.2.3) – Android intents, permissions, dynamic content loading [10], Transport Layer Security (TLS)/SSL weaknesses, advertising libraries [87]
3rd Party Forging fitness results for financial gain [8], information send to and received from advertising companies, app updates triggered through app shops

AV 4 *Sensor Device* Intercepting the update mechanism [33] or physical access to firmware via JTAG (reverse engineering) [11]

AV 5 *Web Application* Vendor-supplied web applications for data storage, analysis and sharing information are subject to known web attacks [88]

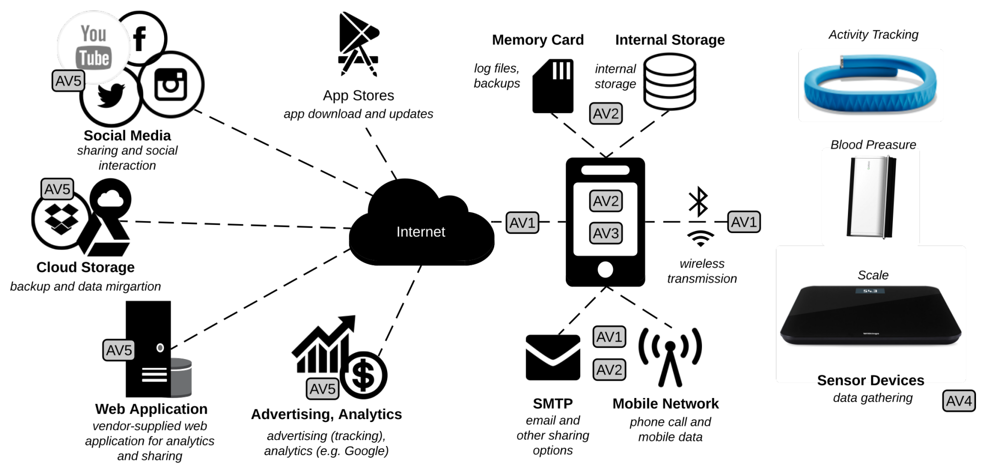


Figure 2.3: Threat landscape

2.4 Data Privacy in Mobile Health

2.4.1 Health Information Privacy

Patient privacy protection is well established for the conventional health care systems. Governmental organizations and regulating bodies are responsible to protect patient's privacy and security. The privacy of an individuals health information has been defined as "[...] an individual's right to control the acquisition, uses, or disclosures of his or her identifiable health data [...]" [44]. The aim of any law or regulation is to protect Personal Identifiable Information (PII) from disclosure.

Across the US, UK and EU regulating authorities have worked on privacy regulations and laws for the past few years [89–91]. Current laws and regulations such as the Health Insurance Portability and Accountability Act (HIPAA) in the US or the EU Data Protection Directive 95/46/EC date back 20 years and have to be reviewed. A comprehensive study by Alliance showed that most of national laws and regulations across North America, Europe and Asia are not covering the full scope of medical, fitness & well-being apps and solutions [92]. To clarify responsibilities between authorities, often the challenge for organizations is to understand the new technology used for health-care and the many different solutions available [45, 93]. Common among legislation is a risk classification of mHealth solutions. Table 2.2 provides an overview of such commonly understood risk classifications. Hence whereas medium security solutions such as any blood value reporting solution may become subject to regulation, fitness and well-being solutions will not be covered by regulations [90, 94, 95]. Nevertheless regulating authorities have recognized the quickly [96] evolving and

changing market and hence addressed this issue with guidelines for vendor and user: e.g. in the EU [97], and in the US [98]. Moreover the EU directives also address issues of cloud data storage and in that context define that data is only allowed to be stored on servers outside of the EU if the country in which the destination is located applies comparable privacy regulations [97]. The EU US "Safe Harbor" agreement allowed US companies to self-certify and transmit data outside the EU. However due to the Snowden revelations the agreement was invalidated and is now to be replaced by the new "Privacy Shield" framework for personal data exchange [99].
Still the gap in legislation remains. Although there are proposals on how to address these issues [100], to-date it is up to manufacturers to carefully implement security and privacy into their products [101]. As discussed this is something many vendors struggle with and hence research and work is required to address this issue [25].

Security Level	**Description**	**Device Examples**
Low	Neither sensitive nor safety-critical activity	PC in hospital used for administrative work, Heart rate watch
Medium	Sensitive activity	PC processing EHR, Smartphone communicating glucose levels
High	Safety-critical activity	Device controlling insulin pump or sending parameters to pacemaker
Very High	Safety-critical activity, input from elsewhere	Pacemaker receiving external parameters

Table 2.2: Medical device classification following [94]

2.4.2 Privacy Principles

Although privacy protection legislation does not cover many mHealth solutions, understanding privacy implications is important [47, 102]. As Ballano Barcena et al. highlight the data collected by such solutions is different from traditional personal data but that an attacker with the right knowledge and the right data would eventually be able to identify the user anyway. That is an attacker might be able to find out about an individuals name and address by e.g. tracking back location information [23].
In 1981 the OECD released guidelines "on the Protection of Privacy and Transborder Flows of Personal Data" [81]. These guidelines define 8 principles which are seen as the base-layer for many subsequent privacy guidelines:

PP 1	*Collection Limitation* – Data collection should be limited to a necessary minimum and this collection should follow established rights and regulations
PP 2	*Data Quality* – Data should fit the purpose, be accurate, complete and up-to-date
PP 3	*Purpose Specification* – The intended purpose of the data should be stated at the time it is collected and any subsequent use should not be different from the intentional use
PP 4	*Use Limitation* – Data should not be published or used in a different way than stated, except from overruling by law
PP 5	*Security Safeguards* – Data has to be protected against common risks following the basic security principles authorization, integrity, accountability and availability
PP 6	*Openness* – The existence, the intended use and the identity and residence of the data controller should be disclosed
PP 7	*Individual Participation* – The individual is preserved the full control over the data upon request; this involves the request of data, retrieval of data and to challenge data validity
PP 8	*Accountability* – The data controller is accountable for any actions to guarantee aforementioned principles

Often guidelines like these are used to examine the quality of privacy policies if such policies are available. In fact researchers have been looking at mHealth applications from a privacy perspective and have checked available policies. Unsurprisingly the availability for a specific solution and the quality of privacy policies in general is low as a study by Sunyaev et al. shows [103]. Just 30% of available apps in relevant application store categories posses a privacy policy. Even worse, a study by Knorr et al. revealed that just 19% of analysed apps of the medical category (medium risk) provide a privacy policy [19]. Further studies showed that many applications and solutions either lack a privacy policy [20, 23, 87], show gaps in their coverage [19], or reveal privacy principle violating behaviour [48].

Chapter 3

mHealth PriSe
A Privacy and Security Analysis Framework

3.1 Overview

As part of this work a privacy and security analysis framework for mHealth "mHealth PriSe" or short "*mH-PriSe* " is proposed. With respect to the solution architecture defined in *2.2.2 Solution Architecture* and the threat model defined in *2.3 Mobile Health Threat Model* for mHealth solutions *mH-PriSe* has been designed and implemented. In this chapter a brief discussion of the proposed framework is provided. A detailed functional description of incorporated tools and techniques is available from the Appendix Tables A.5 to A.10. Aforementioned criteria for security research have been considered following the work of Peisert et al. as *objectivity*, *falsifiability*, *controllability or verifiability* and *reproducibility* (see also [18, p. 8]).
From a functional point of view the following techniques are in the centre of a wide range of software security assurance testing techniques [104], [18, p. 43], [64, p. 106 ff.]:

- *static analysis* – analysing a program's development artefacts without actually running the program
- *dynamic analysis* – running the program and observing the runtime behaviour
- *fuzz testing* – providing random inputs to programs in an attempt to crash the program

- *penetration testing* – attacking the system in order to discover security weaknesses

For the purpose of this research, namely to investigate a solution towards security issues and their privacy impacts, this study is limited to static analysis, dynamic analysis and penetration testing. In particular the focus lies on an application usage close to reality, so that extensive fuzz testing would not serve the purpose of this study. In addition the framework allows for extensibility respectively flexibility in testing and is designed to scale for larger test sets. Hence the four main functional blocks of the framework are highlighted in figure 3.1 and provide the structure for any test run:

1. *preparation* – preparing the environment and devices by adding, downloading and installing Android apps
2. *static analysis* – mainly looking at the Android APK files and related information that can be retrieved from Android market
3. *dynamic analysis* – running experiments on full solutions based on defined *test cases* and *test steps*
4. *post experiment analysis* – investigating the captured traffic, logs and application data
5. *export results* – support for reporting of the results with different levels of detail
6. *tool support* – set of tools and tool-chains that supports reverse engineering of applications (decompiling, re-compiling, source code viewer and more)

The test cases for dynamic analysis are given by attack vectors AV1 - AV5 which have been defined in *2.3.3 Attack Surface & Vectors* and test steps are either for the purpose of general information collection or refer to commonly known attacks, targeting at weaknesses as discussed in *2.3.2 Weaknesses*. The structure defined by test cases and test steps provides the grid for test execution and related data collection.
The following sections define and describe the framework. As this project is not in itself a software engineering project, requirements are only loosely defined and the implementation is described on a functional level. Nevertheless decisions to include different research tools and functionality are explained by means of their intended purpose.

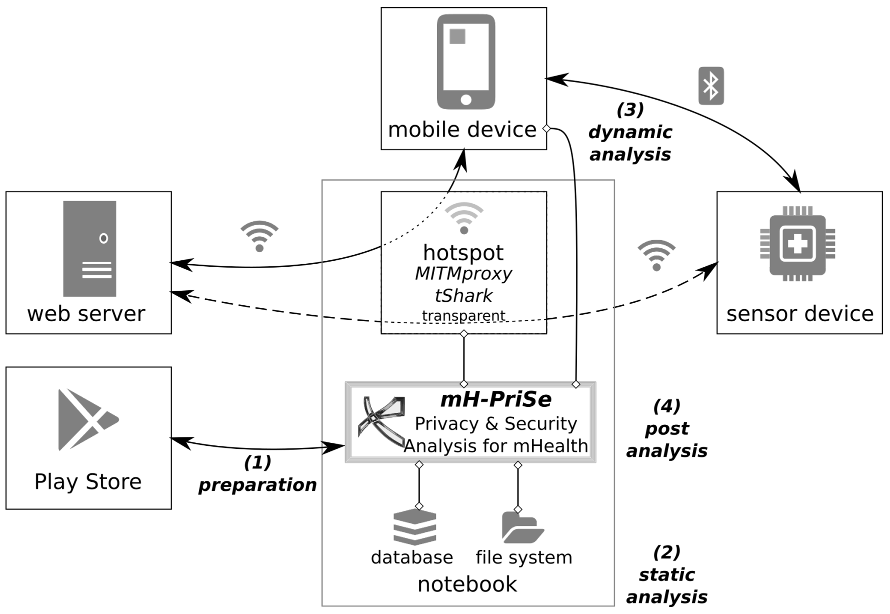


Figure 3.1: mHealth security and privacy analysis framework

3.2 Process and Requirements

The process follows the two questions of "what the solution can be assumed or is supposed to do" versus "what the solution actually does". The former can be inferred from the solutions purpose, provided documentation such as user guides and its privacy policy or terms of use, if they exist. The latter is less accessible to the "normal" user and requires specific hardware equipment, software tools and knowledge. To gain the required insights the research process following the aforementioned four functional blocks is defined as follows:

Preparation Investigating the solutions requires preparatory steps to make sure that they are supported in the test landscape. This includes the preparation of the mobile phone and setting up access points and network connections. As the analysis should focus on one solution at a time, a test start up and shut down procedure is required. This procedure should ensure that any captured data and events originate from the subject of investigation. Specifically noise created by the mobile operating system or other applications running in background is undesirable. Furthermore as solutions require specific mobile apps (companion apps) to be installed these have to be retrieved, installed on the device and kept for traceability. In addition to those companion apps, other tools as apps are required to be installed on the phone (see Appendix Tables A.11 and A.12).

Static Analysis As a next step static analysis helps to identify basic weaknesses and potential vulnerabilities. These weaknesses serve as starting points for a further, more detailed analysis. As will be discussed in the next section various research tools have been published that allow for different kinds of attack vectors to be examined. While static analysis also allows for automated investigation of larger sets of applications, results may include false positives and false negatives. Hence it is reasonable to validate the results of such analysis as part of a dynamic analysis in more detail.

Dynamic Analysis Dynamic analysis requires to observe the real behaviour of an application and in the context of this thesis the entire solution. Hence although it is wishful to automate as much of the process as possible, some manually executed testing will be necessary. The aim is to see how the solutions collect their data from the sensor device and upload it to the server. Accordingly the traffic between the mobile application and the server produces the most accessible and interesting data for this purpose and on these grounds this traffic will be captured. To make the results of dynamic analysis reproducible, test cases with test steps are defined and a documentation is created automatically and interactively during test case execution.

Post Experiment Lastly, as post analysis activity, for the purpose of analysing the solution's actual behaviour, the captured traffic and its destinations respectively sources are to be examined. As laws and regulations differ based on their country of origin the geographical location of such sources and destinations is important. Furthermore web servers communicating with the application will be investigated using some basic security scan as they have to adhere to security standards as well. But as web application and web server security is a well researched and explored topic where knowledge is commonly available on the web, a basic security scan will suffice for the purpose of this work.

Privacy Impacts The sequence of tests explained before are expected to provide a good overview on the security of a single solution. Complementary the privacy aspects of a solution have to be investigated. This may be consequences of revealed security issues but also should include privacy policies or any additional documentation provided by the vendor.

3.3 Test Setup

While the following only provides a high level overview on the set-up that was used to carry out the experiments, more details on the used hardware and software can be found in the Appendix in Table A.11 and Table A.12.

- A Lenovo Thinkpad X230 laptop with a SSD card for the required performance during experiments and extended by a 4 TB hard drive to store all captured information.
- Kali Linux 2.0 [105] as operating system coming with many tools pre-installed
- The LG Nexus 5 with Android 6.0 as mobile test device (Android 4.2.2 was the minimum requirement to cover all apps; Android 6.0 proofed to be most stable with all apps)
- internal WiFi adapter or external card such as Atheros AR9271 to create a hotspot

The laptop runs a hotspot to share its internet connection with the mobile device and makes use the Linux tools *hostapd* and *dnsmasq*. The framework allows to toggle the hotspot on and off whilst it configures the OS *iptables* to re-root the traffic through a locally running Man In The Middle (MITM) proxy. On the laptop MITMproxy [106] is installed. MITMproxy is configured to forge certificates for all incoming communication and to write the SSL master keys which have been used to (symmetrically) encrypt the communication to a log file. Then Wireshark [107] respectively its command-line client tShark reads these keys to decrypt and store the incoming traffic. Where MITMproxy is limited to the capturing of HTTP traffic, this combination has the advantage of recording traffic on lower protocol levels than the application layer which in turn helps to identify the endpoints with which communication has taken place. A MySQL database, a local Apache2 server and phpMyAdmin were used to document the experiments and store the captured results. The Python framework directly connects to that database and to store and read information. PhpMyAdmin [108] served as an additional, simplistic user interface which can be used to view and modify database records. To complement the setup Android development tools and other supporting development and build tools were installed. A full list of installed software can be found in the appendix.
In the following "gross efforts" in the absence of solution specific requirements such as different architectures (see *2.2.2 Solution Architecture*) or issues are mentioned.

The time effort from setting up the experiment environment, over designing and implementing the framework to finishing the first full investigation of a scale meant to complete the first part of the project. The total effort for this part roughly summed up to 130 hours as design decisions to include or exclude tools (see *3.4 Functional Overview*) were taken, scripts to inter link tools and write results to database and file system were added as well as test cases respectively test steps were defined. Naturally, while investigating the first solution, some parts of the framework had to be improved or were extended which meant a total effort of 24 hours for investigating the first solution. After becoming more familiar with the test execution and a few optimizations to the framework, a single solution could be analysed in approximately 5 hours of experimentation plus 2 hours of post experiment analysis.
As discussed in the next section *mH-PriSe* makes use of Python frameworks, Unix tools and 3rd party applications. A full installation of all dependencies, configuration of 3rd party tools and defining settings for *mH-PriSe* will take up to 4 hours on a physical machine for a factory installation of Ubuntu 16.04 or Kali Linux 2.0. To run the framework from within a virtual machine, the use of an external WiFi card is recommended.

3.4 Functional Overview

Where this section just provides a high level overview on functionality and purpose of the framework, Tables A.5 to A.10 show the full details on functionality and tables A.11 and A.12 provide information on tools and versions used for the research. The steps discussed here were either executed manually or wherever possible, automated. Where manual execution is inevitable the framework gives guidance and supports documentation. The major part of the framework has been implemented in Python targeting version 2.7 but also comes with native dependencies. As a simple user interface a terminal based select menu which is based on Python Curses package has been chosen. Tools within this frameworks have been chosen from a wide-range of freely available research tools [63]. Decisions in favour of a specific tool where alternatives would have been available are briefly discussed. The main purpose of the framework is to allow for efficient data extraction, collection, analysis and evaluation to meet the envisaged functionality from *3.2 Process and Requirements*.

Static Analysis Fig. 3.2 shows a variety of different static analysis tools that are available to extract information from development artefacts. For the purpose of this framework their use is limited to basic information gathering as a preparation step for dynamic analysis. Hence the aim of this step is to identify commonly known weaknesses such as communication over insecure channels and improper usage of cryptographic functionality (*Mallodroid* [70]), data storage and data leakage (*Android Debug Tool (ADB)*), exported and potentially unprotected application components (*Drozer* [109]), weak certificates for application signing (*OpenSSL* [14]) and checking for obfuscation (*Androguard* [14, 110]). Mobile Malware and ad-library usage in applications is a well researched and still evolving topic [111–113]. A variety of research tools is available, such as RiskRanker [114], recently Harvester [115] and others. Ultimately these tools investigate which data is being collected by what parts of an application and then send to remote servers. As the framework covers this part through investigating the data in transmission (dynamic analysis), there is no need to revert to one of these tools. Obviously this functionality is by far less comprehensive and less automated compared to the mentioned tools but sufficient for its purpose. As the proposed framework focusses on the real behaviour and therefore on examining data in transmission some basic insights into the application will suffice. Libraries and frameworks that are part of the application are detected (*Addon Detector* [116]). Furthermore Malware detection is used to identify potentially malicious applications (*EviCheck* [117] and *ExplainDroid* [118]).

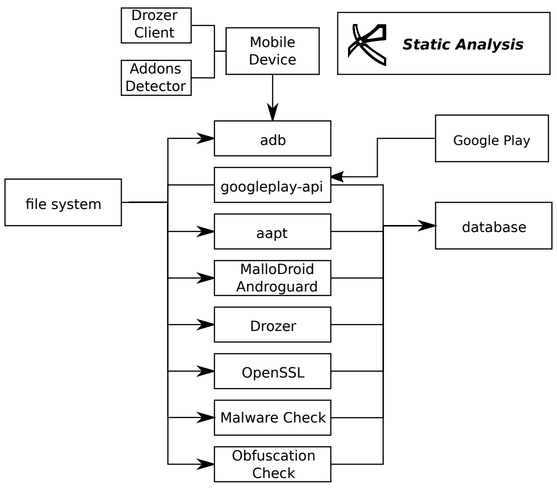


Figure 3.2: Data flow - Data Extraction

Dynamic Analysis Data is captured during dynamic analysis and generated by actual user input as the solution is used by a consumer (see Fig. 3.3). Here the same applies as for Malware and library analysis. Different dynamic anaylsis tools have been proposed and offer capabilities of large scale dynamic analysis. However none of them would suit the specific requirements as they use random GUI input and are mostly based on old Android versions lacking support by some of the tested solutions (*FlowDroid* [119]). Others seem to be out of service (*Copper-Droid* [120], *TraceDroid* [121]) or could not be fixed in a reasonable amount of time (*TaintDroid* [122]). To gather clean and reproducible data a sanitization or start up procedure has been implemented which will ensure data quality when test cases are run. A physical mobile device is connected via USB to the laptop. This allows for remote device control (*Vysor* [123]) and management through the laptop (ADB) and facilitates testing. Data is captured as a screen cast of the mirrored phone screen (*recordmydesktop* [124]), the traffic routed and intercepted through the access point (*mitmproxy* [106] and *tShark* [107]) and Bluetooth traffic on the phone (*Android HCI logging*). As part of the start up a sanity procedure is run which will only allow traffic from the app under investigation to be captured (*DroidWall* [125]) and reset Android logs (*ADB logcat and HCI log*). After finishing the experiments logging data and application sandbox data is copied from the device as well as the application stopped and its application data finally erased (*ADB with shell, toybox*). These start-up and shut down procedures are part of every experiment run and allow for reproducible and clean results.

