The Independent IT-Security Institute
Magdeburg Germany

Internet of Things

Security Evaluation of 7 Fitness Trackers on Android and the Apple Watch

Eric Clausing

Michael Schiefer

July 11, 2016

1 Introduction

As witnessed last year, the interest in and popularity of fitness trackers has further increased, and their successful momentum appears unabated for now. This prompted us to conduct further investigations on the security of fitness trackers, based on our first test in 2015 [Clausing et al., 2015]. For this new test, we use a whole new set of devices from different price segments and with different feature sets. This test set includes 7 trackers tested on Android and Apple's smartwatch tested on iOS. Furthermore, we refined our test methodology and setup to allow for a more detailed view on especially important aspects, such as online communication.

In this first chapter we will briefly introduce the topic, explain the motivation and summarize prior and similar work in this special field of testing. The second chapter addresses the test concept, setup and execution, followed by a presentation and explanation of the test results in chapter three. In chapter four, we briefly discuss concerns and implications for individual privacy when using such devices, even when the actual security concept of the device may be flawless. Finally, the last chapter concludes with a brief summary of the test, and points out approaches for further test and research activity in the specific field of fitness tracking devices.

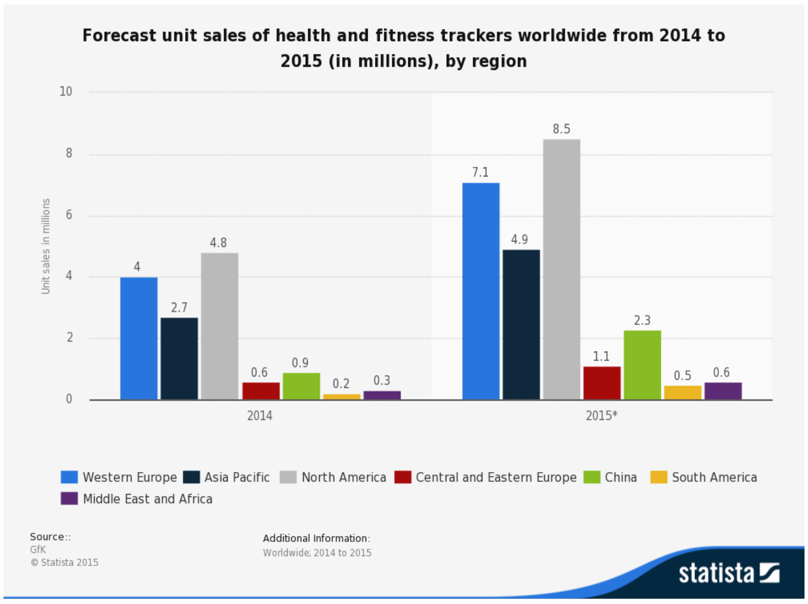


Figure 1.1: Worldwide sales forecast of health and fitness trackers from 2014 to 2015 (in millions) by region

1.1 Motivation and Prior Work

As seen in Figure 1.1, the importance of fitness trackers has been growing immensely from year to year, and sales numbers are increasing significantly. As we already mentioned in our first test, plans to use fitness trackers for actuarial purposes are now underway in the United States in the meantime, and several insurers are offering bonus programs involving vouchers and discounts to reward the use of fitness devices and access to the data collected. In Germany, the purchase of these devices is being subsidized by several insurance companies [Dörner, 2015, Vers, 2015], although insurers state that they are not (yet) interested in the collected data itself. For that purpose alone, fitness trackers ought to meet high security standards, but recent developments in the legal sector demand an even greater emphasis of tamper protection and robust authentication. Multiple news websites are reporting on an increasing number of cases in which data collected by fitness trackers is used as evidence in court [Fleischer, 2016, Gardner, 2016]. But as our tests demonstrate, users should maintain a certain level of skepticism towards manipulation protection and robust authentication, as many of the tested devices are a still a long way off from what ought to be admitted as evidence in court.