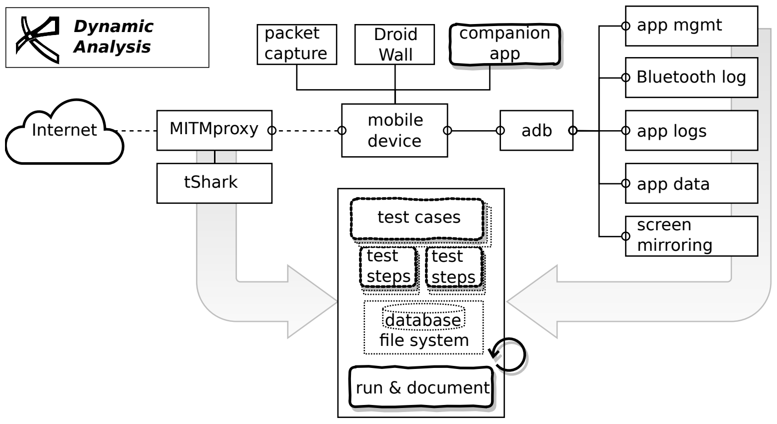


Figure 3.3: Data flow for step Data Capture (highlighted boxes inidicate test parameters)

Post Experiment Following the experiments the captured data can be analysed towards the defined security and privacy aspect (see Fig. 3.4). Data analysis is mostly a manual task (logs, application sandbox data, traffic inspection). Nevertheless the framework offers some basic support for this activity in terms of creating traffic maps based on the captured data (*mitmproxy scripting* [106], *ip database*, *matplotlib* [126]). As MITMproxy logs include HTTP traffic only, the resulting list of communication destinations is validated against those IP addresses extracted from tShark log files. The result will represent a full list of HTTP and TCP based requests fired by the application. As with the number of recorded test cases and test steps the database is populated, the framework helps to navigate through recorded experiments.

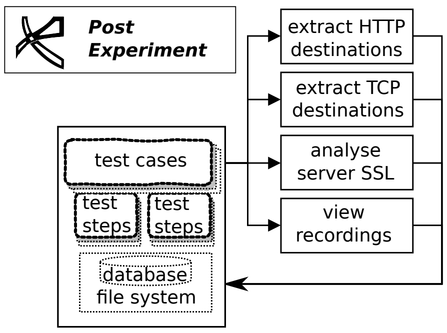


Figure 3.4: Data flow for step Data Analysis (highlighted boxes inidicate test parameters)

Privacy Impacts The main source of information are privacy policies. As many privacy policies exhibit any common structure and their content is often even for humans difficult to understand, research has not yet found a solution to automatically process privacy policies [127]. Therefore manually evaluation with respect to the aforementioned OECD guidelines is inevitable. In order to compare the different solutions based on the gathered data it is not feasible to define scores for specific test steps, due to the limited scope and time of this project. Hence this report will be limited to the pure number of issues that have been detected. This total number can still be used to draw a comparison between different solutions without discussing the specific steps and criteria as well as assigned scores in much detail.

Evaluation To support evaluation data can be exported from the database in different forms including an overview on results of test cases in HTML or PDF with issue counts and comments. It also allows for the generation of csv files and templates for the LaTeX datatool package. These exports include all results of static and dynamic analysis that have previously been written to the database.

3.5 Test Cases and Test Steps

The appendix Tables A.25 to A.29 show detailed description of all test cases and steps including assigned criteria to succeed in a test step. For the purpose of this project test cases have been defined following the Threat Model defined in *2.3 Mobile Health Threat Model*. Each of the test cases has several test steps that cover weaknesses which could be exploited through commonly known attacks.

(AV 1) Data Transmission Data in transmission is such data being transferred from the sensor device or mobile device to another remote entity. This test case aims to exploit bad or no SSL usage through replay attacks, code respectively data injection or traffic tampering.

(AV 2) Mobile Device Attacks that require physical access to the mobile device via a wired connection and are more related to Android OS functionality rather than the mobile application's functionality are summarized under this case. An attacker would be able to read any visible data from within the application sandbox as part of a backup or any information logged to the Android system logging service Logcat.

(AV 3) Mobile App In total 15 different test steps are summarized within that test case. This case includes the detailed investigation of the mobile application embracing the registration process, information collection with the sensor device and mobile application, data export, data wipe feature and correct password handling. These test steps aim to reveal weaknesses in data storage and export (encryption), password management (policy and storage), unintended or not permitted data leakage (ad libraries), input validation (sanitization and content-related) and identity verification (account verification or second factor authentication). For the privacy it is checked if the user is pointed towards a privacy policy or any other information on how their privacy is respected.

(AV 4) Sensor Device The sensor device is not the main focus and hence only four steps are summarized here. All three of them are related to the application behaviour and their results can be inferred by observing the application and its traffic. For further investigation into the sensor device more steps would have to be thought of and added here. Depending on how the pairing process is executed from within the mobile application one can learn some information about the way the Bluetooth protocol is respected. Furthermore the sensor device might be able to receive firmware updates which would be downloaded by the corresponding mobile application. Last characteristics of the data being collected are noted. If the sensor device connects immediately to the internet this traffic can be captured. In other cases if the application acts as a proxy the applications traffic might reveal information.

(AV 5) Web Application Although web applications are a widely explored topic and there are several tools to test their security readily available, the framework includes 17 steps to test parts which are more specific to the scenario these include e.g. communication between the web server and the application as well as specific paring processes for the sensor device and a corresponding user account. Again these steps embrace registration processes, information collection via web interface, data export, data wipe or the right to be forgotten [128], password management, data leakage, input validation and identity verification. In addition commonly known attacks on web pages are included such as Cross Site Scripting (XSS), Cross Site Resource Forgery (CSRF) and SQL injection. As for the privacy policy it is checked whether a policy is visibly linked and accessible to the user.

solution architecture (comprising all AV) As different solutions have similar but sometimes slightly different architectures a description of communication flows and implementation specifics is recorded. This might include comments on the mobile application implementation, on whether it is a pure Java for Android implementation or if other frameworks are used (Xamarin with Monodroid, Apache Cordova and others) and on how sensor devices communicate with the remote web server.

3.6 Testing Schedule

This section will briefly summarize a typical testing schedule for a solution. For this purpose the following example is assumed: The solution includes a smart scale which connects over Bluetooth to a mobile device on which a companion application is running. This application is of high security standards and employs certificate pinning to protect its data stream to the vendor's web server. From a high level perspective *mH-PriSe* is used as follows to analyse the solution:

Step 1 The test environment is being set up, the phone connected to the laptop via USB cable, the hotspot started and the WiFi connection on the phone is established. The *mH-PriSe* is installed on the laptop, configured and ready to run.

Step 2 The application identifier (package ID) is specified and an APK file for this package ID is retrieved from Google Playstore before it is installed to the phone.

Step 3 Through static analysis applied to the APK file basic first insights into the solution are gained. Is the mobile application likely to be malicious (permissions, Malware classification, addons and libraries) and does it adhere to secure coding standards (SSL usage, correct certificate validation)?

Step 4 Iterating through the test steps of the first test case *(AV 1) Data Transmission* the certificate pinning becomes obvious and has to be removed for further insights.

Step 5 Functionality from *mH-PriSe* tools menu is used to decompile, manually remove the pinning and recompile the application. It is automatically package, aligned and signed to be ready for installation to the phone.

Step 6 The remaining test cases and test steps are executed which will provide detailed insights into the application's security.

Step 7 Post analysis functionality of *mH-PriSe* is used to generate a traffic map and manually investigate the recorded traffic and log data. Additional insights and comments are added to the previously recorded test cases. To assess web server security SSLlab.com scans are run on all IP addresses that have been extracted from the captured traffic.

Step 8 All data has been extracted, captured and collected, and has all been stored in the database. Hence the *mH-PriSe* export functions are used to generate CSV files, LaTeXtemplates and HTML exports which will help to further interpret the data. These exports include detailed comments on each test steps as well as accumulated issue counts per test case.

Step 9 Manual interpretation and the comparison of privacy impacts that have been derived from the security analysis, serve as the basis to assess the privacy properties of the solution.

Chapter 4

Scales put on the Scales

4.1 Solutions under Investigation

4.1.1 Overview

After the framework was introduced earlier this chapter focusses on cases studies with mHealth devices, here smart scales from different vendors. Outcomes will help to compare these different solutions with respect their privacy and security properties. In the Appendix Tables A.25 to A.29 provide detailed information on all test cases and test steps that have been executed. This chapter provides an overview of the dataset, meaning all scales in scope of this work, it summarizes flaws and issues that have been observed commonly among all solutions and it lastly provides a more detailed discussion of issues for each solution.

4.1.2 Smart Scales and Mobile Applications

The framework is used to investigate the security and privacy of 8 different bathroom scales from 6 different vendors (see Appendix Table A.1). These scales range from older models starting in 2012 with the Withings WS-30 or Fitbit Aria to newer models such as Withings Body Cardio. As such their price and functionality varies. The selected products range from the lower end scales, which only support measuring weight, to some that allow for body fat measurement and Body-Mass-Index calculation, all the way to the high end functionality of the Withings Body Cardio, which allows users to measure their body water percentage, muscle mass and bone mass. Hence the selection of scales includes major mHealth players such as Withings and FitBit as well as less known start-ups like HAPI, Activ8rlives, Thomson and iChoice. All vendors provide a mobile application as a companion to their scale. Commonly a vendor will supply one application that offers support for many mHealth devices from different categories. The Withings mobile application can consequently be used with all their scales. In contrast, the iCoice S1 scale is supported by two different applications written by an iChoice contractor. Therefore both applications are in scope for this analysis. The full list of mobile applications and which smart scales they are supporting is presented in Tables A.1 and A.3.

4.1.3 Smart Scale Data Flow

Figure 4.1 depicts connections and data flows for all scales in scope. The majority of scales supports a Bluetooth or Bluetooth Low Energy connection to transfer data between scale and mobile device. In addition Withings scales support WiFi enabled communication and Fitbit Aria only supports WiFi communication. Whatever connection is to be used, it has to be set up in the first place. The pairing procedure accordingly depends on the sensor's connection type and is important for further security analysis. For Bluetooth enabled devices this mostly includes the (strict) pairing between smart scale and mobile phone based on the Bluetooth protocol specification. This might involve the usage of a pass code or other security features. However this step is not strictly required as communication with devices is sometimes possible even without pairing. On the other hand if the device is WiFi enabled, the pairing includes the process of providing WiFi credentials to the devices. Subsequently the device will use these credentials to connect.

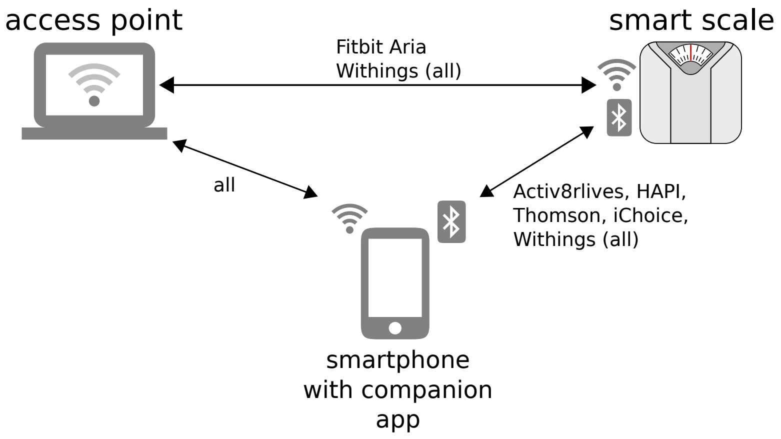


Figure 4.1: Bathroom scales connection overview

4.2 General Findings

4.2.1 Unsuccessful Approaches

As opposed to the general focus of this work on privacy and security issues in mHealth solutions the following paragraph/section will briefly summarize other approaches that have been taken and methods that have been used without successfully revealing any issues. Unsuccessful tests are still important to mention as they indicate good practices.

Android developer certificates are used to sign applications and thus play a major role when updating applications. Only applications signed with the same certificate can be replaced respectively updated automatically, that is without the user's consent. Hence weak developer certificates with publicly available private keys or too short keys, present weaknesses to the application update procedure. None of the applications in scope was signed with such a bad certificate (see Appendix Table A.16).

Port scanning and DDoS attacks for devices using Wifi, namely Withings sales and Fitbit Aria have been executed. These attacks however remained unsuccessful.

4.2.2 Getting SSL Right

	activ8rlives	**FitbitMobile**	**hapiconnect**	**medm**	**ichoice**	**thomson**	**withings**
app to server	✗ trust manager issue	⚠ no pinning	✗ no SSL	✓ certificate pinning	✗ trust manager issue	✗ no SSL	⚠ no pinning
CWE-295: Improper Certificate Validation							

Table 4.1: SSL issues in mobile applications

Since Georgiev et al. first reviewed issues with SSL/TLS implementations in various software and underlying libraries [69], this area has been researched intensively. Fahl et al. provided evidence that the very same issues exist with mobile applications [70]. More recent studies, including the report at hand, confirm that SSL implementations are still broken in many applications [71, 72, 129]. Although researchers have proposed different possible solutions from API redesign [130] to dynamic inspection [131, 132] implementing SSL correctly still seems to be a challenge. Possible causes are the need to use self-signed certificates, an enterprise owned CA or simply failing to correctly implement certificate pinning. In Android the classes `SSLSocketFactory`, `HostnameVerfier` and `TrustManager` have to be overwritten for this purpose. Accidentally applications end up (1) trusting all certificates, (2) allowing all host names or (3) trusting many certificate authorities. Often the developer's intention in such cases was to trust a specific certificate, to allow a specific host name only or to add the corporate certificate authority as a trusted entity.

In a working solution a provided server certificate is validated through its trust chain driven by the trust manager implementation. The standard implementation runs this validation through any trust relationships it is able to build based on certificates from the operating system certificate store. In customized implementations an overwritten `TrustManager` class may point to an application specific store (for certificate pinning [133]) or to a single certificate. In addition the hostname of a target server being contacted is validated against the Common Name (CN) taken from the certificate it is providing (`HostnameVerifier`). Failure in implementing this check correctly will lead to trusting all server hostnames.

The consequences are alarming. The purpose of X.509 public key infrastructure and its trust model is defeated entirely. There is no assurance the application is communicating with a specific server. In fact broken SSL implementations allow for MITMA. In a MITMA an attacker manages to eavesdrop on the traffic between two benign entities. The paper by Song et al. showcases an successful attempt of researchers creating a malicious hotspot at Starbucks and eavesdropping on any communication [134]. As mentioned in *2.2.4 Eavesdropping on Communication* MITMA are important for the discoveries of this project.

Out of the 7 applications in scope only 3 seem to have implemented SSL correctly. The Fitbit and Withings application follow Android guidelines and use certificates signed by public CA. Hence their trust chain can be validated correctly. But, if an attacker manages to install her CA certificate to an attackee's certificate store she still is able to intercept traffic. Installation of such a certificate is e.g. possible through social engineering – see Starbucks paper [134]. Recent Android versions however notify the user about suspicious certificates in their store. This advanced type of MITMA which requires the CA certificate to be installed on the phone, can only be avoided through certificate pinning. There is only one application out of the 7 in scope that implements certificate pinning correctly: com.medm.medmwt.diary (medm). Interestingly com.medm.ichoice.diary (ichoice) is developed by the same company, apparently contracted by iChoice, a Switzerland based company called "Swissmed Mobile". However this application which was released last year fails to validate certificates correctly.

4.2.3 Playing with Traffic

	activ8rlives	**FitbitMobile**	**hapiconnect**	**medm**	**ichoice**	**thomson**	**withings**
replay attacks	✗ one record per day limit	✗ possible	✓ UUID defeats attack	✓ UUID defeats attack	✓ UUID defeats attack	✗ possible	✗ possible within session
tampering	✗ no content validation	✗ possible – MAC defeated	✗ no content validation	⚠ possible – upper weight limit	⚠ possible – upper weight limit	✗ possible	✗ possible within session
CWE-311: Missing Encryption of Sensitive Data							

Table 4.2: Tampering with data in transmission

One consequence of SSL implementation issues described earlier is the exposure to potential injection, tampering and replay attacks. Missing encryption facilitates these attacks by allowing unrestricted access to the data stream. Regardless of this, for this sub section privileged access to the data stream is assumed. Briefly a replay-attack is described as capturing and replaying traffic without any modification. As such the attack would be possible even if the traffic is SSL encrypted [76, ch. 6.2]. Opposed to this attack a tampering includes the fabrication or manipulation of data. To protect against replay attacks the use of one time passwords or in a less complex approach employing UUID for data records suffices. Tampering attacks are harder to protect from. A one-time password for each data record used to encrypt the same would be one possibility. Another way would be the use of Message Authentication Codes (MAC) which are used to verify the integrity of a message. If a sender provides an MAC with his message, a receiver is able to verify the integrity based on recomputing the MAC. Often MAC are used in combination with public key cryptography to prevent attackers from tampering with data and recomputing the MAC. For this purpose the sender will encrypt the MAC using a private key. The receiver retrieves the senders public key to decrypt the MAC and then proceeds as described before [135, p. 215, pp. 271 ff.].
Only three applications protected against replay attacks by making use of client side generated UUID which also facilitate data synchronisation by uniquely identifying single data records. However non of the applications in scope protects successfully against an attacker tampering with data. The Fitbit Aria scale uses a MAC while communicating with the Fitbit server. However messages can still be fabricated by an attacker with the required knowledge (see *4.3.2 Fitbit Aria*). On top of that only two

web servers protect against unreasonably high weights by validating the message content. Other servers allow for unrealistic measurements. As corresponding applications often apply upper limits for inputs this issue is similar to *CWE-20: Improper Input Validation*. However, none of the examined solutions exhibits vulnerabilities related to improper input validation or missing sanitization in the sense of the CWE description.

4.2.4 The Power of Passwords

	activ8rlives	**FitbitMobile**	**hapiconnect**	**medm**	**ichoice**	**thomson**	**withings**
password policy	✗ no policy	⚠ min. 6 characters					
password change	⚠ no enforcement						
CWE-521: Weak Password Requirements							

Table 4.3: Issues with passwords

Passwords, password policies to support strong passwords and password management are other well research topics [136–138]. Accordingly recommendations and guidelines to enforce good password policies and password change management are freely available [139]. As good practice password policies should enforce high entropy of passwords [139], that is passwords should meet 3 out of 4 following complexity rules: one or more upper case letters, one or more lower case letters, one or more digits and one or more special characters. Passwords should also have a minimum length of 10 characters. Often such passwords are very hard to memorize which is the reason to weaken such requirements. Furthermore passwords should be changed within some period and password reuse should be prohibited. In case the password database would be compromised such steps limit the information gain for the attacker.
All applications in scope failed to provide adequate password policies. All but one application (Activ8rlives) enforced a minimum password length. None would enforce the change of the password at any time. Only HAPI application disallows to reuse an old password when setting a new one. In general none of the applications required to re-enter the password any time after the first login.

4.2.5 Managing Data

	activ8rlives	FitbitMobile	hapiconnect	medm	ichoice	thomson	withings
data wipe	⚠ feature not implemented						
account deletion	⚠ via web page or email only						
–							

Table 4.4: Data management issues

The application developer has several choices on Android to store data. In general data is either stored inside the application sandbox or on external storage. Whereas the application sandbox is exclusively accessible to an application (with the exception of root user), the external storage is used by all applications. Hence the preferred choice for better security is to store data inside the application sandbox. All applications in scope follow this principle and store their data either in form of a sqlite database or cache files. Hence the data in that position is as secure as the application sandbox. In turn the sandbox is as secure as the application has been developed [62, p.13 ff.]. Although any data stored inside that sandbox will be erased when the application is removed from the device, data in external storage will remain. This data requires to be deleted manually or through a data wipe feature. With reference to the OECD Individual Participation Principle an user should be enabled to delete their account with a service vendor entirely, including all data. All solutions in scope require users to access their website where they might find a delete or deactivate button on their profile page. In other cases account deletion is only possible via email, but in no case account deletion would have been possible through the mobile app.

4.2.6 Android Permissions

	activ8rlives	FitbitMobile	hapiconnect	medm	ichoice	thomson	withings
level of privilege	✗ over privileged – record_audio, bluetooth_privileged, fine_location	⚠ privileged – get_accounts, read_contacts, fine_location, ...	✗ over privileged – bluetooth_privileged, download_without_not camera	✓ reasonably privileged	✓ reasonably privileged	✓ reasonably privileged	⚠ privileged – get_accounts, fine_location, ...
CWE-250: Execution with Unnecessary Privileges							

Table 4.5: Privileged apps and their permissions

Android permissions are an evolving topic and an active area of research [117, 140, 141]. The permission model has undergone many changes adding new permissions and removing old ones in reaction to various issues. Applications are called *overprivileged* if they request more privileges than required to fulfil their intended purpose. There are various reasons for applications to be over privileged. In this context a malicious intent is assumed. An application would first behave as expected but ask for a large set of permissions when it is installed. In subsequent updates code is added to the application that uses these unused privileges to execute malicious functionality. Although applications no longer ask for all permission at once when being installed [142], overprivileged applications might still cause harm if the user decides to allow all permission being asked for and checks the never ask again box.

A reasonable set of permissions for a weight measurement application might include: Bluetooth, internet, Bluetooth_admin (to discover and pair with devices), access external storage for read and write. Depending on their further functionality other permissions might be added. Out of the applications in scope only the iChoice and Swissmed Mobile (medm) applications follow this by only requiring a minimum set of permissions. Worrying are applications Activ8rlives and HAPI as they are highly overprivileged. For example there is no reason why Activ8rlives would need to access the microphone and record audio or use privileged Bluetooth which allows to access contacts without any notice. Similar application HAPI which requests the same Bluetooth permission but also to download content in background without user notification. This permission set should be urgently updated. Applications Fitbit and Withings support functionality beyond simply synchronising weight data and hence request more permissions. Whereas all of their permissions are reasonable for their purposes, a user should be aware of the application having access to data beyond weight measurements including the phones address book which is used to find friends using the same app.

Conspicuously all applications but one require some kind of location access, coarse or fine. Applications designed for Android 6.0 or higher require this location permission in order to start scanning for Bluetooth devices. The idea behind this design is based on location approximation using beacons, which would require to run a Bluetooth device discovery scan. Unfortunately this design is very confusing from a developer and user point of view and hence is discussed controversially in the community.

4.3 Studies of Smart Scales

4.3.1 Activ8rlives Body Analyser

sensor	Activ8rLives Body Analyser
release date	Apr, 2014
price (GBP)	59.99
wifi	✗
Bluetooth	✓

package	com.activ8rlives.mobile
release date	May, 2016
company	Aseptika Ltd
installs (k)	10
version	3.16.4

Activ8rlives Body Analyser is accompanied by an Android application based on the Xamarin framework [143]. Xamarin is a cross-platform development framework based on C#. The virtual machine which runs Xamarin applications is called MonoDroid and is packaged with any Xamarin application. Thus it has recently been updated, it was difficult to run the application as it crashed regularly on Android 6.0. Two major technical flaws were found for this solution including no password policy and a broken SSL implementation as mentioned before. In addition the following major issue were detected.
Complementary to the client side SSL issues, the web server shows several concerning misconfiguration allowing known attacks on SSL. These attacks include the *DRAWN* (CVE-2016-0800, CVE-2016-0703), *POODLE* (CVE-2014-3566) and the recent *Padding Oracle* found in OpenSSL CBC Ciphersuites (CVE-2016-2107). These vulnerabilities would allow to steal cookies and sessions even if the mobile application was properly using SSL. For this purposes the *POODLE* attack would suffice if the attacker manages to control the network [144].
According to its privacy policy account deactivation or deletion is available up on request per email. For testing purposes two different accounts had been created. On request both of them were happily deleted even though the email was send from another account. Meanwhile the Xamarin framework is the basis for this application it also collects extensive information of its host and sends it to their servers. The device fingerprinting techniques used in this framework have to be updated to newest Android programming guidelines. Reading the MAC address of a device is no longer recommended.

4.3.2 Fitbit Aria

sensor	Fitbit Aria
release date	Apr, 2012
price (GBP)	99.99
wifi	✓
Bluetooth	✗

package	com.fitbit.FitbitMobile
release date	Jul, 2016
company	Fitbit Inc.
installs (k)	10
version	2.29

Whereas the Fitbit mobile application successfully protects against most security issues found in other applications, the weak point in this case is the scale itself. The scale connects via WiFi directly to the internet and sends newly recorded data. Due to the popularity of Fitbit activity trackers the scale has been investigated by various security researchers. Starting with the research of Mewes who started to reverse engineer the communication protocol between scale and server [145] further research targeted updated firmware versions [146–148]. This arms race so far has led to firmware version v39.

Fitbit decided to run their user registration and device pairing processes solely through an web browser online session. Hence their mobile application is just another view on their web page but lacks user registration and pairing functionality. Lodge showed in his work how vulnerable the pairing process is. His findings are confirmed based on experiments with the latest firmware version. The user retrieves a registration token from Fitbit web server. Using this token, the device serial number (MAC address) and WiFi SSID the scale is associated with an account. In fact the association is based on the token. Lodge also states that the entropy of this token is very low and hence it may be brute-forced [147]. Earlier Farrell had successfully reverse engineered the scales protocol to upload data [146]. Their findings can be confirmed with the newest firmware. Although the protocol applies MAC traffic injection and tampering it is still possible. The MAC is simply calculated as a `CRC-16-CCITT` [149, ch. 14] on the whole message. In short the protocol is still broken.

Meanwhile another issue where the scale would show the wifi password as part of its error page has been fixed [148]. Still the pairing process requires to connect via wifi to the scale and then provide network SSID and password. As the wifi hotspot the scale exposes is not protected, credentials are transferred visible, in clear text to the scale.

4.3.3 HAPI Connected Scale

sensor	HAPI Connected Scale
release date	May, 2015
price (GBP)	49.99
wifi	✗
Bluetooth	✓

package	com.hapiconnect
release date	Jul, 2016
company	HAPILABS
installs (k)	100
version	1.2.8_US_Server

The HAPI Connected Scale communicates via Bluetooth only. Its companion application lacks SSL encryption and although no malicious behaviour could be found, it is highly over privileged. Without traffic encryption an attacker can learn PII through eavesdropping on the connection including user name, password, date of birth, weight and height.

It is recommended to store hashed passwords only, and while computing the hash to append a randomly generated string (salt) to the password. Hash functions are required to be collision resistant, that is no two different messages should produce the same hash value. Whereas this property is required to maintain a prohibitive computational complexity which would prevent an attacker from brute forcing the hash, it does not protect against lookup attacks. For these attacks large database of computed hash values mapped to the messages they originated from are maintained. An attacker will simply look up the captured hash and retrieve the password. Large databases are freely available on the internet [150]. Adding a salt to the password before hashing it defeats the look up attack. The HAPI application suffers from that vulnerability as passwords are hashed before they are transmitted to the server but no salt is used. Hence if the idea was to protect passwords being send over an unencrypted connection by hashing, the implementation needs to be reconsidered.

Another main issue identified with this application leads to Android logcat. An application is able to log information to the Android system log. Whereas this feature is very helpful during application development it should be disabled when publishing the app. The HAPI Connected app logs extensive information to logcat. An attacker with access to this log file can learn about the database structure and content based on SQL queries which are being logged.

4.3.4 iChoice S1

sensor	iChoice S1
release date	Aug, 2015
price (GBP)	44.69
wifi	✗
Bluetooth	✓

pacakge	com.medm.medmwt.diary	com.medm.ichoice.diary
release date	Apr, 2016	Mar, 2015
company	MedM Inc	ChoiceMMed America
installs (k)	500	1
version	2.0.26	1.7.8

The iChoice S1 scale is the only scale in scope that is supported by two different mobile applications. Apparently iChoice decided to use software developed by SwissMed for their mobile application and web application. At the same time SwissMed use the same software to offer their own service. As a consequence the web applications and mobile applications are of similar functionality but run with different web servers. Both mobile applications are hybrid applications based on Android Native Development Kit (NDK) and Android SDK components. Surprisingly the com.medm.medmwt.diary (MedM) performed better than the com.medm.ichoice.diary (iChoice) application. Being signed using different developer certificates (see Appendix Table A.16) and with different release dates and version numbers the MedM application has apparently been improved on security.

The iChoice application has a major flaw trusting all certificates when establishing SSL connections. As opposed to that the MedM application employs certificate pinning which makes it impossible to forge a certificate and establish MITMA. Hence in order to run this attack the application had to be patched. Luckily the certifcate serial number turned out to be not hardcoded, but a separate certificate store packaged with the application. Hence adding the MITMproxy CA certificate to that certificate store sufficed. After unpacking the application using *apktool* [151] the certificate was added to the store. Using the same tool to repackage the application, signing it with a developer key and aligning it for the Android file system was required to regain a valid APK file. A good starting point explaining required steps and tools provides the paper by Sierra et al. [152]. With this patch applied to the application the traffic of both applications looked very similar. For these applications it can be stated that every request being send or received serves the immediate purpose of providing their service.

The web offering provided as part of the solution is remakably well structured and allows for fine grained role and permission configurations. It becomes apparent that this vendors approach the mHealth market less from a fitness and well-being perspective than to support medical services through data collection. However the MedM server requires an update to protect against the *Padding Oracle vulnerability* in SSL (CVE-2016-2107).

4.3.5 Thomson TBS705

sensor	Thomson TBS705
release date	Aug, 2014
price (GBP)	59.99
wifi	✗
Bluetooth	✓

package	com.stabxtom.thomson
release date	Oct, 2015
company	Thomson Healthcare
installs (k)	1
version	v1.1-B018

The Thomson TBS705 is another scale with Bluetooth support only. The Thomson Healthcare application allows to synchronize data from the scale and send it to the server. Meanwhile the data is indeed being send to a server there is no web page to view and manage that data nor is there any privacy policy available explaining how Thomson would deal with collected data. After deleting and reinstalling the application there is no possibility to restore previous data.
SSL traffic encryption is missing as well and all registration as well as measurement data is being leaked. Passwords are similar to HAPI Connected Scale MD5 hashed but not sufficiently salted. Either way passwords being send on an unencrypted channel hashed or not hashed are of value to an attacker. Hashed passwords still allow to authenticate with the server and not hashed or badly hashed passwords can be retrieved and tried on other web offerings with the same email account as username.

Umeng is an advertising library by a major Chinese advertising company. It is commonly classified as adware with high risk sending device information including IMEI and potentially location information to their servers [153]. The Thomson application makes use of this library and hence will send detailed information on your phone a Chinese server (see Fig. 4.2).

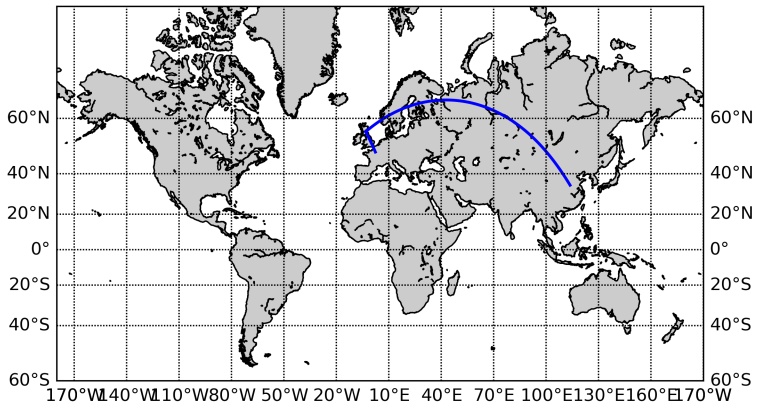


Figure 4.2: Thomson traffic map

4.3.6 Withings WS-30 and WS-50

sensor	Withings WS-30	Withings WS-50
release date	Dec, 2012	Mar, 2013
price (GBP)	79.99	129.99
wifi	✓	✓
Bluetooth	✓	✓

package	com.withings.wiscale2
release date	Jun, 2016
company	Withings
installs (k)	500
version	2.1.6

Withings were the first to offer a smart scale in 2009. Their models usually support Bluetooth and Wifi communication. If the scale is configured to use Wifi communication directly with their server similar issues as with Fitbit Aria become apparent. Communication is unencrypted and can be spoofed on by an attacker. After the first scale was released two reports highlighted this issue and reverse engineered the scales communication protocol. This work aimed to replace the Withings server to achieve full control of collected data [154, 155]. Research by Coppola targeted on replacing the devices firmware to manipulate its display [11].

The Withings user registration and pairing process is different from Fitbit Aria. User registration is possible through the mobile application as well as the web page. The device association is more complex and could not be reconstructed entirely. The missing piece is to analyse the Bluetooth communication between scale and mobile phone. For this part only assumptions can be made. Hence the following is to discuss briefly why the Withings protocol is more secure then Fitbit.

1. Logging in from the app for the first time two users with assigned public keys are provided by the server. The own user and a *repository user*. Hence the server

may hold the private keys. In addition the app receives a session ID with the server.

2. Next the app retrieves an once from the server and uses this once to construct a session request one behalf of the scale. As this session request includes information on the scale (firmware version, mac adress and more) the scale must have communicated this information. With that request the app provides a computed hash which seems to be used for authentication.
3. The scale requests an once and constructs its own session request. Using the session it sends a *deviceassociationrequest* to the server. Now the server seems to know that the scale identified by its MAC address is ready to be paired.
4. An association request with the two session ID retrieved by the app earlier is send to the server. The server associates the devices and responds with a computed secret and association ID.

Key to completely reverse this protocol is to understand how the hash values used for authentication are computed. This information can be retrieved partly from the application coding. The *smali* code[1] reveals a string that is composed of `<mac>:<secret>:<once>` and then is hashed. As the secret is only retrieved after the association of user account and scale the first hash values must be computed differently. However the idea of a shared secret and to use Bluetooth communication between scale and mobile phone to share details that can be used to securely establish a session, provides more security than Fitbit.

The traffic between mobile application and server reveals another detail about the data Withings is collecting. The location of a scale is also stored on their servers. This information is used to display a daily weather forecast to the user. Also a link to firmware updates can be extracted from communication. Through traffic manipulation firmware update attacks become possible [11].

4.3.7 Withings Body Cardio

sensor	Withings Body Cardio
release date	Jun, 2016
price (GBP)	139.95
wifi	✓
Bluetooth	✓

package	com.withings.wiscale2
release date	Jun, 2016
company	Withings
installs (k)	500
version	2.1.6

[1]see class `Lcom/withings/device/ws/DeviceSessionFactory`

The Withings Body Cardio is the newest scale in scope. Based on the same architecture and using the same companion application as the Withings WS-30 and WS-50 it sends data directly to the Withings server. The new scale is able to use SSL encryption. As a first step the scale requests the server certificate via HTTP connection. After receiving the certificate it will change to SSL encrypted traffic. This form of certificate pinning looks suspiciously vulnerable against certificate forgery attacks. In fact the scale sends its MAC address, a challenge (16 character) and a method, an hash algorithm, as part of the request. The response includes JSON string with following information. The current date (unix timestamp), a set of keys each including key size (key length), modulus (RSA public key) and exponent. For each of these keys a digest is computed which the scale will use to verify message integrity. In order to forge a certificate and run successfully MITMA one would need to provide a valid certificate of the MITMproxy and also a digest that can be verified successfully by the scale. So far the digest could not be reverse engineered. Luckily the Withings server is happy to serve as an oracle. Sending fabricated requests to the server the scale's MAC address, the challenge and the UNIX timestamp send back by the server could be identified as parts of the digests. However this information was not sufficient to recompute the digest and hence no certificate could be forged.

Chapter 5

The Real Weight of Smart Scales

5.1 Summary of Findings

5.1.1 Security Issues

Results form security analysis in Chapter 4 are aggregated in Table 5.2. To provide a classification scheme three different classes where chosen. Three different levels of qualification include: *pass* with minor issues detected (green), *warning* with some issues found that should be taken into account (yellow) and *fail* with highly severe issues detected that require action (red). This action might include a fix delivered by the vendor or a mitigation action by a user as will be discussed later. Full details on findings listed by test case and test step are available from the Appendix.
To complement this view the last row also indicates the total number of issues found and the number of possible issues for the specific solution. Per test step one or two issues, depending on the counting scheme, can be identified. As each solution comes with slightly different functionality and different use cases, the total number of issues can not be consulted to base a comparative analysis on. As an example the Thomson solutions misses a web page and consequently does not have any of the issues assigned to that part. Another example are different registration processes between solutions such as Fitbit and Withings.

5.1.2 Privacy Impacts

In Table 5.4 results of manual privacy policy inspection show that almost all comply with OECD privacy principles as described in Subsection 2.4.2. This applies to all solutions but Thomson for which no privacy policy could be found. To achieve a pass

	Activ8rLives	Fitbit Aria	HAPI Connected Scale	iChoice iChoice app	iChoice Medm app	Thomson TBS705	Withings WS-30 & WS-50	Withings Body Cardio
app to server	✗ trust issues; traffic tampering	✓ standard SSL	✗ no SSL; poor crypto usage	✗ trust issues;	✓ strong SSL	✗ no SSL; poor crypto usage	✓ standard SSL	
sensor to server	N/A	✗ no SSL; protocol reversed	N/A	N/A		N/A	✗ no SSL	✓ SSL
app & mobile device	✗ data leakage; no password policy; overprivileged	⚠ modern analytics library	✗ no encryption; broken crypto for passwords; no data wipe; *highly* overpriviledged	⚠ pwd chang policy and data wipe		✗ overpriviledged;data leakage; logging leakage; leaks credentials; no data wipe	⚠ modern analytics; no password change policy; reasonably priviledged	
sensor security	⚠ no strict BT pairing; no firmware update process	✗ leaks wifi credentials; unencrypted traffic; protocol reversed	⚠ no strict BT pairing; no firmware update process	⚠ no strict BT pairing; no firmware update process		⚠ no strict BT pairing; no firmware update process	✗ leaks session; no SSL	✓ safe pairing with SSL
web server	✗ broken account deletion process; weak password policy; vulnerable SSL config	✓ no password change policy; fine grained privacy settings	⚠ input not validated; no wipe option; password (change) polciy can be improved	✓ weak pwd policy; good privacy	⚠ same but SSL dated;	undetermined *not available*	✓ no password change policy; password policy can be improved	
n# issues (n# possible)	28 (42)	17 (37)	32 (46)	21 (40)	18 (40)	24 (27)	24 (41)	22 (41)

Table 5.2: Security analysis results (qualified)

for any of these principles required information has to be available from their policy. For example many fail to specify an accountable body which would include a company name, address and contact information. "P" as partly indicates that only an email address is specified. As start-ups are often subject to acquisitions and many mHealth vendors belong to this category, privacy policies may also contain statements on where data is stored (own server or 3rd party), what happens to data if the company is being bought and where the company is based. These three attributes hence have been added to the table.

Privacy policies by Swiss Med (iChoice S1) and Withings are of commendable standard. Withings provides detailed information on all principles that have been looked at. They state their "servers are located within the European Union (France) and are therefore subject to regulations that guarantee you a high level of protection for your personal data" [156]. Swiss Med are very precise on the information they collect and how information is shared with custodians defined by the user through their platform: "You can specify how long they have access (custodian access does not expire but, like all sharing access, it can be revoked at any time) and whether they can modify the information in the record." [157].

package	data usage	purpose specification	3rd party forward	security safeguarding	openess	individual participation	accountability	3rd party storage	merger	country
com.activ8rlives.mobile	✓	P	✓– A	✓	✓	✓	✗	✓	✓	UK
com.fitbit.FitbitMobile	✓	✓	✓– A	✓	✓	✓	P	✗	✓	US
com.hapiconnect	P	P	✓– A	✓	✓	✓	✓	✗	✓	US
com.medm.medmwt.diary	✓	✓	✓– A	✓	✓	✓	✗	✗	✗	?
com.medm.ichoice.diary	✓	✓	✓– A	✓	✓	✓	P	✗	✗	US
com.stabxtom.thomson	?	?	?	?	?	?	?	?	?	?
com.withings.wiscale2	✓	✓	✓– A	✓	✓	✓	✓	✓	✗	FR

Table 5.4: Privacy policy analysis results (P - partly; A - anonym)

As observations from security analysis are being matched with information from privacy policies, relevant properties are summarized in Table 5.6. The data leakage feature indicates if information is collected that may fall out of the usage limitation principle. Examples are location tracking data, device identifier or any other data the phone or sensor might be recording. In other cases data might be collected and send to other destinations then the vendor's web servers. The data storage location has been extracted based on captured traffic. With the limitation of potential redirections behind this visible endpoint – something for example Google or Cloudfare are known for [158] – this location is assumed as the data's destination.

	Activ8rLives	Fitbit Aria	HAPI Connected Scale	iChoice iChoice app Medm app	Thomson TBS705	Withings WS-30 & WS-50 Body Cardio
data leakage	✗ data leaked; overprivileged app puts risk on updates	⚠ analytics data tracking; wlan ssid send	⚠ analytics tracking	✓ minimal data exchange and no leakage	✗ no data control; leaks device identifier	⚠ approx. location shared with vendor; modern analytics
data storage	UK	United States	Canada	United States	Europe	Europe
according to privacy policy	✓Europe	✓United States, Safe-Harbor	✗none mentioned	✓United States	✗no policy available	✓Europe

Table 5.6: Privacy related findings

As Discussed in Section 2.4.1 the safe-harbor agreement has been overruled by the European Court of Justice. Meanwhile privacy shield is still negotiated and although a gray zone remains for the moment, companies are encouraged to still adhere to the standards of safe harbor [159]. The last "according to privacy policy" row in Table 5.6 indicates that out of all solutions having a privacy policy all but HAPI follow this recommendation. Vendors will have to be prepared to adhere to new regulations and standards.

5.2 Discussion

5.2.1 Countermeasures to Main Issues

As a detailed discussion and assessment of related privacy risks is beyond the scope of this project, in this thesis 8 main issues have been chosen. For each of these issues counter measures including their adequacy and coverage to defeat or remediate the attack are defined.

ISS 1 *unencrypted data transmission* – refers to unencrypted data being transmitted over the (wireless) network and includes app-to-server and sensor-to-server communication (see *4.2.2 Getting SSL Right* and Table 5.2).