Our test concept and execution are based on our initial test [Clausing et al., 2015]. Some of the test criteria, however, were adjusted and/or refined in order to place greater focus on certain criteria and somewhat less focus on others.

2 Test Concept

In this chapter, we describe our test environment, including setup and execution. Furthermore, we present a list of all tested products and provide details on their feature set. The product applications are also listed, noting the specific software version examined.

2.1 Test Setup

The test setup for this second round of testing is slightly changed in comparison to the first test. As this test seeks to reveal potential for manipulation of data traffic with the server, the test setup involves an additional computer placed between the smartphone and Internet to act as a man-in-the-middle. Figure 2.1 illustrates the setup. When redirected to this computer, network traffic can be monitored and selectively manipulated. The man-in-the-middle attack is launched with the help of the tool *mitmproxy* [mitmproxy, 2014], particularly suited for this purpose, thanks to its configurability and convenient handling. *mitmproxy* is a Linux tool, allowing for penetration of encrypted HTTPS connections by means of a man-in-the-middle approach and enabling the text content of requests and their corresponding responses to be read and manipulated. We use it to check whether or not a user or attacker is theoretically capable of altering the sync data transmitted by the app or whether the corresponding server even notices the manipulation. For our tests, we use *mitmproxy* in transparent mode with a custom gateway, as shown in Figure 2.2.
Otherwise, the basic test setup is essentially the same for this second test. Naturally, the relevant tracker is still connected via Bluetooth to a smartphone (either with the original or test app installed), the Bluetooth traffic is monitored, the original app is analyzed and the online communication between the app and the server is tested for potential vulnerabilities, i.e. unencrypted or weak encrypted connections.

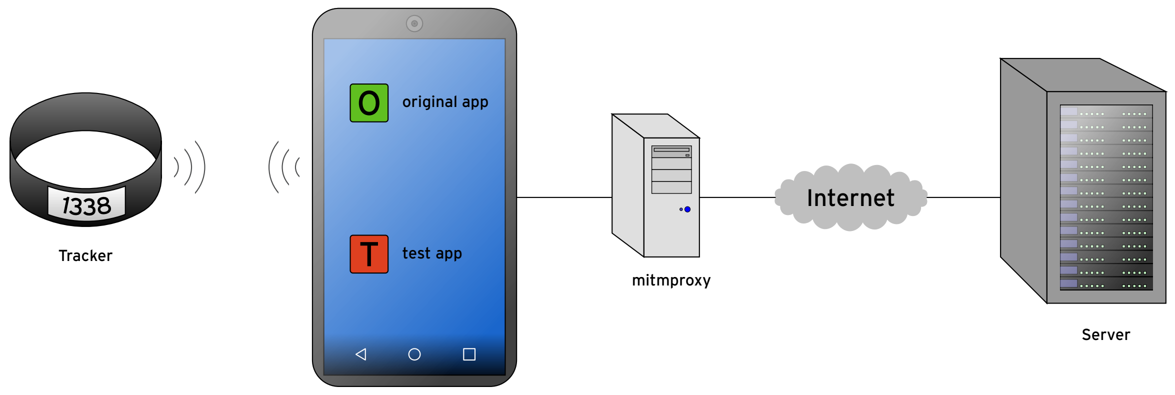
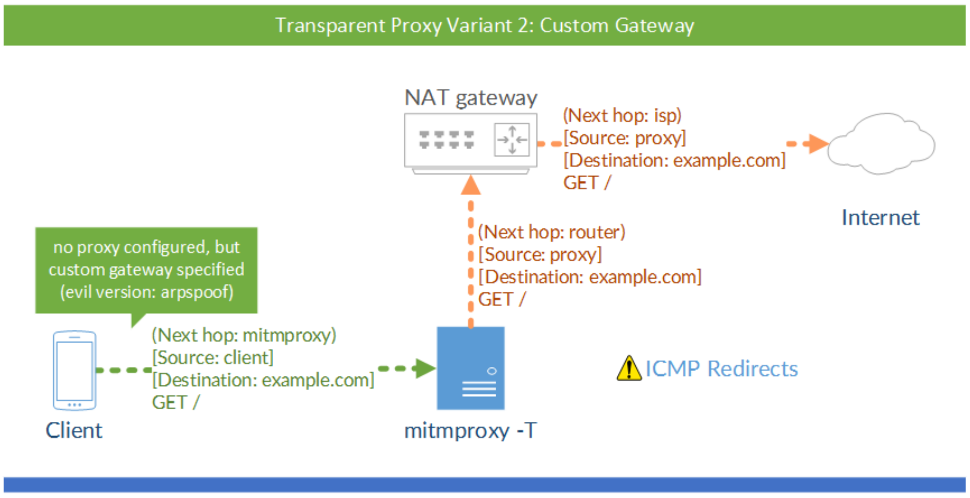