ISS 1.1 *app to server* – apps that lack SSL encryption leak all data they transfer; to capture this data an attacker tricks the victim to use a malicious hotspot [134] or forces the device through ARP spoofing to reconnect to that hotspot. Without connecting to the network a Wifi adapter set to promiscuous mode will be able to listen on data; remediation can only be achieved through a patch delivered by the application developer

ISS 1.2 *sensor to server* – essentially the same as in ISS 1.1 with the addition that proximity to a sensor sending unencrypted data is required if such sensor are stationary (e.g. smart scales); for (home) networks under full control by the user, network encryption on protocol level suffices to protect from threats outside the network; insiders are still able to pose threats

ISS 2 *improper certificate validation* – trust manager issues or invalid certificate validation (see *4.2.2 Getting SSL Right*); if the app trusts al certificates only a patch can solve this problem; in cases of broken certificate validation the user should avoid adding any certificate to their device's trust store

ISS 3	*missing tampering protection* – traffic and messages are not protected against tampering (see *4.2.3 Playing with Traffic*). To be successfully leveraged either of ISS 1.1, ISS 1.2 or ISS 2 is required. By ensuring that none of those three will work potential attacks are defeated.
ISS 4	*personal or PII data leaked* – unnoticed data leakage (see *5.1.2 Privacy Impacts*); requires patches by vendors in order remediate old library versions which are leaking deprecated device identifiers and in other cases this requires discussion with vendors and developers
ISS 5	*improper cryptography usage* – inadequate usage of cryptographic libraries and functions (see *4.3.3 HAPI Connected Scale* and *4.3.2 Fitbit Aria*); in cases of bad content encryption such as unsalted hashes the channel could be secured through traffic encryption
ISS 6	*weak password policies* – missing or weak password policies (see *4.2.4 The Power of Passwords*); as weak passwords are subject to brute force attacks stronger password policies are required to strengthen account login procedures
ISS 7	*account deletion* – flawed account deactivation or deletion processes (see *4.2.5 Managing Data*); this trust issue should be addressed by revising internal organizational processes
ISS 8	*overprivileged application* – overprivileged applications installed on device (see *4.2.6 Android Permissions*); developers should be asked to reduce permission to a minimum and user are required to provide justification through the new API (since Android 6.0) while asking for permissions

5.2.2 Solution Comparison

Since in all cases minor issues with respect to security and privacy properties have been found and different solutions show different strengths and weaknesses, no single superior solution can be identified. Solutions by iChoice with Swissmed Mobile app and Withings rank equally good whereas Thomson, Activ8rlives and HAPI fail largely. With focus on *privacy* the iChoice S1 solution with Swissmed Mobile app performed very well. With the exception of a dated SSL server connection this solution preserves user privacy not only by following a strict privacy policy including data collection limitation, but also by applying correct cryptographic methods were required. With data collection reduced to a minimum that is required to offer their services, users can be assured that their data is in good hands. Furthermore their web interface allows for detailed access settings and delegation principles. From a functional point of view it might not be the most attracting solution, but certainly user privacy is carefully protected. The Withings solutions also offer good user privacy but based on the amount and nature of the data they transmit (extensive crash/application logs, location of scale), it is seen as less privacy preserving. On the good end they keep their data in Europe making it subject to European legislation and regulations. Although Swissmed Mobile who are developing app and web page for iChoice are based in Switzerland and the US their servers are in the US only. This fact is not mentioned to the user but makes the collected data subject to US legislation which is quite different from European legislation as the debate around the safe-harbour agreement shows (see Subsection /refsubsec:health-information-privacy). On the lower end of the spectrum one can find Thomson with their scale TBS705. Their application not only leaks device identifiers to a Chinese server but also synchronises data with their own server, on which it remains inaccessible and unmanageable for the user. Upon request via email to the developer asking to access this data no response was received till the end of the project. Unsurprisingly a privacy policy is neither from Google Playstore nor from elsewhere on the web available.

Towards security Withings with their new Body Cardio scale is setting new standards. Their application employs stand-of-the-art security, their authentication is OAuth delegation based and the scale pairing process is based on shared secrets with the mobile app as a delegate. By adding SSL support their last weakness is addressed. iChoice S1 with Swissmed Mobile app employs even stronger security (certificate pinning) but connects to a less secure web server (SSL configuration issue). The Withings server configuration in combination with app and sensor device outperforms any other solution. The older Withings scales which are lacking SSL encryption are relatively secure on an encrypted home network. Worst among the scales is HAPI Connected Scale with HAPI Connected app. Whilst missing SSL encryption, it fails to use cryptography correctly, is highly overprivileged and logs extensive data to Android system log. Thomson TBS705 and Activ8lives Body Analyzer are equally bad in terms of security.

5.3 Project Evaluation

At the start of this project two hypothesis have been postulated. Firstly that mHealth solutions expose well known weaknesses which can be exploited as vulnerabilities and secondly that in order to discover these weaknesses a security and privacy analysis framework could be defined and run on the test set. The further document was then structured through asking the three main research questions addressing the security in terms of known issues and collected or leaked data, the privacy related impacts that could be inferred from privacy policies or retrieved from captured data and the question on how to compare different solutions and which recommendations can be deduced.
Designing the analysis framework a threat model has been defined as part of the literature review. Using the framework various security issues and privacy impacts have been identified. Sufficient evidence to accept both hypothesis is given with the analysis of scales and related findings. The methodology introduced by the framework proofed to be adequate so that answers to the three research questions are provided: Research Question 1 and sub questions are answered in *5.1.1 Security Issues*, Research Question 2 and sub questions in *5.1.2 Privacy Impacts* and finally Research Question 3 in *5.2 Discussion*.
Nevertheless there are several limitations to this study that have to be considered. Naturally this analysis is limited to the test set of sensor devices and apps specified. All results and findings are hence only valid for this set including specific versions of hardware and software Tables A.1 and A.3. Furthermore experiments are based on a privileged attack model which relies on MITMA. Whereas this serves the needs of efficient security analysis, actual behaviour and possible attacks in the wild will be different. Due to time constraints vendors and their developers have not been contacted for further insights. False positives might be included in evaluation criteria which earned a pass for specific solutions. With a project duration of three month and a start-up phase all devices, apps and user accounts with vendors have just occasionally been used. Hence this study does not include a long term evaluation and analysis of solutions. Disregarding long term usage of these solutions some malicious behaviour might remain undetected. In addition the study was only run on Android test devices. If one were to use the same sensor with Android and iOS simultaneously, issues specifically related to iOS need to be considered. With Bluetooth communication being out of scope parts of the results have been inferred rather than observed directly. For example if the traffic between app and server did not show any trace of firmware update func-

tionality, this feature was assumed to be missing for the sensor.

As discussed in *Section 3.3* designing and implementing the analysis framework was a time consuming exercise. However the time required was taken and the framework designed and implemented with due diligence as it would largely facilitate the investigations of mHealth devices.

Fig. 5.1 provides a high level overview on the overall process of this work. As part of *mH-PriSe* the weaknesses that have been defined in the threat model have been linked to test steps. These test steps yielded results and findings among which the main issues have been identified. Neither the test steps mentioned here nor the issues that have been listed are exclusive. Fig. 5.1 is meant to show how the work covers aspects of accepted standards such as the STRIDE approach in this case. STRIDE is an acronym for **S**poofing, **T**ampering, **R**epudiation, **I**nformation **D**isclosure, **D**enial of Service and **E**levation of Privilege. The capital letters in Fig. 5.1 indicate where STRIDE aspects are included. Properties Spoofing, Tampering, Information Disclosure are covered through dynamic analysis and penetration testing (see ISS 1, ISS 3, ISS 4). Repudiation is closely related as all tampering and manipulation attacks are executed in user authenticated sessions. Hence there is no possibility to back trace a traffic manipulation to an attacker. Elevation of Privilege can be referred to overprivileged apps and although the actual term implies more attacks which would be beyond our scope (see ISS 8). Although DDoS attacks have been tried their applicability for this area is limited as devices are usually not expected to be permanently online. Hence downtimes are part of the design and do not harm any asset.

Three different views to the contribution of this project have been thought: Security Analysts, Software Developer and End User. The analysis framework is published online and hopefully helps for future analysis. The results and recommendations should help Software Developer to improve their products. Of most interest to the End User will be the discussion and comparison as long as the information becomes more accessible.

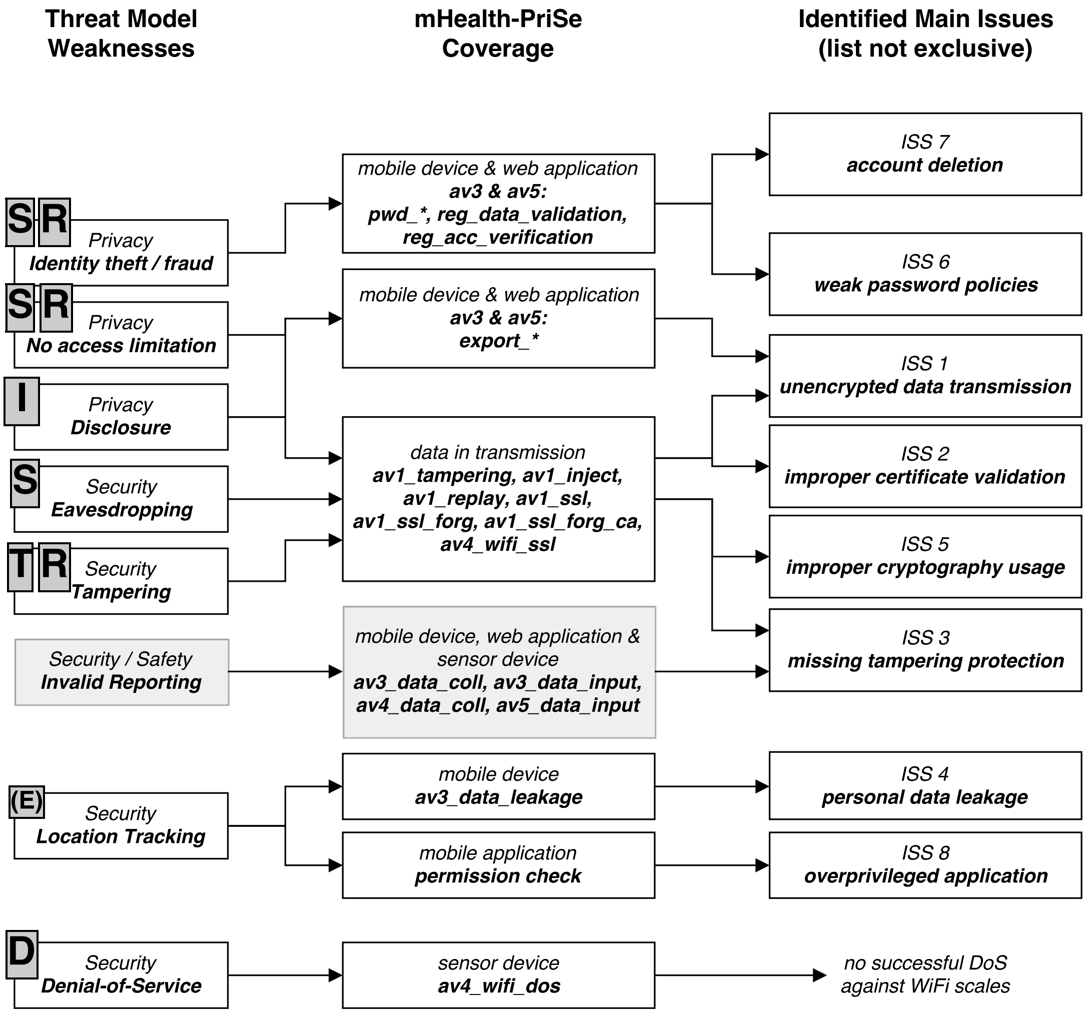

Threat Model Weaknesses
mHealth-PriSe Coverage
Identified Main Issues (list not exclusive)
S R Privacy Identity theft / fraud
S R Privacy No access limitation
I Privacy Disclosure
S Security Eavesdropping
T R Security Tampering
Security / Safety Invalid Reporting
(E) Security Location Tracking
D Security Denial-of-Service
mobile device & web application av3 & av5: pwd_*, reg_data_validation, reg_acc_verification
mobile device & web application av3 & av5: export_*
data in transmission av1_tampering, av1_inject, av1_replay, av1_ssl, av1_ssl_forg, av1_ssl_forg_ca, av4_wifi_ssl
mobile device, web application & sensor device av3_data_coll, av3_data_input, av4_data_coll, av5_data_input
mobile device av3_data_leakage
mobile application permission check
sensor device av4_wifi_dos
ISS 7 account deletion
ISS 6 weak password policies
ISS 1 unencrypted data transmission
ISS 2 improper certificate validation
ISS 5 improper cryptography usage
ISS 3 missing tampering protection
ISS 4 personal data leakage
ISS 8 overprivileged application
no successful DoS against WiFi scales

Figure 5.1: STRIDE coverage plus relation of main issues to threat model

Chapter 6

Conclusion

6.1 Findings and Summary of Work

The report at hand investigated mHealth solutions with a focus on smart scales for security and privacy issues. Out of 9 solutions in scope, for 7 severe security or privacy issues have been revealed and only two solutions were of commendable standard. Fig. 6.1 shows a comparison of the different solutions and the proportion of detected versus avoided issues. This comparison makes the strong assumptions that every issue has the same severity and implies that all solutions would have the same number of possible issues. Nevertheless it shows that among all solutions Thomson by far performed the worst whereas iChoice with SwissMed app and Withings Body Cardio performed among the best. As an example good example the SwissMed mobile application employs certificate pinning which defeats any MITMA attacks. Their web application allows for detailed privacy settings and their privacy policy follows OECD standards. In contrast, the Thomson solution does not reveal to the user where the data that has been collected is finally send. In fact, the user is not given any control over that data. While the application collects measurements from a connected scale, it also sends information to a Chinese advertising company including the device IMEI and more device identifying data. On top of the solution lacks any kind of traffic encryption. Other then for the Thomson solution, privacy policies have been of good standard, where they were provided. Though vendors can improve on privacy aspects in terms of account management (deletion/deactivation, verification) or information on data storage locations and data usage.

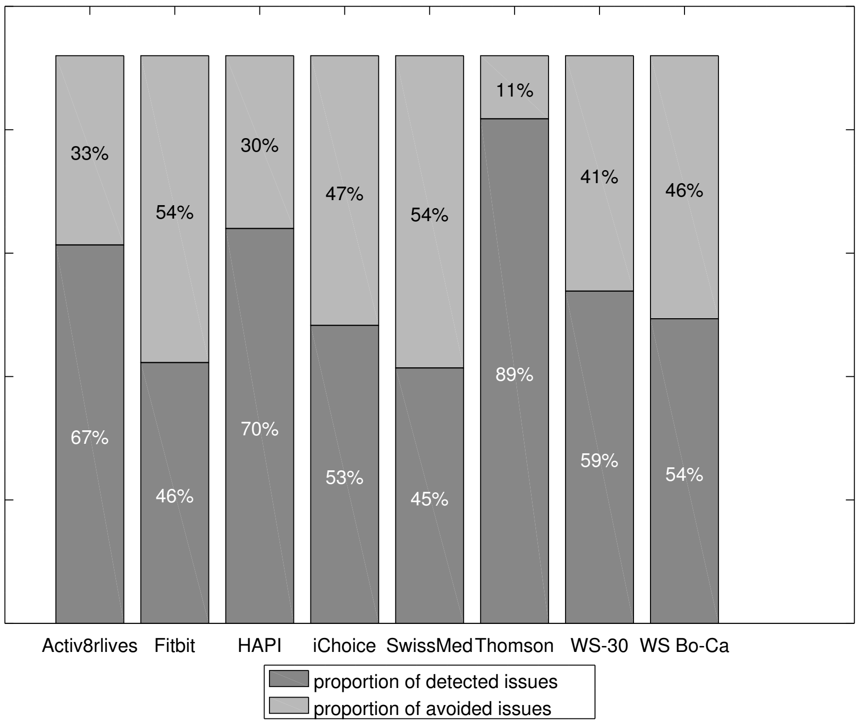


Figure 6.1: Comparison of solutions with number of issues detected and avoided

Users have to assume vulnerabilities in new solutions they buy. It is up to them to take precautions adding further security layers to their network or changing their usage behaviour. This on the other hand also shows that research in those areas is important and its findings are of high value to users. Results and recommendation consequently have to be documented in an accessible way for non experts (see *6.3 Dissemination of Results*).

6.2 Future Work

Several different areas provide promising starting points for further analysis. Although this project provides a comprehensive view on mHealth solutions further investigation will be able to provide more insights. To gain such insights further analysis of Bluetooth communication and sensor device firmware is a good starting point. Detailed information on these parts will allow to entirely reverse engineer communication protocols used by scales, e.g. to understand the Withings communication and pairing process. A good starting point are firmware packages which could be retrieved for

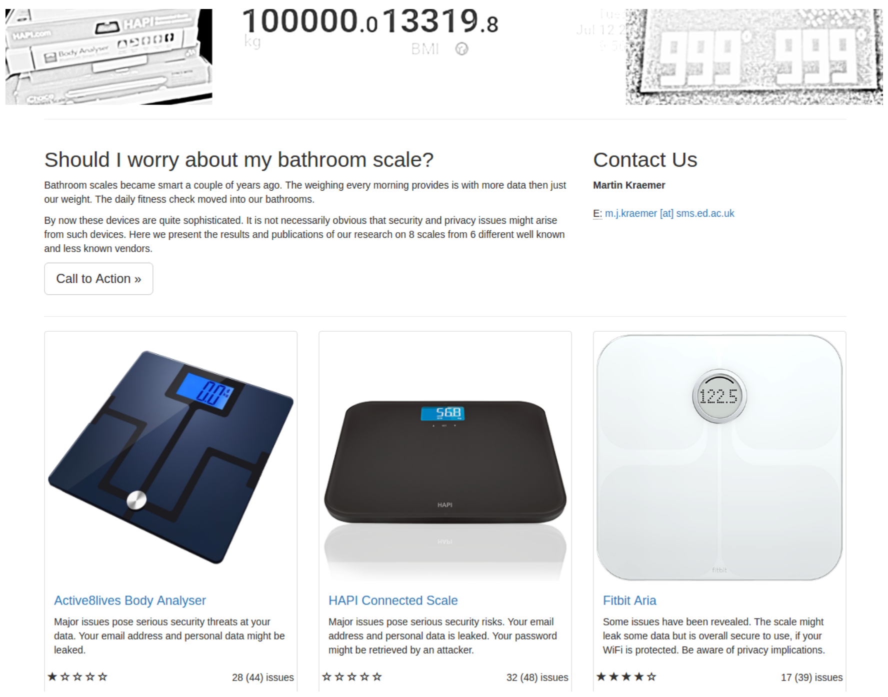
Withings devices and Fitbit Aria. Another interesting experiment would be long term observations of solutions. With overprivileged apps offering a surface for malicious software, future attacks are theoretically possible.
Another route to follow is a detailed privacy impact assessment and risk analysis. For these matters the data set however has to be larger. Nevertheless such analysis would yield useful metrics for comparison and evaluation. A good starting point might be the DREAD methodology [160] or more detailed a scoring and classification scheme like the Common Vulnerability Scoring System (CVSS) [161].
A major part of this work has been to analyse network traffic between different components of a solution. Thus all solutions are meant to serve the same purpose, their traffic largely differs in terms of the chosen data representation (JSON, XML, proprietary binary), number of requests, amount of data and destination. In an approach similar to Alan who identified applications based on their start-up TCP/IP traffic [162], ranking or classification of applications towards privacy levels could be interesting. The before mentioned differences in observed traffic could make properties an applications traffic would be evaluated on. This could be a nice application for data science methods to the mHealth sector.
Meanwhile the analysis framework itself can always be extended with additional tools. The version developed as part of the project is certainly a starting point up on which further extensions could be developed. On another note some steps could be further automated, especially some of the manually executed steps in the dynamic analysis part.

6.3 Dissemination of Results

The maximise the impact of this research and to provide adequate information to the target groups that have been identified in the Introduction (Security Analyst, Software Developer, End User) the findings and results of this work will be publicised through different channels.

Web Page With the intention to present the results and findings in an accessible way for non Computer Scientists, a web page will be created (see Fig. 6.2). This web page will be further populated with publications and additional information once issues have been disclosed with vendors.

`http://martin-kraemer.net/mhealth.html`



Should I worry about my bathroom scale?

Bathroom scales became smart a couple of years ago. The weighing every morning provides is with more data then just our weight. The daily fitness check moved into our bathrooms.

By now these devices are quite sophisticated. It is not necessarily obvious that security and privacy issues might arise from such devices. Here we present the results and publications of our research on 8 scales from 6 different well known and less known vendors.

Contact Us

Martin Kraemer

E: m.j.kraemer [at] sms.ed.ac.uk

Active8lives Body Analyser

Major issues pose serious security threats at your data. Your email address and personal data might be leaked.

★☆☆☆☆ 28 (44) issues

HAPI Connected Scale

Major issues pose serious security risks. Your email address and personal data is leaked. Your password might be retrieved by an attacker.

☆☆☆☆☆ 32 (48) issues

Fitbit Aria

Some issues have been revealed. The scale might leak some data but is overall secure to use, if your WiFi is protected. Be aware of privacy implications.