Figure 2.1: Schematic test setup



Transparent Proxy Variant 2: Custom Gateway
NAT gateway
(Next hop: isp)
[Source: proxy]
[Destination: example.com]
GET /
Internet
(Next hop: router)
[Source: proxy]
[Destination: example.com]
GET /
no proxy configured, but custom gateway specified (evil version: arpspoof)
(Next hop: mitmproxy)
[Source: client]
[Destination: example.com]
GET /
Client
mitmproxy -T
ICMP Redirects

Figure 2.2: ***mitmproxy* transparent mode with custom gateway as used in our tests (Source:** http://docs.mitmproxy.org/en/latest/_images/proxy-modes-transparent-2.png**)**

2.2 Test Execution

The test execution is divided into three steps, ultimately yielding the aggregate results for the test. These three steps are:

- Analysis of original app
- Analysis of Bluetooth communication between tracker and smartphone (with the original and/or test app installed)
- Analysis of online communication of original app

The first step involves analyzing the original app for potential vulnerabilities. In doing so, we check for overall security measures like code obfuscation and protection of sensitive user data stored by the app. In the second step, we observe and analyze the Bluetooth communication, i.e. we monitor the data traffic between the tracker and the app and eventually try to mimic the communication (by communication replay from a second smartphone, for example) to check for robust authentication and the possibility for an attacker to obtain access to tracker functions or stored data. New to this second round of testing is the inclusion of a "fake tracker" check. In Android 5.0, the Bluetooth functionality was expanded by adding the option of using an Android smartphone in Peripheral Mode, which theoretically allows for implementation of a fake tracker, i.e. a smartphone claiming to be a tracker. We check whether or not it is possible to implement a fake tracker capable of connecting to the original app. In the last of the three steps, we analyze the complete online communication to and from the original app. Therefore we first check whether the vital connections for user authentication, cloud synchronization and firmware updates are protected by HTTPS. Furthermore, for this second test round we additionally evaluate whether the content of these secure connections can be read and theoretically even manipulated.

The test for the Apple product is performed differently for some aspects. The results and more details on the test execution for the *Apple Watch* are separately discussed in section 3.4.

2.3 Product Details

In Tables 2.1 and 2.2 below, there is a listing of the tested product features, along with information on the testing of the corresponding app and app version. As one can see, a total of 8 products are tested, 7 on Android one on iOS. The specific products are chosen based on their popularity and prevalence. The tested app is the official version downloaded from Google Play store or delivered with iOS. As the tests were performed over a rather long period, not all tested versions may reflect the latest version at release of this document. All results refer to the analysis of the version listed in Table 2.2.