★★★★☆ 17 (39) issues

Figure 6.2: Screenshot of web page `http://martin-kraemer.net/mhealth.html`

Open Source *mH-PriSe* which has been developed as part of this project is publicly available on GitHub under the Apache 2 license. The framework may be used by anyone and can be enhanced or changed freely.
`https://github.com/markraemer/mhealth-priv-sec-analysis/`

Security Research Conferences The poster "Weighing in eHealth Security" [163] is based on this work and has been accepted to the ACM Conference on Computer and Communications Security in Vienna, Austria (this is one of the top conferences worldwide in the field, the poster was one of 38 accepted from 92 submissions). Work on a workshop paper with focus on *mH-PriSe* and an additional paper looking at users' perception of privacy and security of issues with eHealth in general and smart scales in particular, is still ongoing.

Vendor Notification The notification of vendors and their feedback is still to be initiated. So far questions have just been asked through their support channels targeting at specific actions or information.

Press Coverage As similar results have gained coverage by media, currently plans are made to contact the School of Informatics' public relations office to discuss options for further publication

Appendix A

Appendix

A.1 Data Set

Table A.1: Smart scales in scope

manufacturer	model	mobile app	release date	price (GBP)	WiFI	BLE	FW
FitBit	aria	com.fitbit.FitbitMobile	Apr, 2012	99.99	✓	✗	V39
Withings	WS-30	com.withings.wiscale2	Dec, 2012	79.99	✓	✓	881
Withings	WS-50	com.withings.wiscale2	Mar, 2013	129.99	✓	✓	1221
Activ8rLives	Body Analyser	com.activ8rlives.mobile	Apr, 2014 *	59.99	✗	✓	
Thomson	TBS705	com.stabxtom.thomson	Aug, 2014	59.99	✗	✓	
HAPI	Connected Scale	com.hapiconnect	May, 2015 *	49.99	✗	✓	
iChoice	S1	com.medm.medmwt.diary com.medm.ichoice.diary	Aug, 2015 *	44.69	✗	✓	
Withings	Body Cardio	com.withings.wiscale2	Jun, 2016	139.95	✓	✓	941

Table A.3: Details on investigated apps extracted from Google Play Store

package	title	version	company	installs (k)	rating	releasedate	pripol
com.activ8rlives.mobile	Activ8rlives Health & Food	3.16.4	Aseptika Ltd	10	2.87	2016-05-23	http://data.activ8rlives.com/Terms/TCs.htm
com.fitbit.FitbitMobile	Fitbit	2.29	Fitbit, Inc.	10	3.94	2016-07-11	http://www.fitbit.com/privacy
com.hapiconnect	HAPI Connect	1.2.8_US_server	HAPILABS	100	2.78	2016-07-08	http://www.hapi.com/privacypolicy
com.medm.ichoice.diary	iChoice Life Pro	1.7.8	ChoiceMMed America Co.	1	2.96	2015-03-17	http://ichoicelife.com/support/privacy
com.medm.medmwt.diary	MedM Weight	2.0.26	MedM Inc	500	4.44	2016-04-21	https://health.medm.com/privacy
com.stabxtom.thomson	Thomson Healthcare	v1.1-B018	Thomson Healthcare	1	3.45	2015-10-05	
com.withings.wiscale2	Health Mate	2.16	Withings	500	3.51	2016-06-07	http://vitrine.withings.com/privacy-terms

A.2 Analysis Framework

A.2.1 Functional Overview

Table A.5: Framework functional overview - prepare

step	tool	data	purpose
maintain apps list	linux tools	package ID in properties file	maintains an list of application (package) ID which are the unique identifier for each application; typically this includes one application per solution but not exclusively
maintain tools app list	linux tools	package ID in properties file	same purpose as the normal application list but for applications supporting the analysis
download and install tools	googleplay-api [164]	APK in file system as backup	download the tool APK files from Google Playstore and install them on the device
download apps	googleplay-api [164]	APK and screen shot in file system; details such as rating, popularity and category in database	all apps which have been added to the list are downloaded and additional information from Google Play Store is stored in the database; furthermore a screenshort documents the Google Play Store view for a user
install apps	Android Debug Bridge	none	one or all apps can be installed to the device via Android Debug Bridge, the APK is taken from the file system

Table A.6: Framework functional overview - static

step	tool	data	purpose
extract base information	Android Asset Packaging Tool	version name, version code, icon, permission, filesize in database	basic information (version, permissions and more) is extracted from APK files and stored in the database
Mallodroid	Mallodroid [70] based on Androguard [110]	mallodroid results in database	research tool Mallodroid is executed for each APK file to detect and evaluate internet usage towards badly implemented SSL use (failing hostname verification or or overly trusting CA)
Application Attack Surface	Drozer [109]	drozer analysis results (intents, broadcasts, activities and services) in database	Drozer is used to identify exported intents, broadcasts, activities and services as these expose potential weaknesses

Developer Certificate Info	OpenSSL	information on certificate parameters (validity, key length and others) in database	applications signed with weak developer certificates can be replaced on device without the users consent as an attacker manages to reporduce a developers certificate
Detect Addons	Addon Detector App [116]	libraries and frameworks used to develop the application in database	Addons can be development libraries , known advertising or malware libraries and hence include potential threats
Check Obfuscation	Androguard [110] and Python Script [14]	obfuscation score calculated based on decomplied sourcecode	Obfuscation through e.g. Google's ProGuard tool is a very common way to hide information inside applications
Malware Check	ExplainDroid [118], Evicheck [117]	results from ExplainDroid and Evicheck in database	ExplainDroid and EviCheck are run on APK files to identify potentially malicious apps

Table A.7: Framework functional overview - dynamic

step	**tool**	**data**	**purpose**
run test case	Vysor [123], recordmydesk-top [124], mitmproxy [106], tshark [107], Android Debug Bridge (shell, monkeyrunner, logcat, hci bluetooth log), DroidWall [125]	meta information on the test run and commented results in database (test case and test steps); captured files (traffic, screencast, HCI log, logcat, app data) in file system	picks a predefined test case from the database and starts a test run for that case while prompting the user for feedback on each test step assigned to that test case
document test case		documented test steps with comment and rating in database	previously recorded test cases can be further documented by adding comments for already documented test steps or adding entirely new test steps
show missing test cases		console output	displays an overview of missing test cases and steps

Table A.8: Framework functional overview - post

step	**tool**	**data**	**purpose**
extract URL	script for mitmproxy [106], ipinfo.io database; matplotlib [126]	URL, hostname, IP in database	extract URL and IP from previously recorded traffic sessions and creates traffic map

scan pcap file	tcpdump, tshark, linux tools	IP addresses shown on console output	parses IP addresses from tShark recorded pcap files and finds IP missing from MITMproxy recorded HTTP traffic (covers TCP-only traffic)
check server SSL	ssllabs.com	classification and link to detailled report in database	runs a SSL check on extracted hosts to identify their basic configuration and commonly known vulnerabilities
show recording	none	none	from a list of recorded cases opens the recording folder or captured traffic in mitmproxy

Table A.9: Framework functional overview - export

step	**tool**	**data**	**purpose**
static analysis - tables to csv		addons, permission, certificates, privacy policy, malware and obfuscation data in csv files	exports collected data to csv files to for further use in reports
static analysis - latex templates			generates latex templates for latex datatools package
dynamic analysis - results (HTML, PDF)		overview report with issue count and detailled report with comments as PDF, HTML in file system	exports results of dynamic analysis in HTML and PDF as an overview including aggregated issue counts

Table A.10: Framework functional overview - tools and helper

step	**tool**	**data**	**purpose**
convert APK to SMALI	apktool [151]	smali code in file system	converts the APK file to SMAIL code to enable further manual investigation of e.g. cryptography usage in these libraries
repackage from SMALI and install	apktool [151], jarsigner, Android Command Line Tools (zipalign), Android Debug Bridge	APK file in file system and installed on device	repackages SMALI code after modification, signs, aligns and replaces the original version with the changed version on device
convert APK to JAR	dex2jar [165]	jar file in file system	decompiles APK to dex and from dex to jar in order to have readable java code for further code analysis
open APK in GUI			opens an APK file in JD GUI after it has been converted to a jar file
stop all 3rd party apps		Android Debug Bridge Shell	part of sanitization procedure that stops all apps on device

validate log folder structure		validates log folders in the file system with existent database entries for integrity (a MySQL trigger should delete log folders alongside db entries normally - MySQL UDF plugin)

A.2.2 Components

Table A.11: Framework software components

name	version	usage
Kali Linux	2016.1	Penetration Testing and Ethical Hacking Linux Distribution
PyCharm	Community 2016.1	Python development environment
Android SDK Tools	24.0.2	Android Build and Development Tools
Android Studio	2.1.2	developing and reverse engineering Android applications
MySQL Server	5.6.27-2 (Debian)	database server used to store experiment results and documentation
Apache	Apache/2.4.20 (Debian)	web pages used as administration interface to database
PHP	7.0	dependency for phpMyAdmin
Androguard	2.0	Android decompiler and analysis tools
Mallodroid	31-12-2016	detecting SSL implementation weaknesses in Android applications
Drozer	2.3.4	Android penetration testing tool suite
apktool	2.1.1	dissassembler for Android Dalvik byte code to smali and back (baksmali)
OpenSSL	1.0.2e 3 Dec 2015	Cryptographic toolkit
dex2jar	reader-1.15 translator-0.0.9.15 ir-1.12	decompiling Android dex to Java jar files
Wireshark / tShark	2.0.1	network traffic capture and analysis
mitmproxy	0.15	Man-In-The-Middle proxy used to forge certificates, traffic manipulation, fabrication and more
MySQL Workbench	6.3.7	administration interface to MySQL server
hostapd	v2.3	IEEE 802.11 AccessPoint management
dnsmasq	2.76	network infrastructure for small networks
googleplay-api	commit c463cbe	downloading apk files from google play store
Vysor	1.1.3	Android screen mirroring
Addon Detector	3.24	Android app reading addon libraries and frameworks from installed apps
EviCheck	0.1	Digital Evidence for Android - Malware classification
ExplainDroid	0.1	text mining based approach for malware classification
recordmydesktop	0.3.8.1	desktop screen scraping
DroidWall	1.5.7	Android app Android firewall application
Device Identifier	1.3.2	Android app displays several device identifier like IMEI, MAC and others
tPacketCapture	2.0.1	Andorid app capturing network traffic on the phone

Table A.12: Framework hardware components

name	**version**	**usage**
Lenovo Thinkpad X230		main workstation to run the framework and share internet connection
Atheros Wifi Card	AR9271	external Wifi adapter to expose hotspot
LG Nexus 5	Android 6.0 (rooted)	test device used to run experiments
External Harddrive 4GB		used to store results

A.3 Results Static Analysis

Table A.14: Malware detection tools run on APK files

package	**EviCheck**	**ExplainDroid**
com.activ8rlives.mobile	valid	benign
com.fitbit.FitbitMobile	valid	benign
com.hapiconnect	valid	malicious
com.medm.medmwt.diary	valid	benign
com.medm.ichoice.diary	valid	benign
com.stabxtom.thomson	valid	benign
com.withings.wiscale2	valid	benign

Table A.16: Developer certificate information 1

package	**certsigalgo**	**certnb**	**certna**	**certpka**	**certpkl**	**certsn**
com.activ8rlives.mobile	dsaWithSHA1	Oct 16 17:31:28 2012	Mar 3 17:31:28 2040	dsaEncryption	0	1350408688 (0x507d99f0)
com.fitbit.FitbitMobile	sha1WithRSAEncryption	Mar 16 21:17:50 2012	Dec 18 21:17:50 2066	rsaEncryption	2048	1331932670 (0x4f63adfe)
com.hapiconnect	sha256WithRSAEncryption	Oct 13 02:02:19 2014	Sep 30 02:02:19 2064	rsaEncryption	2048	1578608323 (0x5e17a6c3)
com.medm.ichoice.diary	sha256WithRSAEncryption	Oct 13 09:35:23 2014	Oct 7 09:35:23 2039	rsaEncryption	2048	113008560 (0x6bc5fb0)
com.medm.medmwt.diary	dsaWithSHA1	Sep 4 10:47:31 2013	Jun 7 10:47:31 2068	dsaEncryption	0	2078463580 (0x7be2d65c)
com.stabxtom.thomson	sha256WithRSAEncryption	Sep 3 09:40:21 2014	Aug 21 09:40:21 2064	rsaEncryption	2048	884689147 (0x34bb48fb)
com.withings.wiscale2	sha1WithRSAEncryption	Oct 28 16:00:10 2010	Oct 15 16:00:10 2060	rsaEncryption	1024	1288281610 (0x4cc99e0a)

Table A.18: Developer certificate information 2

package	certissuer	certsubject
com.activ8rlives.mobile	C=UK, ST=Cambridgeshire, L=Huntingdon, O=Aseptika Ltd, OU=Activ8rlives, CN=Kevin	C=UK, ST=Cambridgeshire, L=Huntingdon, O=Aseptika Ltd, OU=Activ8rlives, CN=Kevin
com.fitbit.FitbitMobile	C=US, ST=California, L=San Francisco, O=Fitbit, Inc., OU=Unknown,	C=US, ST=California, L=San Francisco, O=Fitbit, Inc., OU=Unknown,
com.hapiconnect	None	None
com.medm.ichoice.diary	C=CN, ST=beijing, L=beijing, O=choicemmed, OU=choicemmed,	C=CN, ST=beijing, L=beijing, O=choicemmed, OU=choicemmed,
com.medm.medmwt.diary	C=CH, ST=Zug, L=Zug, O=Swissmed Mobile AG, OU=Unknown, CN=Michael	C=CH, ST=Zug, L=Zug, O=Swissmed Mobile AG, OU=Unknown, CN=Michael
com.stabxtom.thomson	O=Thomson	O=Thomson
com.withings.wiscale2	C=33, ST=92, L=Issy les Moulineaux, O=R&D, OU=Mobile Soft, CN=Carlos	C=33, ST=92, L=Issy les Moulineaux, O=R&D, OU=Mobile Soft, CN=Carlos

Table A.20: Results of manual privacy policy analysis

package	**URL**	**version**	**Country**	**AP**	**SSP**	**OP**	**PSP**	**IPP**	**intUsage**	**3rdPartyStorage**	**Merger**	**3rdPartyForward**
com.activ8rlives.mobile	http://www.activ8rlives.com/legal-information/privacy-and-cookies-policy.html	24 Jul 2016	UK	✗	✓	✓	P	✓	✓	✓	✓	✓– A
com.fitbit.FitbitMobile	http://www.fitbit.com/privacy	10 Aug 2014	US	P	✓	✓	✓	✓	✓	✗	✓	✓– A
com.hapiconnect	http://www.hapi.com/privacypolicy	01 July 2013	US	✓	✓	✓	P	✓	P	✗	✓	✓– A
com.medm.medmwt.diary	https://health.medm.com/privacy	14 May 2014	?	✗	✓	✓	✓	✓	✓	✗	✗	✓– A
com.medm.ichoice.diary	https://go.ichoicelife.com/en/privacy	26 Mar 2013	US	P	✓	✓	✓	✓	✓	✗	✗	✓– A
com.stabxtom.thomson		?	?	?	?	?	?	?	?	?	?	?
com.withings.wiscale2	http://vitrine.withings.com/privacy-terms	26 Apr 2015	FR	✓	✓	✓	✓	✓	✓	✓	✗	✓– A

Table A.22: Addons and libraries found in APK files

package	**addons**
com.fitbit.FitbitMobile	Amazon AWS, Android Support Library, BoltsFramework, Butter Knife, Facebook, Google Analytics, Google Maps, Google Play Services, Jackson, Multidex, Picasso
com.hapiconnect	aChartEngine, Android Support Library, Fabric, Gson, Protobuf, ZXing
com.medm.medmwt.diary	Android NDK, Android Support Library, Google Play Services
com.medm.ichoice.diary	Android NDK, Android Support Library, Google Play Services
com.stabxtom.thomson	aChartEngine, Android Asynchronous Http Client, Android Support Library, BoltsFramework, DragSortListView, Facebook, Gson, Otto, Twitter4j, Umeng, Universal Image Loader, Volley, WeChat, XStream, ZXing
com.withings.wiscale2	Android NDK, Android Support Library, Butter Knife, Calligraphy, Crashlytics, DragSortListView, EventBus, Glide, Google Play Games, Google Play Services, Gson, HoloColorPicker, Joda, Multidex, OkHttp, Okio, ORMLite, Retrofit, Smack, ViewPagerIndicator

permission	**com.activ8rlives.mobile**	**com.fitbit.FitbitMobile**	**com.hapiconnect**	**com.medm.ichoice.diary**	**com.medm.medmwt.diary**	**com.stabxtom.thomson**	**com.withings.wiscale2**
android.permission.DISABLE_KEYGUARD	-	-	-	-	X	-	-
com.google.android.providers.gsf.permission.READ_GSERVICES	X	X	-	-	-	-	-
android.permission.READ_LOGS	-	-	X	-	-	-	-
android.permission.ACCESS_COARSE_LOCATION	X	-	-	-	X	-	X
android.permission.BLUETOOTH	X	X	X	X	X	X	X
android.permission.ACCESS_WIFI_STATE	X	-	X	-	-	X	X
android.permission.INTERNET	X	X	X	X	X	X	X
com.withings.wiscale2.permission.C2D_MESSAGE	-	-	-	-	-	-	X
activ8map.Android.permission.MAPS_RECEIVE	X	-	-	-	-	-	-
android.permission.BLUETOOTH_ADMIN	X	X	X	X	X	X	X
android.permission.NFC	-	X	-	-	X	-	-
android.permission.ACCESS_FINE_LOCATION	X	X	-	-	-	-	X
com.google.android.c2dm.permission.RECEIVE	-	X	-	-	X	-	X
android.permission.ACCESS_NETWORK_STATE	X	X	X	-	-	X	X
android.permission.BLUETOOTH_PRIVILEGED	X	-	X	-	-	-	-
android.permission.GET_TASKS	-	-	X	-	X	X	-
com.medm.medmwt.diary.permission.C2D_MESSAGE	-	-	-	-	X	-	-
android.permission.WRITE_EXTERNAL_STORAGE	X	X	X	X	X	X	X
android.permission.RECEIVE_SMS	-	X	-	-	-	-	-
android.permission.MANAGE_ACCOUNTS	-	-	X	-	-	-	-
android.permission.READ_EXTERNAL_STORAGE	X	X	X	X	X	X	X
android.permission.RECEIVE_BOOT_COMPLETED	-	X	-	-	X	-	X
android.permission.DOWNLOAD_WITHOUT_NOTIFICATION	-	-	X	-	-	-	-
com.fitbit.FitbitMobile.permission.C2D_MESSAGE	-	X	-	-	-	-	-
android.permission.CHANGE_CONFIGURATION	-	-	-	-	-	X	-
android.permission.FLASHLIGHT	-	-	X	-	-	X	X
android.permission.READ_PHONE_STATE	-	X	X	-	-	X	-
android.permission.MOUNT_UNMOUNT_FILESYSTEMS	-	-	X	-	-	-	-
android.permission.VIBRATE	-	-	X	-	-	X	X
android.permission.SYSTEM_ALERT_WINDOW	-	-	-	-	X	-	-
com.samsung.android.providers.context.permission.WRITE_USE_APP_FEATURE_SURVEY	-	-	-	-	-	-	X
android.permission.CAMERA	-	X	X	-	-	X	X
android.permission.WAKE_LOCK	-	X	-	X	X	-	X
android.permission.ACCESS_DOWNLOAD_MANAGER	-	-	X	-	-	-	-
android.permission.CHANGE_WIFI_STATE	-	-	X	-	-	X	-
android.permission.RECORD_AUDIO	X	-	-	-	-	-	-
android.permission.READ_CONTACTS	-	X	-	-	-	-	-
android.permission.GET_ACCOUNTS	-	X	X	-	-	-	X

A.4 Results Dynamic Analysis

Table A.25: Experiment results – AV1 data transmission

test step	rating	desc	Activ8rLives Body Analyser com.activ8rlives.mobile	Fitbit Aria com.fitbit.FitbitMobile	HAPI Connected Scale com.hapiconnect	iChoice S1 com.medm.medmwt.diary	iChoice S1 com.medm.ichoice.diary	Thomson TBS705 com.stabxtom.thomson	Withings (all) com.withings.wiscale2
av1_ssl_forg_ca	vulnerable: 1 / not vulnerable: 0	in addition to data_com_ssl_forg the trusted CA certificate is installed	1 – working as without CA certificate	1 – working	1 – NO SSL	0 – not working	1 – vulnerable as for other	1 – NO SSL	1 – working
av1_ssl_forg	vulnerable: 1 / not vulnerable: 0	secure and correct implementation of communication encryption	1 – working	0 – not working	1 – no SSL usage	0 – not working	1 – yes, trustmanager issue	1 – working	0 – not working
av1_ssl	not using SSL: 1 / using SSL: 0	is using SSL	0 – using SSL	0 – using ssl	1 – no SSL	0 – using SSL	0 – yes	1 – no SSL	0 – using SSL
av1_replay	vulnerable: 1 / not vulnerable: 0	finding some way to replay information to the server	1 – replaces existing record	1 – possible and listed by sync time on server	0 – simple replay is not possible due to client side generated signatures	0 – replay is not possible as every record comes with a separate uuid	0 – no; UUID defeats a simple replay attack	1 – session stealing and replay attacks possible	1 – replay attacks possible within session
av1_inject	vulnerable: 1 / not vulnerable: 0	inject new data to the stream	1 – unauthenticated and just mapped by user ID	1 – possible based on reverse engineering the transfer protocol	1 – recomputing signature MAC possible; injection possible	1 – injection is only possible as long as the user provides an authenticated session	1 – auth token valid for longer time; UUID can be generated	1 – injection is possible	1 – possible as long as scale has session and this can be used

av1_tamper	not vulnerable: 0 / upper limit e.g. weight: 1 / no limit or validation: 2	tampering the data before it can reach the server defeating input validation	2 – weights up to 99999999 kg can be entered	2 – MAC can be recomputed and message changed	2 – possible at any point in time	1 – max weight is validated and limited to 300kg	1 – changes possible within the limits of up to 300 kg	2 – tampering successful; data not validated by server	2 – server not validating data; tampering possible with high values
number of issues			6.0	5.0	6.0	2.0	4.0	7.0	5.0