	Apple *Watch*	Basis *Peak*	Microsoft *Band 2*	Mobile Action *Q-Band*	Pebble *Time*	Runtastic *Moment Elite*	Striiv *Fusion*	Xiaomi *MiBand*
Bluetooth 4.0 (Low Energy)	✔	✔	✔	✔	✔	✔	✔	✔
Wi-Fi	✔	✘	✘	✘	✘	✘	✘	✘
Microphone	✔	✘	✔	✘	✔	✘	✘	✘
Display	✔	✔	✔	✔	✔	✘	✔	✘
Pulsemeter	✔	✔	✔	✘	✘	✘	✘	✘
Pedometer	✔	✔	✔	✔	✔	✔	✔	✔
Sleep Tracking	✘	✔	✔	✔	✔	✔	✔	✔
Integrated GPS	✘*	✘	✔	✘	✘	✘	✘	✘
Magnetometer	✘	✘	✘	✘	✔	✘	✘	✘
Barometer	✘	✘	✔	✘	✘	✘	✘	✘
UV-Sensor	✘	✘	✔	✔	✘	✘	✘	✘
Ambient Light Sensor	✔	✘	✔	✔	✔	✘	✘	✘
Capacitive/Galvanic Sensors	✘	✔	✔	✘	✘	✘	✘	✘
Skin Temperature	✘	✔	✔	✘	✘	✘	✘	✘

✔integrated/supported ✘not integrated/supported

* using GPS of *iPhone* where possible

Table 2.1: Listing of physical tracker features

App name	Version
Apple *Apple Watch*	
Watch	2.1
Basis *Peak*	
Basis Peak	1.17.1
Microsoft *Band 2*	
Microsoft Health	1.3.20213.1
Pebble *Time*	
Pebble Time	3.9.1-966-bc5f043
Runtastic *Moment Elite*	
Runtastic Me	1.5.3
SportPlus *Q-Band*	
i-gotU Life	1.2.1506.947
Striiv *Fusion*	
Striiv Activity Tracker	1.0.1024p
Xiaomi *MiBand*	
Mi Fit	1.8.441

Table 2.2: Listing of tested trackers and app versions

3 Test Results

Table 3.1 summarizes the results for the tested products in the three categories *Tracker*, *Application* and *Online Communication*. Each category and its items are explained in the following sections.

	Basis *Peak*	Microsoft *Band 2*	Mobile Action *Q-Band*	Pebble *Time*	Runtastic *Moment Elite*	Striiv *Fusion*	Xiaomi *MiBand*
Tracker							
Controlled Visibility	✘	✔	≈	✔	✘	✘	✘[1]
Bluetooth Smart LE Privacy	✘	✘	✘	✘	✘	✘	✘
Controlled Connectivity	✔	✔	✘	✔	✘	✘	✔
Adequate Authentication	✔	✔	✘	✔	✘	✘	≈
Tamper Protection	✔	✔	✘	✔	✘	✘	≈
Application							
No Unsecured Local Storage	✔	✔	✔	✔	✔	✔	✘
Code Obfuscation	≈	✘	✔	✔	≈	✘	✔
No Log/Debug Output	✘	✘	✔	✘	✘	✘	✘
Online Communication							
Encryption	✔	✔	≈	✔	✔	≈	≈
Tamper Protection	✔	≈	≈	✔	≈	≈	≈

✔Yes ≈Partially ✘No

[1]always invisible after pairing

Table 3.1: Summary of test results

3.1 Tracker

The *Tracker* category includes relevant aspects allowing for adequate evaluation of the tracker's overall security level. These aspects mostly concern visibility and traceability of the tracker by third-parties, protection against unauthorized access and protection from tampering by third-parties and users themselves.