Table A.26: Experiment results – AV2 mobile device

test step	rating	desc	Activ8rLives Body Analyser com.activ8rlives.mobile	Fitbit Aria com.fitbit.FitbitMobile	HAPI Connected Scale com.hapiconnect	iChoice S1 com.medm.medmwt.diary	iChoice S1 com.medm.ichoice.diary	Thomson TBS705 com.stabxtom.thomson	Withings (all) com.withings.wiscale2
av2_export_backup	data not encrypted: 2 / data in backup encrypted: 1 / no data stored: 0	data is being backed up using the android backup command	0 – no data in backup found	2 – database not encrypted and shared prefs in clear text	2 – database unencrypted, password in clear text in shared prefs,	1 – sqlite databases not encrypted; shared preferences are obscured and encrypted	2 – data in caches but unencrypted	2 – unencrypted database	2 – database not encrypted
av2_export_logging	information available: 1 / no information: 0	investigate the Android log for unencrypted information	0 – no suspicious data in logging information	0 – no information in log file	1 – detailed information on each reqeust	0 – no information leaked in logcat	0 – no additional logging information	1 – information on the recorded data and partly http traces	0 – no information written to logcat
number of issues			0.0	2.0	3.0	1.0	2.0	3.0	2.0

Table A.27: Experiment results – AV3 mobile device

test step	rating	desc	**Activ8rLives Body Analyser com.activ8rlives.mobile**	**Fitbit Aria com.fitbit.FitbitMobile**	**HAPI Connected Scale com.hapiconnect**	**iChoice S1 com.medm.medmwt.diary**	**iChoice S1 com.medm.ichoice.diary**	**Thomson TBS705 com.stabxtom.thomson**	**Withings (all) com.withings.wiscale2**
av3_reg_pwd_policy	no policy: 2 / min characters: 1 / mixed charsets: 0	is a password policy enforced	2 – no password policy	n/a – n/a	1 – min 6 characters	1 – min 6 character	1 – min 6 characters	1 – 6-16 characters	2 – no hint to any kind of password policy could be found
av3_data_coll	informational	data collected and send by the mobile app	– normal data collection of pictures and other details on the phone	– weight can be entered and other fitness data	– not possible to enter data manually	– manual input only and photo capture to set profile picture	– weight can be entered manually and is limited to 300	– weight data and phone information	– SSL encrypted as opposed to scale
av3_export_sdcard	unsecure feature: 1 / secure feature: 0	can data be exported to SD card or other locations on device and is that data encrypted	n/a – n/a	n/a – no export possible	n/a – n/a	n/a – n/a	n/a – n/a	n/a – no option	n/a – no export option
av3_export_cloud	unsecure feature: 1 / secure feature: 0	data being exported somewhere into the cloud (dropbox et al)	n/a – n/a	n/a – no export possible	n/a – screenshots can be exported which show some graph with weight overview	n/a – n/a	n/a – n/a	n/a – no possibility	n/a – no export option in app
av3_reg_data_input	informational	Specifies the data fields and types which are required as inputs	– weight, height, waistline, birthdate and bio data, fitness goal and setps towards the goal basic validation on weight, height and waistline	n/a – n/a	– username, email, weight and height, and password	– email, password and later bio data	– name, email, gender optionally later weight, height and profile picture	– email, password, name, gender, name, birthday, height, weight, waistline, picture	– select scale n email and password n firstname, lastname, birthdate, height, weight

av3_data_wipe	unsecure feature: 1 / secure feature: 0	can data be wiped from the mobile application	1 – no possibility to wipe data from inside the app	1 – no feature	1 – not possible from within the app	1 – no data wipe option	1 – no option	1 – no feature available; no contact information	1 – no feature in app
av3_code_comment	informational	any comment on the code	– app crashing with Android 6.0.1 so reverting back to Android 6; also App is Xamarin based and running in Monodroid VM on Android which is some C style based implementation	tbd – tbd	– passwords are just hased with MD5 no salt; after re login signature looks suspicious as no password is transmitted - seems to be generated based on current time, previous access token and some generat random number; but all this information is available to an attacker	– code is somewhat supprinsingly just a few lines and decompiled code doesn't show all the information hence something must have been applied to the code also certificate pinning is implemented and had to be removed in order to run further tests	– java coding with native dependencies	– code shows many different frameworks and addons integrated with the solution (also see addons table) the app itself shows less possibilities to make actually use of these frameworks and addons the password seems to be MD5 hashed but since the communication is not encrypted and the hash is not randomiyed that is pointless, simply use the hash value to execute anything	– tbd
av3_pwd_change_freq	no period or frequency: 2 / either one: 1 / both: 0	password change period (must) and frequency (can)	n/a – can't change password from within the app	n/a – n/a	2 – nothing enforced	n/a – n/a	n/a – n/a	2 – no password change policy or frequency enforced	2 – n/a
av3_pwd_change_policy	no policy: 0 / min characters: 1 / mixed charsets: 2	password policy enforced on password change	n/a – n/a	n/a – n/a	1 – min 6 characters	n/a – n/a	n/a – n/a	1 – min 6 character	n/a – n/a

av3_reg_pripol_link	no link or information: 1 / link available: 0	link to privacy policy or any kind of information on such topic	0 – user acceptance for terms and conditions required to use app	n/a – n/a	1 – no explicit link visible	1 – not visible during registration but linked from within the app	1 – no link visible	1 – no link to privacy policy apparent	0 – link to privacy policy is provided http://www.withings.com/uk/en/legal
av3_data_leakage	no data leaked: 0 / complies with privacy policy: 1 / data leaked beyond privacy policy: 2	identifiable for the user or the phone that is being collected	2 – xamarin framework extensive event tracking and device fingerprinting	0 – device fingerprinting and application usage tracking but without any personal data; all data is being send to mixpanel.com	2 – no SSL; passwords MD5 hashed without salt	1 – as far as captured no data was being leaked	0 – no leakage identified	2 – Umeng library sending device identifier (IMEI, MAC); reveals full details during pairing as serial numbers...	0 – none
av3_export_social	unsecure feature: 1 / secure feature: 0	export variants to social media and data encryption while exporting	n/a – n/a	0 – connection to facebook possible but for information only	0 – screenshot of the graph can be shared with any 3rd party app	n/a – n/a	n/a – n/a	0 – twitter, facebook and email but none of them worked	n/a – just invitations through various channels
av3_reg_acc_verif	none: 2 / email account verification: 1 / two factor authentication 0	email account verification or two factor authentication	2 – received no email on account verification/ registration	n/a – n/a	2 – not at all	2 – no - but received email with pointers to privacy statements and similar stuff	2 – received registration confirmation email	2 – no verification email received	2 – no account verification; no registration confirmation via email
av3_pwd_change_reuse	yes, unlimited: 0 / no, but history can be bypassed (see change freq): 1 / no without limitation: 2	can a password be reused and how much difference is required	n/a – n/a	n/a – n/a	1 – last password can't be reused	n/a – n/a	n/a – n/a	2 – unlimited reuse of passwords	n/a – n/a
av3_reg_data_validation	data not validated: 1 / data is validated: 0	Comments on data validation during user registration: weight, heitgth, email...	0 – basic validation on weight, height and waistline	n/a – n/a	0 – weight and height only	0 – height and weight are limited; no other validation	1 – limited to 300 height and weight	0 – limited values for weight, height, waistline, age	0 – high weights accepted, mail and password checked
number of issues			7.0	1.0	11.0	6.0	6.0	12.0	7.0

Table A.28: Experiment results – AV4 sensor device

test step	rating	desc	Activ8rLives Body Analyser com.activ8rlives.mobile	Fitbit Aria com.fitbit.FitbitMobile	HAPI Connected Scale com.hapiconnect	iChoice S1 com.medm.medmwt.diary	iChoice S1 com.medm.ichoice.diary	Thomson TBS705 com.stabxtom.thomson	Withings WS-30 & WS-50 com.withings.wiscale2	Withings Body Cardio com.withings.wiscale2
av4_wifi_ssl	0 - correct ssl usage / 1 - vulnerable ssl usage / 2 - no ssl usage	ssl usage by sensor device	n/a – n/a	2 – no SSL	n/a – n/a	n/a – n/a	n/a –	n/a – n/a	2 – 2	SSL – 0
av4_pairing_process	informational	pairing process between sensor and phone	– no pairing process; communicates without authentication with app	– pairing through web page, then connecting to scale and providing wifi credentials; scale communication unencrypted	– no strict device pairing	– no strict bluetooth pairing	– no real pairing process	– no real pairing	– pairing using bluetooth and afterwards scale can use wifi or bluetooth	– pairing using bluetooth and afterwards scale can use wifi or bluetooth
av4_swfw_update	vulnerable: 1 / not vulnerable: 0	update cycle related issues	1 – no trace in app to server communication	0 – firmware downloaded by scale directly	1 – no information shown in app traffic	1 – no version information exchanged between app and server	1 – no software or firmware update cycle	1 – scale sends its version on registration to the server but doesn't expect any updates	0 – scale requests firmware from server	0 – scale requests firmware from server
av4_data_coll	informational	characteristics of data collected by the sensor	– synchronised through mobile app	– weight and body fat are collected; data assigned to user	– weight is collected and uploaded; as far as seen no other data is added	– with some delay data is synchronised to the server	– possible as long as token valid	– collecting weight information just as expected	– HTTP only; scale gets weather data	– various data including weather
number of issues			1.0	2.0	1.0	1.0	1.0	1.0	2.0	2.0

Table A.29: Experiment results – AV5 web application

test step	rating	desc	Activ8rLives Body Analyser com.activ8rlives.mobile	Fitbit Aria com.fitbit.FitbitMobile	HAPI Connected Scale com.hapiconnect	iChoice S1 com.medm.medmwt.diary com.medm.ichoice.diary	Thomson TBS705 com.stabxtom.thomson	Withings (all) com.withings.wiscale2
av5_reg_data_input	informational	Specifies the data fields and types which are required as inputs	– same as app	– mail, name, dob, weight (limited), height (limited)	– name, email, date of birth, goals	– same as app	n/a – n/a	– same as mobile app
av5_xss	vulnerable: 1 / not vulnerable: 0	basic xss attacks on the web page with a select statement	0 – not successful for password field and manual entries	0 – oauth and csrf token protect	0 – no	0 – not possible	n/a – n/a	0 – not possible
av5_pri_pol	no link or information: 1 / link available: 0	pirvacy poclicy linked and available	0 – the privacy policy is linked from the web page	0 – accessible from the page footer area where it is linked	0 – link in footer but not very visible	0 – linked on web page	n/a – n/a	0 – visibly linked from web interface
av5_pwd_change_freq	no period or frequency: 2 / either one: 1 / both: 0	password change period (must) and frequency (can)	2 – not implemented	2 – no frequency defined	2 – none	2 – none	n/a – n/a	2 – no frequency enforced
av5_pwd_change_policy	no policy: 2 / min characters: 1 / mixed charsets: 0	password policy enforced on password change	2 – none	1 – min 8 characters	1 – 6 characters	1 – min 6 characters	n/a – n/a	1 – min 6 char; any char no type enforcement
av5_reg_pwd_policy	no policy: 2 / min characters: 1 / mixed charsets: 0	is a password policy enforced	2 – no	1 – min 8 character	1 – min 6 characters	1 – min 6 characters	n/a – n/a	1 – min 6 char
av5_reg_acc_verif	none: 2 / email account verification: 1 / two factor authentication 0	email account verification or two factor authentication	2 – none	1 – account verification per email send but not enforced	2 – no	1 – no.. just email with confirmation that account has been created	n/a – n/a	2 – no verification

av5_sql_injection	vulnerable: 1 / not vulnerable: 0	inject sql queries to learn information from the server	0 – protects against basic sql injection	0 – not possible	0 – no	0 – not possible	n/a – n/a	0 – not possible
av5_reg_data_validation	data not validated: 1 / data is validated: 0	Comments on data validation during user registration: weight, heitght, email...	0 – same as app: only medical data	0 – sanitized and validated	1 – no	0 – same as app	n/a – n/a	0 – some data validation for height and weight
av5_data_input	data not validated: 1 / data is validated: 0	data input via web UI and validation	0 – data is limited to max 300kg	0 – weight data and other measurement data can be entered on web page	1 – data is not validated since unreasonably high measurements can be entered	1 – data is not validated but sanitized	n/a – n/a	0 – limited
av5_export_cloud	insecure feature: 1 / secure feature: 0	data being exported somewhere into the cloud (dropbox et al)	n/a – no export options	n/a – n/a	n/a – n/a	n/a – n/a	n/a – n/a	n/a – n/a
av5_export_disk	insecure feature: 1 / secure feature: 0	can data be exported to SD card or other locations on device and is that data encrypted	0 – excel export	0 – your data belongs to you - comment and download via htt	n/a – n/a	0 – download as csv or html page	n/a – n/a	0 – only after account deletion data can be downloaded
av5_csrf	vulnerable: 1 / not vulnerable: 0	basic CSRF attack	1 – csrf token not renewed	0 – ouath and csrf tokens protect	0 – tested on password change and found that websites protects against CSRF	0 – not vulnerable.. is using authenticity tokens for requests	n/a – n/a	0 – protected
av5_reg_pripol_link	no link or information: 1 / link available: 0	link to privacy policy or any kind of information on such topic	1 – not visible but very hidden in footer	0 – privacy policy has to be acknowledged; hint that data is sent to the US	1 – not visible but very hidden in footer	0 – visible in registration form	n/a – n/a	0 – yes
av5_export_social	insecure feature: 1 / secure feature: 0	export variants to social media and data encryption while exporting	n/a – only on platform itself; carer and caree invitations	0 – social sharing possible but with detailled privacy settings	0 – facebook, twitter	n/a – only within the same platform	n/a – n/a	n/a – no hint that data can be shared online

av5_data_wipe	insecure feature: 1 / secure feature: 0	wipe and delete data from the web account	1 – social engineered account deletion of third account via email	0 – per email request only	0 – no option	0 – account deletion possible	n/a – n/a	0 – 7 days account deletion period
av5_pwd_change_reuse	yes, unlimited: 0 / no, but history can be bypassed (see change freq): 1 / no without limitation: 2	can a password be reused and how much difference is required	2 – none	2 – unlimited reuse	2 – unlimited	2 – not limited	n/a – n/a	2 – unlimited reuse
number of issues			13.0	7.0	11.0	8.0	0.0	8.0

A.5 Post Experiment Analysis

A.5.1 Web Server SSL Check

Table A.30: Web server SSL check for com.activ8rlives.mobile

package	rating
xtra3.gpsonextra.net	52.84.212.13 T (trust issues)
xtra2.gpsonextra.net	analysis failed
connectivitycheck.gstatic.com	2607:f8b0:4005:801:0:0:0:200e:A, 216.58.192.14:A
xaapi.xamarin.com	54.164.34.41:A, 52.20.116.4:A
android.clients.google.com	216.58.192.14:A
api.activ8rlives.com	83.151.211.45:F

Table A.32: Web server SSL check for com.fitbit.FitbitMobile

package	rating
clients3.google.com	2607:f8b0:4005:801:0:0:0:200e:A, 216.58.192.14:A
graph.facebook.com	31.13.70.1:B, 2a03:2880:f022:6:face:b00c:0:2:B
www.fitbit.com	104.16.65.50:A, 104.16.66.50:A
m.facebook.com	2a03:2880:f10d:83:face:b00c:0:25de:B, 31.13.70.36:B
static.xx.fbcdn.net	2a03:2880:f00d:8:face:b00c:0:1:B, 31.13.70.7:B
decide.mixpanel.com	169.54.129.38:B, 169.54.129.6:B, 169.54.129.2:B, 169.54.129.13:B, 169.54.129.12:B, 169.54.129.14:B, 169.54.129.9:B, 169.54.129.8:B, 169.54.129.11:B, 169.54.129.21:B, 169.54.129.19:B, 169.54.129.29:B,
api.mixpanel.com	169.54.129.38:B, 169.54.129.7:B, 169.54.129.6:B, 169.54.129.36:B, 169.54.129.35:B, 169.54.129.2:B, 169.54.129.32:B, 169.54.129.9:B, 169.54.129.10:B, 169.54.129.40:B, 169.54.129.17:B, 169.54.129.28:B,
static0.fitbit.com	151.101.52.67:A
android-cdn-api.fitbit.com	104.16.65.50:A, 104.16.66.50:A
iedc.fitbit.com	151.101.52.67:A
32.scale.www.fitbit.com	104.16.65.50:T, 104.16.66.50:T
scontent.xx.fbcdn.net	2a03:2880:f00d:8:face:b00c:0:1:B, 31.13.70.7:B

Table A.34: Web server SSL check for com.hapiconnect

package	rating
hapi.lhealthcenter.com	analysis failed

Table A.36: Web server SSL check for com.medm.ichoice.diary

package	rating
go.ichoicelife.com	168.61.152.21:A+

Table A.38: Web server SSL check for com.medm.medmwt.diary

package	rating
health.medm.com	104.208.31.165:F

Table A.40: Web server SSL check for com.stabxtom.thomson

package	rating
app.thomson-hc.eu	analysis failed
alog.umeng.com	analysis failed

Table A.42: Web server SSL check for com.withings.wiscale2

package	rating
static.withings.com	46.105.202.21:A-
withings.com	89.30.121.170:A
scalews.withings.net	89.30.121.150:A
www.withings.com	89.30.121.170:A

A.5.2 Traffic Maps

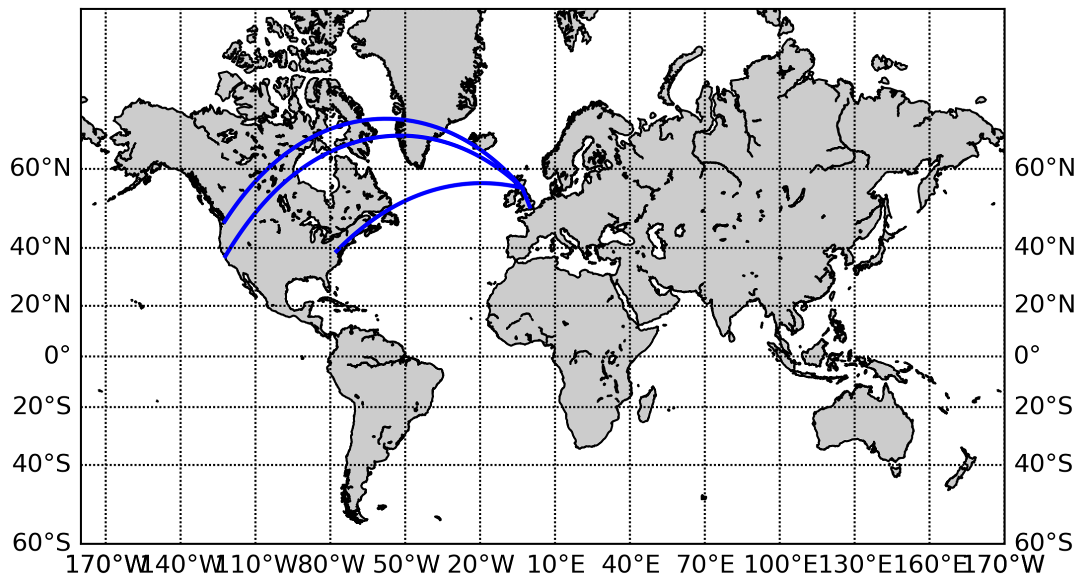


Figure A.1: Activ8rlives traffic map

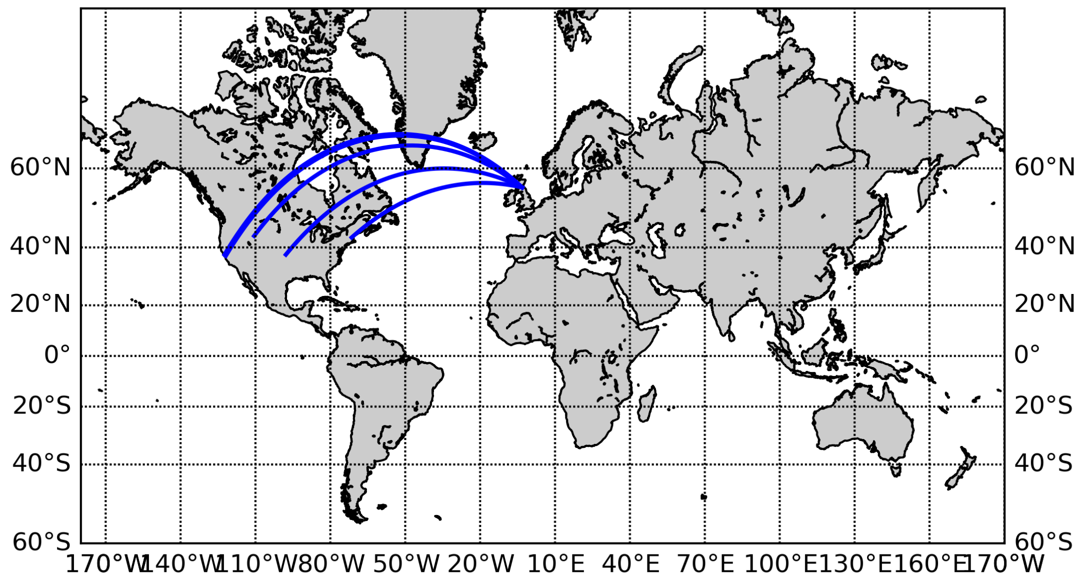


Figure A.2: Fitbit traffic map

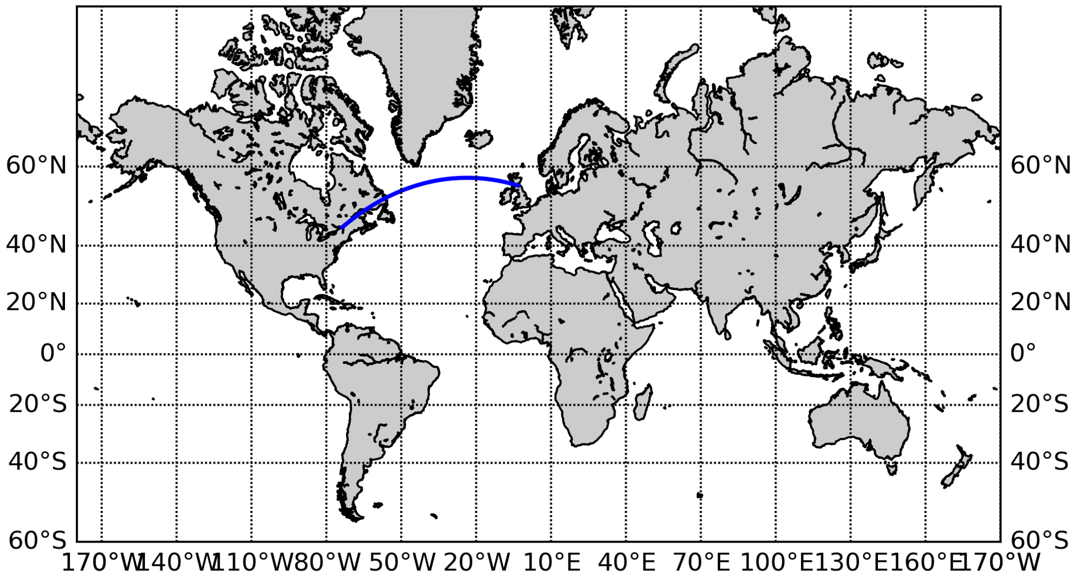


Figure A.3: HAPI traffic map

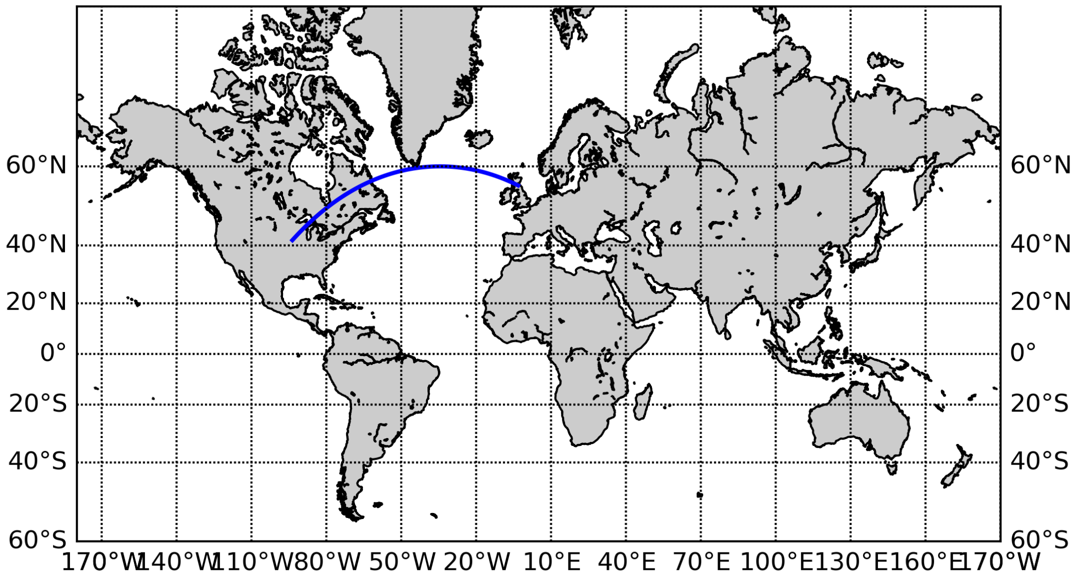


Figure A.4: iChoice traffic map

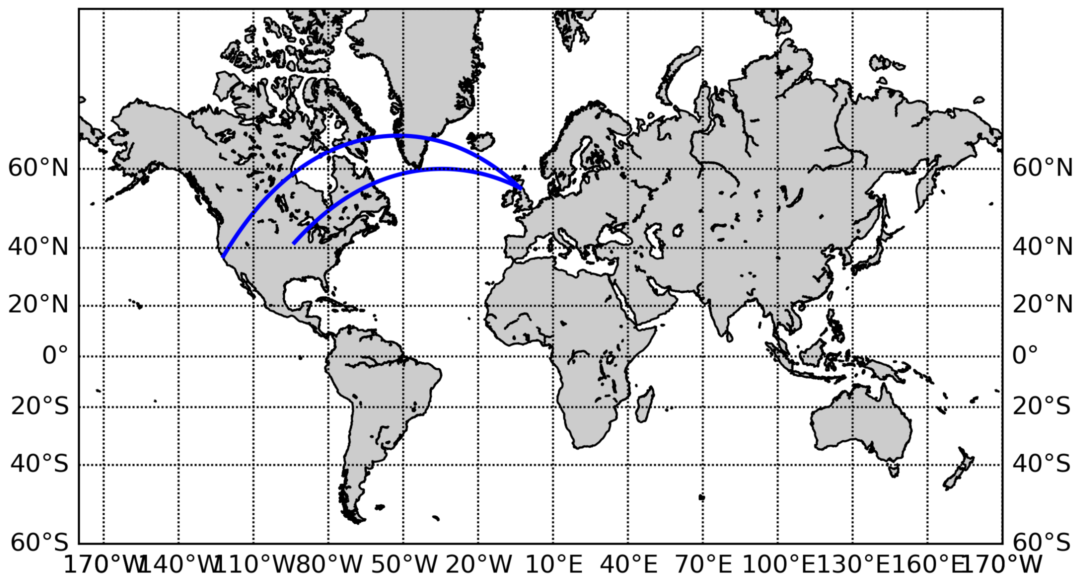


Figure A.5: iChoice MedM traffic map

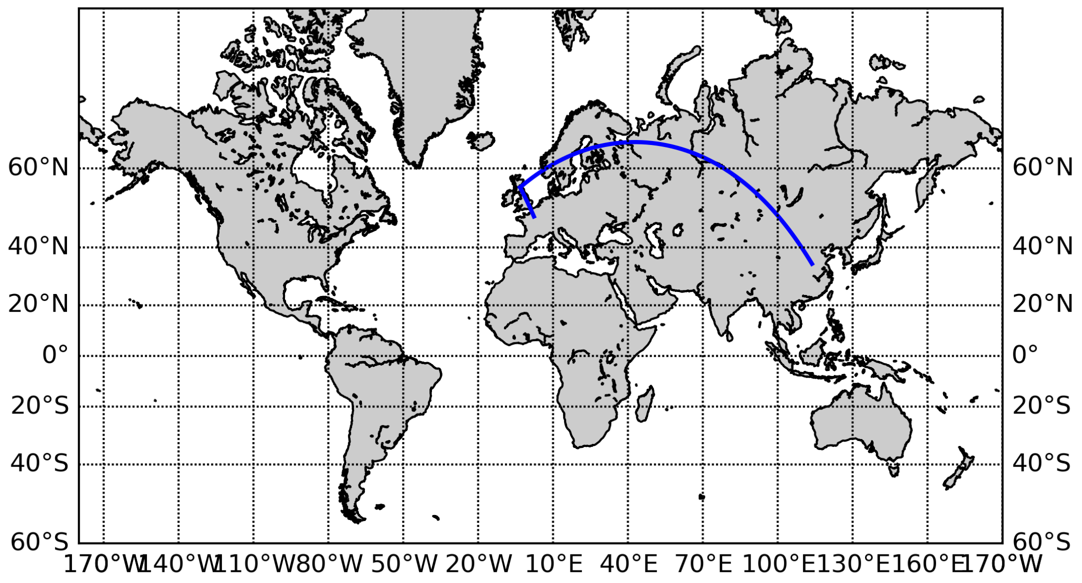


Figure A.6: Thomson traffic map

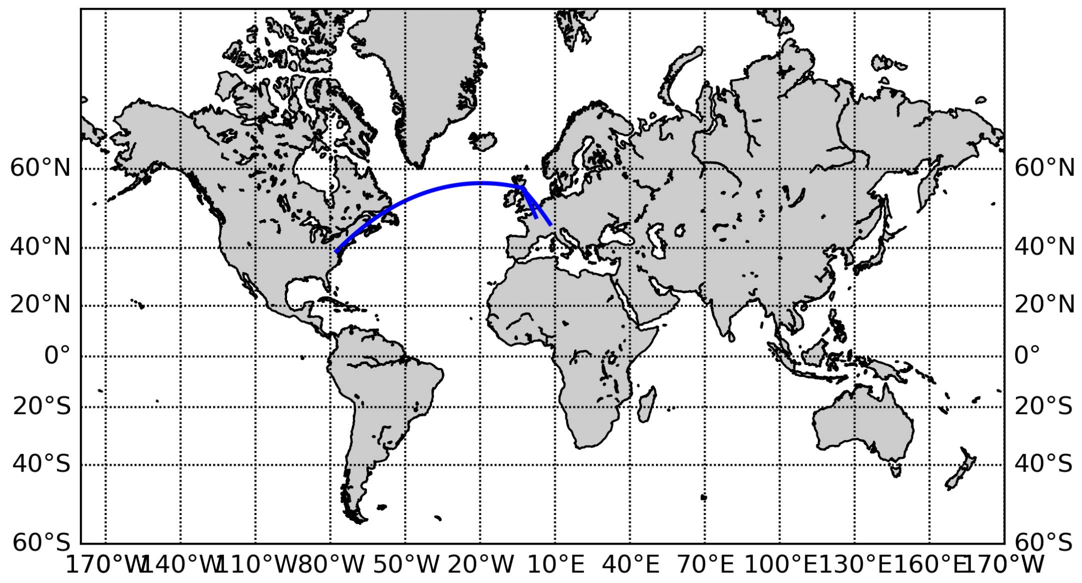


Figure A.7: Withings traffic map

Bibliography

[1] Kip Webb et al. *Accenture consumer survey on patient engagement*. Tech. rep. 2016. URL: https://www.accenture.com/_acnmedia/Accenture/Conversion-Assets/DotCom/Documents/Local/en-gb/PDF/Accenture-Patient-Engagement-Research-Recap-England.pdf#zoom=50.

[2] HIMSS. *2015 HIMSS Mobile Technology Survey*. Tech. rep. April. Healthcare Information and Management Systems Society, 2015.

[3] Susannah Fox and Maeve Duggan. “Tracking for Health”. In: *Pew Internet* October (2013), pp. 1–40.

[4] Peggy J. Mancuso et al. “Can patient use of daily activity monitors change nurse practitioner practice?” In: *Journal for Nurse Practitioners* 10.10 (2014), pp. 787–793.

[5] Parmy Olson. *Fitbit Data Now Being Used In The Courtroom*. 2014. URL: http://www.forbes.com/sites/parmyolson/2014/11/16/fitbit-data-court-room-personal-injury-claim/#71a82d4209f8 (visited on 06/15/2016).

[6] Claus Peter H Ernst. “Risk hurts fun: The influence of perceived privacy risk on social network site usage”. In: *Factors Driving Social Network Site Usage* (2015), pp. 45–56.

[7] Florian Rheingans, Burhan Cikit, and Claus-Peter H. Ernst. “The Potential Influence of Privacy Risk on Activity Tracker Usage: A Study”. In: *The Drivers of Wearable Device Usage*. Springer International Publishing, 2016, pp. 25–35.

[8] Mahmudur Rahman, Bogdan Carbunar, and Madhusudan Banik. “Fit and vulnerable: Attacks and defenses for a health monitoring device”. In: *Privacy Enhancing Technologies Symposium* (2013).

[9] Mahmudur Rahman, Bogdan Carbunar, and Umut Topkara. "SensCrypt: A secure protocol for managing low power fitness trackers". In: *Proceedings - International Conference on Network Protocols, ICNP* (2014), pp. 191–196.

[10] Britt Cyr et al. *Security Analysis of Wearable Fitness Devices (Fitbit)*. Tech. rep. Massachusetts Institute of Technology, 2014, pp. 1–14.

[11] Michael Coppola. *Hacking the Withings WS-30*. 2013. URL: `https://poppopret.org/2013/06/10/summercon-2013-hacking-the-withings-ws-30/` (visited on 06/10/2016).

[12] AFP. *Jawbone Server Hacked, Customer Data Accessed by Attackers*. 2013. URL: `http://www.securityweek.com/jawbone-server-hacked-customer-data-accessed-attackers` (visited on 06/17/2016).

[13] Information is beautiful. *World's Biggest Data Breaches*. 2016. URL: `http://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/` (visited on 06/17/2016).

[14] Konstantin Knorr and David Aspinall. "Security testing for Android mHealth apps". In: *Software Testing, Verification and Validation Workshops (ICSTW), 2015 IEEE Eighth International Conference on*. IEEE. 2015, pp. 1–8.

[15] Thomas E. Carroll et al. "Realizing scientific methods for cyber security". In: *Proceedings of the 2012 Workshop on Learning from Authoritative Security Experiment Results - LASER '12* (2012), pp. 19–24.

[16] By Mark Saunders and Paul Tosey. "The Layers of Research Design". In: *Rapport: The Magazine for NLP Professionals* 14.4 (2012), pp. 58–59.

[17] Sean Peisert and Matt Bishop. "How to design computer security experiments". In: *IFIP International Federation for Information Processing* 237 (2007), pp. 141–148.

[18] Josiah Dykstra. *Essential cybersecurity science : build, test, and evaluate secure systems*. O'Reilly Media, p. 174.

[19] Konstantin Knorr, David Aspinall, and Maria Wolters. "On the privacy, security and safety of blood pressure and diabetes apps". In: *IFIP Advances in Information and Communication Technology*. Ed. by Hannes Federrath and Dieter Gollmann. Vol. 455. 2015, pp. 571–584.

[20] Craig Michael Lie Njie. *Technical Analysis of the Data Practices and Privacy Risks of 43 Popular Mobile Health and Fitness Applications*. Tech. rep. Privacy Rights Clearinghouse, 2013, pp. 1–31. URL: https://www.privacyrights.org/mobile-health-and-fitness-apps-what-are-privacy-risks.

[21] Dongjing He et al. "Security Concerns in Android mHealth Apps." In: *AMIA ... Annual Symposium proceedings / AMIA Symposium. AMIA Symposium* 2014 (2014), pp. 645–54.

[22] Rajindra Adhikari, Deborah Richards, and Karen Scott. "Security and Privacy Issues Related to the Use of Mobile Health Apps". In: *25th Australasian Conference on Information Systems (ACIS 2014)*. 2014.

[23] Mario Ballano Barcena, Candid Wueest, and Hon Lau. *How Safe is your Quantified Self?* Tech. rep. Symantec Corporation, 2014. URL: http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/how-safe-is-your-quantified-self.pdf.

[24] Alexander Mense et al. "Analyzing Privacy Risks of mHealth Applications". In: *Volume 221: Transforming Healthcare with the Internet of Things*. 2016, pp. 41–45.

[25] Kit Huckvale et al. "Unaddressed privacy risks in accredited health and wellness apps: a cross-sectional systematic assessment." In: *BMC medicine* 13.1 (2015), p. 214.

[26] Mirza Mansoor Baig, Hamid GholamHosseini, and Martin J Connolly. "Mobile healthcare applications: system design review, critical issues and challenges." In: *Australasian physical & engineering sciences in medicine* 38.1 (2015), pp. 23–38.

[27] Rohit Goyal, Nicola Dragoni, and Angelo Spognardi. "Mind The Tracker You Wear - A Security Analysis of Wearable Health Trackers". In: *SAC '16*

Proceedings of the 31st Annual ACM Symposium on Applied Computing. 2016, pp. 131–136.

[28] Fitbit. *Fitbit flex*. 2016. URL: `http://www.fitbit.com/uk/flex` (visited on 06/20/2016).

[29] Wei Zhou and Selwyn Piramuthu. "Security/privacy of wearable fitness tracking IoT devices". In: *Iberian Conference on Information Systems and Technologies, CISTI* (2014).

[30] Dipl-Inform Eric Clausing et al. *Security Evaluation of nine Fitness Trackers*. Tech. rep. AV Test, 2015.

[31] Daniel Halperin et al. "Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses". In: *Proceedings - IEEE Symposium on Security and Privacy* (2008), pp. 129–142.

[32] Chunxiao Li, Anand Raghunathan, and Niraj K. Jha. "Hijacking an insulin pump: Security attacks and defenses for a diabetes therapy system". In: *2011 IEEE 13th International Conference on e-Health Networking, Applications and Services, HEALTHCOM 2011* (2011), pp. 150–156.

[33] Jakob Rieck. "Attacks on Fitness Trackers Revisited: A Case-Study of Unfit Firmware Security". In: *CoRR* 1604.03313 (2016), pp. 33–44.

[34] Andrew Hilts, Christopher Parsons, and Jeffrey Knockel. *Every Step You Fake*. Tech. rep. Open Effect Report, 2016.

[35] Misha Kay. "mHealth: New Horizons for Health through Mobile Technologies". In: *World Health Organization* 3 (2011), pp. 66–71.

[36] David Kotz. "A threat taxonomy for mHealth privacy". In: *2011 3rd International Conference on Communication Systems and Networks, COMSNETS 2011* (2011).

[37] Murray Aitken. "Patient Apps for Improved Healthcare From Novelty to Mainstream". In: *IMS Institute for Healthcare Informatics* October (2013), p. 61.

[38] Withings. *Wireless Scale (WS-30) – Withings*. URL: `https://withings.zendesk.com/hc/en-us/categories/200118117-Wireless-Scale-WS-30-`.

[39] IChoice. *Body Analyser Bluetooth*. (Visited on 06/21/2016).

[40] Activ8rlives. *Blood Pressure Bluetooth Monitor*. URL: `http://www.activ8rlives.com/devices/blood-pressure-monitor.html`.

[41] IHealth. *iHealth Wireless Pulse Oximeter | Health and Fitness Devices*. URL: `https://ihealthlabs.com/fitness-devices/wireless-pulse-oximeter/`.

[42] Medisana. *Medisana ® - made for Life | ThermoDock® Infrared Thermometer Module*. URL: `http://www.medisana.com/en/Health-control/Thermometer/ThermoDock-Infrared-Thermometer-Module.html`.

[43] T Scott Saponas et al. *Devices That Tell On You: The Nike+iPod Sport Kit*. Tech. rep. Department of Computer Science and Engineering, University of Washington, 2006. URL: `http://www.cs.washington.edu/research/systems/privacy.html`.

[44] Simon P Cohn. "Recommendations on Privacy and Confidentiality". In: *National Commitee on Vital and Health Statistics(NCVHS)* (2006), pp. 1–48.

[45] Daniel F. Schulke. "THE REGULATORY ARMS RACE: MOBILE-HEALTH APPLICATIONS AND AGENCY POSTURING". In: *BUL Rev.* 93 (2014), p. 1699.

[46] Jacob Betzner. *Hacking Into Your Fitbit | Valley News*. 2015. URL: `http://www.vnews.com/news/business/18063934-95/hacking-into-your-fitbit` (visited on 03/09/2016).

[47] Kota Tsubouchi, Ryoma Kawajiri, and Masamichi Shimosaka. "Working-relationship detection from fitbit sensor data". In: *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication - UbiComp '13 Adjunct* (2013), pp. 115–118.

[48] Greig Paul and James Irvine. "Privacy Implications of Wearable Health Devices". In: *SIN '14 Proceedings of the 7th International Conference on Security of Information and Networks* (2014), p. 117.

[49] Kate Green. *Self Surveillance*. 2008. URL: `https://www.technologyreview.com/s/410806/self-surveillance/` (visited on 08/16/2016).

[50] Eleonora Borgia. "The internet of things vision: Key features, applications and open issues". In: *Computer Communications* 54 (2014), pp. 1–31.

[51] Bluetooth Special Interest Group. *Bluetooth core specification*. URL: `https://www.bluetooth.com/specifications/bluetooth-core-specification` (visited on 06/23/2016).

[52] Carles Gomez, Joaquim Oller, and Josep Paradells. "Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology". In: *Sensors (Switzerland)* 12.9 (2012), pp. 11734–11753.

[53] IEEE. *IEEE-SA -IEEE Get 802 Program - 802.11: Wireless LANs*. URL: `http://standards.ieee.org/about/get/802/802.11.html` (visited on 06/23/2016).

[54] Dynastream Innovations Inc. *ANT+ Basics - THIS IS ANT*. URL: `https://www.thisisant.com/developer/ant-plus/ant-plus-basics` (visited on 06/23/2016).

[55] Moshaddique Al Ameen, Jingwei Liu, and Kyungsup Kwak. "Security and privacy issues in wireless sensor networks for healthcare applications". In: *Journal of Medical Systems* 36.1 (2012), pp. 93–101.

[56] Pardeep Kumar and Hoon-Jae Lee. "Security Issues in Healthcare Applications Using Wireless Medical Sensor Networks: A Survey". In: *Sensors* 12.12 (2011), pp. 55–91.

[57] Mengmeng Ge and Dong Seong Kim. "A Framework for Modeling and Assessing Security of the Internet of Things". In: *2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS)* (2015), pp. 776–781.

[58] Thaier Hayajneh et al. *A survey of wireless technologies coexistence in WBAN: analysis and open research issues*. Vol. 20. 8. 2014, pp. 2165–2199.

[59] Nadeem Qaisar Mehmood and Rosario Culmone. "An ANT+ Protocol Based Health Care System". In: *2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops* (2015), pp. 193–198.

[60] OWASP. *OWASP Zed Attack Proxy Project - OWASP*. URL: `https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project` (visited on 03/25/2016).

[61] Sufatrio et al. "Securing Android : A Survey , Taxonomy , and Challenges". In: *ACM Computing Surveys (CSUR)* 47.4 (2015), pp. 1–45.

[62] Aditya. Gupta and Elad. Shapira. *Learning pentesting for Android devices a practical guide to learning penetration testing for Android devices and applications*. Packt Pub, 2014.

[63] Joshua J. Drake et al. *Android Hacker's Handbook*. John Wiley & Sons, 2014, p. 576.

[64] Anmol. Misra. *Android security attacks and defenses*. CRC Press, 2013, p. 255.

[65] Dongkwan Kim et al. *BurnFit : Analyzing and Exploiting Wearable Devices*. Tech. rep. 2015, pp. 227–239.

[66] Elvis Collado. *Reversing and Exploiting Embedded Devices*. 2016. URL: `https://www.praetorian.com/blog/reversing-and-exploiting-embedded-devices-part-1-the-software-stack` (visited on 07/20/2016).

[67] Craig. *Reverse Engineering a D-Link Backdoor – /dev/ttyS0*. URL: `http://www.devttys0.com/2013/10/reverse-engineering-a-d-link-backdoor/` (visited on 03/25/2016).

[68] Tim Dierks and E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.2*. 2008. URL: `https://tools.ietf.org/html/rfc5246#page-33` (visited on 01/07/2016).

[69] Martin Georgiev, Suman Jana, and Dan Boneh. "The Most Dangerous Code in the World : Validating SSL Certificates in Non-Browser Software". In: *ACM CCS '12*. 2012, pp. 38–49.

[70] Sascha Fahl et al. "Why Eve and Mallory Love Android : An Analysis of Android SSL (In) Security Categories and Subject Descriptors". In: *Proceedings of the 2012 ACM conference on Computer and communications security* (2012), pp. 50–61.

[71] Oliver Grubin. "Encryption is Hard : Android ' s TLS Misadventures". Master Thesis. Imperial College London, 2015.

[72] Will Dormann. *Vulnerability Note VU#582497 - Multiple Android applications fail to properly validate SSL certificates*. 2014. URL: `https://www.kb.cert.org/vuls/id/582497` (visited on 01/07/2016).

[73] Mike Ryan. "Bluetooth: With Low Energy Comes Low Security". In: *Proceedings of the 7th USENIX Conference on Offensive Technologies* (2013), p. 4.

[74] Nitesh Dhanjani. *Abusing the Internet of Things*. O'Reilly Media, 2015.

[75] John Paol Dunning. "Taming the blue beast: A survey of bluetooth based threats". In: *IEEE Security and Privacy* 8.2 (2010), pp. 20–27.

[76] Charles P. Pfleeger, Shari Lawrence Pfleeger, and Johnathan Margulies. *Security in Computing*. Vol. 2nd. 1. 2006, p. 880.

[77] Codenomicon. *The Heartbleed Bug*. 2014. URL: `http://heartbleed.com` (visited on 08/15/2016).

[78] Zakir Durumeric et al. *Tracking the FREAK Attack*. 2015. URL: `https://freakattack.com/` (visited on 12/10/2015).

[79] Martin Georgiev et al. "The most dangerous code in the world". In: *Proceedings of the 2012 ACM conference on Computer and communications security - CCS '12*. New York, New York, USA: ACM Press, 2012, p. 38. URL: `http://dl.acm.org/citation.cfm?id=2382196.2382204`.

[80] Sourya Joyee De and Daniel Le M. *PRIAM : A Privacy Risk Analysis Methodology*. Tech. rep. Inria - Research Centre Grenoble – Rh{\^o}ne-Alpes, 2016.

[81] H.P Gassmann. "OECD guidelines governing the protection of privacy and transborder flows of personal data". In: *Computer Networks (1976)* 5.2 (1981), pp. 127–141.

[82] Jerome Radcliffe. “Hacking medical devices for fun and insulin: Breaking the human SCADA system”. In: *Black Hat Conference presentation slides* (2011), p. 13.

[83] Barnaby Jack. *IOActive Labs Research: "Broken Hearts": How plausible was the Homeland pacemaker hack?* URL: `http://blog.ioactive.com/2013/02/broken-hearts-how-plausible-was.html` (visited on 03/22/2016).

[84] Dieter Gollmann. *Computer Security*. 3rd Revise. John Wiley & Sons, 2011.

[85] Sasikanth Avancha, Amit Baxi, and David Kotz. “Privacy in mobile technology for personal healthcare”. In: *ACM Computing Surveys (CSUR)* 45.1 (2012), pp. 1–56.

[86] Bari Faudree and Mark Ford. “Security and Privacy in Mobile Health - Deloitte CIO - WSJ”. In: *CIO Journal* (2013).

[87] Linda Ackerman. *Mobile Health and Fitness Applications and Information Privacy Report to California Consumer Protection Foundation*. Tech. rep. 2013, pp. 1–26.

[88] *OWASP*. URL: `https://www.owasp.org/index.php/Main_Page` (visited on 06/25/2016).

[89] European Commission. *Revisions of Medical Device Directives*. 2015. URL: `https://ec.europa.eu/growth/sectors/medical-devices/regulatory-framework/revision/index_en.htmhttp://ec.europa.eu/growth/sectors/medical-devices/regulatory-framework/revision/index_en.htm` (visited on 06/25/2016).

[90] FDA. “Mobile Medical Applications”. In: *U.S. Deapartment of Health and Human Services Food and Drug Administration* (2015), p. 44.

[91] Dan Sung. *Wearable tech and regulation: Should fitness trackers face the FDA?* 2015. URL: `http://www.wareable.com/health-and-wellbeing/wearable-tech-and-regulation-5678` (visited on 06/10/2016).

[92] mHealth Alliance. “Patient Privacy in a Mobile World: A Framework to Address Privacy Law Issues in Mobile Health”. In: *Thomson Reuters Publication* June (2013), pp. 1–116.

[93] Anne Marie Helm and Daniel Georgatos. “Privacy and mHealth: How Mobile Health ’Apps’ Fit into a Privacy Framework Not Limited to HIPAA”. In: *Syracuse Law Review* 64 (2014).

[94] B Y Johannes Sametinger et al. “Security Challenges for Medical Devices”. In: *Communications of the ACM* 58.4 (2015), pp. 72–82.

[95] Gabrielle Addonizio. “The privacy risks surrounding consumer health and fitness apps, associated wearable devices, and HIPAA’s limitations”. In: *Law School Student Scholarship* Paper 827 (2016).

[96] MIT Technology Review. *Mobile Medical Apps: A Market on the Move*. 2014. URL: `http://www.technologyreview.com/view/532661/mobile-medical-apps-a-market-on-the-move/`.

[97] European Comission. *Commission Staff Working Document on the existing EU legal framework applicable to lifestyle and wellbeing apps*. 2014. URL: `http://ec.europa.eu/digital-agenda/en/news/commission-staff-working-document-existing-eu-legal-framework-applicable-lifestyle-and`.

[98] Food Administration and Drug. “General Wellness : Policy for Low Risk Devices Draft Guidance for Industry Staff”. In: *U.S. Department of Health and Human Services Food* (2015), pp. 1–11.

[99] European Comission. *European Commission - PRESS RELEASES - Press release - Restoring trust in transatlantic data flows through strong safeguards: European Commission presents EU-U.S. Privacy Shield*. 2016. URL: `http://europa.eu/rapid/press-release_IP-16-433_en.htm` (visited on 06/26/2016).

[100] Michelle M. Christovich. “Why Should We Care What Fitbit Shares - A Proposed Statutroy Solution to Protect Sensative Personal Fitness Information”. In: *Hastings Communication and Entertainment Law Journal* 38.1 (2016), pp. 91–116.

[101] Casey Erdmier, Jason Hatcher, and Michael Lee. “Wearable device implications in the healthcare industry”. In: *Journal of Medical Engineering & Technology* 40.4 (2016), pp. 141–148.

[102] Ken Kawamoto, Takeshi Tanaka, and Hiroyuki Kuriyama. "Your activity tracker knows when you quit smoking". In: *Proceedings of the 2014 ACM International Symposium on Wearable Computers - ISWC '14* (2014), pp. 107–110.

[103] Ali Sunyaev et al. "Availability and quality of mobile health app privacy policies." In: *Journal of the American Medical Informatics Association : JAMIA* December 2013 (2014), amiajnl–2013–002605.

[104] Michael Felderer et al. "Security Testing: A Survey". In: *Advances in Computers* (2015), pp. 1–43.

[105] Offensive Security. *Our Most Advanced Penetration Testing Distribution, Ever.* 2015. (Visited on 07/26/2016).

[106] Aldo Cortesi. *mitmproxy - home*. URL: `https://mitmproxy.org/` (visited on 03/23/2016).

[107] WireShark. *WireShark*. URL: `https://www.wireshark.org` (visited on 06/14/2016).

[108] Software Freedom Conservancy. *phpMyAdmin*. URL: `https://www.phpmyadmin.net/` (visited on 07/26/2016).

[109] MWR Labs. *Drozer*. URL: `https://labs.mwrinfosecurity.com/tools/drozer` (visited on 07/27/2016).

[110] Anthony Desnos and Geoffroy Gueguen. *androguard/androguard: Reverse engineering, Malware and goodware analysis of Android applications ... and more (ninja !)* 2012. URL: `https://github.com/androguard/androguard` (visited on 07/27/2016).

[111] Adrienne Porter Felt et al. "A survey of mobile malware in the wild". In: *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices - SPSM '11* (2011), pp. 3 –14.

[112] Michael C Grace et al. "Unsafe exposure analysis of mobile in-app advertisements". In: *Proc. 5th ACM conference on Security and Privacy in Wireless and Mobile Networks* 067.Section 2 (2012), pp. 101–112.

[113] Kevin Allix et al. “A forensic analysis of android malware - How is malware written and how it could be detected?” In: *Proceedings - International Computer Software and Applications Conference* (2014), pp. 384–393.

[114] Michael Grace et al.
“RiskRanker: Scalable and Accurate Zero-day Android Malware Detection”.
In: *10th International Conference on Mobile Systems, Applications, and Services* (2012), pp. 281–294.

[115] Siegried Rasthofer et al. “Harvesting Runtime Data in Android Applications for Identifying Malware and Enhancing Code Analysis”. In: *Ndss* (2016).

[116] Denis Peretto and Peter de Kraker.
Addons Detector | inside the app – SDK analytics.
URL: `https://public.addonsdetector.com/` (visited on 07/28/2016).

[117] Mohamed Nassim Seghir and David Aspinall.
“EviCheck: Digital evidence for android”. In: *Automated Technology for Verification and Analysis, 13th International Symposium, ATVA 2015*.
Vol. 9364. Springer International Publishing, 2015, pp. 221–227.

[118] Wei Chen et al. “A Text-Mining Approach to Explain Unwanted Behaviours”.
In: *Proceedings of the 9th European Workshop on System Security*.
ACM, 2016, pp. 1–6.

[119] Steven Arzt et al. “Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps”.
In: *ACM SIGPLAN Notices* 49.6 (2014), pp. 259–269.

[120] Alessandro Reina, a Fattori, and L Cavallaro.
“A system call-centric analysis and stimulation technique to automatically reconstruct android malware behaviors”.
In: *ACM European Workshop on Systems Security (EuroSec).* (2013), pp. 1–6.

[121] Victor van der Veen, Herbert Bos, and Christian Rossow.
“Dynamic Analysis of Android Malware”.
Master Thesis. VU University Amsterdam, 2013.

[122] William Enck et al. “TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones”.
In: *ACM Transactions on Computer Systems (TOCS)* 32.2 (2014), p. 5.

[123] Vysor. *Vysor Screencast*. 2016.
URL: http://www.vysor.io/ (visited on 07/27/2016).

[124] RecordMyDesktop dev group. *About recordMyDesktop*.
URL: http://recordmydesktop.sourceforge.net/about.php (visited on 07/27/2016).

[125] *DroidWall - Android Firewall*. 2012.
URL: https://code.google.com/archive/p/droidwall/ (visited on 07/28/2016).

[126] John Hunter. *matplotlib: python plotting*.
URL: http://matplotlib.org/ (visited on 07/27/2016).

[127] Fei Liu et al. “A Step Towards Usable Privacy Policy: Unsupervised Alignment of Privacy Statements”. In: *Proceedings of The 25th International Conference on Computational Linguistics (COLING 2014)*.
Dublin, Ireland, 2014, pp. 884–894.
URL: http://www.coling-2014.org/accepted-papers/585.php.

[128] European Comission.
Factsheet on the "Right to be Forgotten" ruling (c-131/12). 2014.
URL: http://ec.europa.eu/justice/data-protection/files/factsheets/factsheet_data_protection_en.pdf.

[129] Lucky Onwuzurike and Emiliano De Cristofaro. “Danger is My Middle Name: Experimenting with SSL Vulnerabilities in Android Apps”.
In: *CoRR* (2015).

[130] Marten Oltrogge et al. “To Pin or Not to Pin — Helping App Developers Bullet Proof Their TLS Connections”. In: *Proceedings of the 24th USENIX Conference on Security Symposium* (2015).

[131] Mauro Conti, Nicola Dragoni, and Sebastiano Gottardo.
“MITHYS: Mind The Hand You Shake - Protecting mobile devices from SSL usage vulnerabilities”. In: *Security and Trust Management* 8203 (2013).

[132] Hui Liu et al. “TagDroid: Hybrid SSL Certificate Verification in Android”.
In: *Information and Communications Security*.
Springer International Publishing, 2015, pp 120–131.

[133] OWASP. *Certificate and Public Key Pinning*. 2014. URL: http://www.netcraft.com/anti-phishing/phishing-site-feed/https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning (visited on 01/08/2016).

[134] Yimin Song, Chao Yang, and Guofei Gu. “Who is peeping at your passwords at Starbucks? — To catch an evil twin access point”. English. In: *2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN)*. IEEE, 2010, pp. 323–332. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5544302.

[135] Nigel P Smart, Kenny Paterson, and Ronald Cramer. *Cryptography Made Simple*. 2015, pp. 403–416.

[136] Dinei Florencio and Cormac Herley. “A large-scale study of web password habits”. In: *Proceedings of the 16th international conference on World Wide Web - WWW ’07* (2007), p. 657.

[137] Saranga Komanduri et al. “Of Passwords and People: Measuring the Effect of Password-Composition Policies”. In: *Proceedings of the 2011 annual conference on Human factors in computing systems - CHI ’11* (2011), p. 2595.

[138] Matt Weir et al. “Testing metrics for password creation policies by attacking large sets of revealed passwords”. In: *Proceedings of the 17th ACM conference on Computer and communications security - CCS ’10* (2010), p. 162.

[139] Eoin Keary. *Authentication Cheat Sheet*. 2012. URL: https://www.owasp.org/index.php/Authentication_Cheat_Sheet#Implement_Proper_Password_Strength_Controls (visited on 08/05/2016).

[140] Adrienne Porter Felt et al. “Android permissions demystified”. In: *Proceedings of the 18th ACM conference on Computer and communications security*. ACM. 2011, pp. 627–638.

[141] Shuang Liang and Xiaojiang Du. “Permission-combination-based scheme for Android mobile malware detection”. English. In: *2014 IEEE International Conference on Communications (ICC)*.

IEEE, 2014, pp. 2301–2306. URL: `http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=6883666`.

[142] Google. *Requesting Permissions at Run Time*. 2016. URL: `https://developer.android.com/training/permissions/requesting.html` (visited on 08/05/2016).

[143] Xamarin. *Xamarin*. 2016. URL: `https://www.xamarin.com` (visited on 08/05/2016).

[144] Bodo Möller, Thai Duong, and Krzystof Kotowicz. *This POODLE Bites: Exploiting The SSL 3.0 Fallback*. Tech. rep. Google, 2014, pp. 1–6.

[145] Fabian Mewes. *Fitbit Aria Wi-Fi Smart Scale*. URL: `https://github.com/micolous/helvetic` (visited on 08/06/2016).

[146] Michael Farrell. *Fitbit without fitbit.com*. 2014. URL: `https://github.com/micolous/helvetic/blob/master/protocol.md` (visited on 08/02/2016).

[147] David Lodge. *Are your smart weighing scales lying to you? Quite possibly*. 2015. URL: `https://www.pentestpartners.com/blog/are-your-smart-weighing-scales-lying-to-you-quite-possibly-part-1/` (visited on 08/06/2016).

[148] Ken Munro. *Extracting your WPA PSK from bathroom scales*. 2015. URL: `https://www.pentestpartners.com/blog/extracting-your-wpa-psk-from-bathroom-scales/` (visited on 08/02/2016).

[149] Herny S. Warren. *Hacker's Delight*. 2nd. Addison Wesley, 2012, p. 512.

[150] Project Rainbow Crack. *List of Rainbow Tables*. 2016. URL: `http://project-rainbowcrack.com/table.htm` (visited on 08/06/2016).

[151] Ryszard Wiśniewski and Tumbleson Connor. *Apktool - A tool for reverse engineering Android apk files*. URL: `https://ibotpeaches.github.io/Apktool/` (visited on 07/27/2016).

[152] Felipe Sierra and Anthony Ramirez. "Defending Your Android App". In: *RIIT '15: Proceedings of the 4th Annual ACM Conference on Research in Information Technology* (2015), pp. 29–34.

[153] Symantec. *Android.Umeng*. 2015. URL: `https://www.symantec.com/security_response/writeup.jsp?docid=2014-040307-5749-99` (visited on 08/06/2016).

[154] Mark Paul Kamichoff. *Withings Scale Hacking*. 2010. URL: `http://www.prolixium.com/mynews?id=915` (visited on 08/06/2016).

[155] W00t? *Hacking the Withings WiFi Body Scale*. URL: `https://blog.chris007.de/hacking-the-withings-wifi-body-scale-2/` (visited on 08/06/2016).

[156] Withings. *Privacy Policy*. 2015. URL: `https://www.withings.com/uk/en/legal/privacy` (visited on 08/07/2016).

[157] Swissmed Mobile. *MedM Account Privacy Statement*. 2014. URL: `https://health.medm.com/privacy` (visited on 08/07/2016).

[158] Daniel Brandt. *Uncovering bad guys hiding behind CloudFlare*. URL: `http://www.crimeflare.com/cfs.html` (visited on 08/07/2016).

[159] Information Commissioner's Office. *Data transfers to the US and Safe Harbor – interim guidance*. Tech. rep. October 2015. 2016.

[160] David LeBlanc and Michael Howard. *Writing Secure Code*. 2nd. Microsoft Press, 2004.

[161] CVSS Special Interest Group. *Common Vulnerability Scoring System, V3*. 2015. URL: `https://www.first.org/cvss` (visited on 08/09/2016).

[162] Hasan Faik Alan. "Can Android Applications Be Identified Using Only TCP / IP Headers of Their Launch Time Traffic?" In: *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. ACM, 2016, pp. 61–66.

[163] Martin Krämer, David Aspinall, and Maria Wolters. "Weighing in eHealth Security: A Security and Privacy Study of Smart Scales". In: *ACM CCS '16*. 2016. Forthcoming.

[164] Emilian Girault. *Unofficial GooglePlay-API*. 2012. URL: `https://github.com/egirault/googleplay-api/` (visited on 07/25/2016).

[165] Dex2jar. *dex2jar download | SourceForge.net*. URL: `https://sourceforge.net/projects/dex2jar/` (visited on 07/27/2016).