Controlled Visibility

Active Bluetooth-LE devices are visible by default for every other Bluetooth device within range, as they actively advertise their presence and connection address. From a security perspective, this behavior results in two potential threats. First, it is a threat to users' privacy – a device that is always visible can itself easily become the target of unwanted tracking. Especially if the device uses a fixed physical MAC address as its connection address, it is child's play to easily derive movement profiles for a specific target. Second, a device that is always visible is far more vulnerable to attacks, as it can be easily located by potential attackers. Of course, it would be possible to protect the device by implementing other mechanisms that ensure that unwanted connections are refused, but it would be much simpler and safer to just be invisible to these connection attempts.
For our test, we checked for adequate measures for controlling the visibility of the device when there is no active Bluetooth connection to the user's registered smartphone. According to our findings, there are only four tested products that implement a visibility control mechanism. These include products from Microsoft, Pebble and Mobile Action, although the mechanism only works as intended for the first two out of the three. The *Q-Band* from Mobile Action provides the option of activating visibility by pushing a hardware button on top of the tracker, but in our tests, we were able to find the device multiple times without first activating its visibility. Furthermore, armed with the knowledge of its physical address, we were able to directly connect to it without having to scan for it or activate its visibility beforehand. Apart from these three devices, no other Android device in our test implements a visibility control mechanism. Consequently, all other devices are basically trackable whenever there is no active connection to the user's smartphone. Although that does not mean they are automatically vulnerable to attacks, as many of the devices implement other mechanisms to prevent unauthorized access (see 3.1), it is still a potential threat that could be eliminated with the help of rather simple measures.

Bluetooth LE Privacy

As mentioned in the previous section, the active Bluetooth-LE advertisement, when used in combination with the physical MAC address as the connection address, is a good opportunity for third parties to launch a targeted tracking on a certain device. As a countermeasure, a privacy feature was introduced with Android 5.0 which implements a periodic randomization of the connection address and additionally forbids the use of the physical MAC address as the connection address. This ought to prevent any tracking and the generation of movement profiles by logging and analyzing advertised connection addresses.
In our test, we checked the tested products as to whether they are already implementing this feature and thereby actively support user privacy against potential third-party tracking. According to our findings, only one of the tested products now supports this feature: Microsoft's *Band 2*. The missing implementation of this feature on all other devices is especially critical for those without a visibility control mechanism, as they can be identified and tracked for a considerable length of time with the help of their static and permanent physical connection address. The generation of movement profiles for these devices is therefore extremely easy.
Our tests also surprisingly showed, however, that there are practical advantages to not having the BLE privacy

features. With Android 5.0, the possibility of using an Android smartphone as a Bluetooth-LE peripheral is introduced. As a consequence, it is theoretically possible to mimic the behavior of a fitness tracker with a smartphone – the device name can be changed, the tracker services can be generated, the corresponding characteristics can be defined and all the logic connecting them can be re-implemented. By doing so, an attacker might be able to disguise itself as a fitness tracker and connect to a victim's smartphone, or a user could create his own version of his tracker that only outputs fitness data he has specified. However, in our tests we failed to implement such a fake tracker – and that is the surprise – because support of BLE privacy is missing on nearly all trackers (except for the Microsoft *Band 2*) and product apps. The Android API forbids the deactivation of this feature for the implementation of a BLE-Peripheral and additionally does not allow editing of the advertised connection address. Yet, as most of the tested apps in our test searched for a static MAC address or at least a static address scheme to identify genuine trackers, the fake tracker has no chance of being accepted as genuine. As a consequence, all products with missing support for the BLE Privacy feature are for now relatively safe from that kind of spoofing attack.

Controlled Connectivity

To further improve the protection of a tracker against random attacks, it makes perfect sense not to allow connection attempts from any random device but rather to exclude certain devices (or device classes) even before the actual authentication process takes place. The simplest way to achieve this is via pairing and bonding. These mechanisms allow devices to be introduced to one another, and based on the information exchanged during pairing and bonding, decide on a later connection query whether to communicate and accept the connection or otherwise refuse the connection.
For our test, we checked which products use pairing or some another adequate means of ensuring that no unauthorized devices are allowed to establish a connection to the tracker. As our tests demonstrated, 4 products out of our test set implement effective mechanisms to prevent unauthorized connection attempts to the tracker. The Basis *Peak* and Microsoft *Band 2* pursue a classic strategy and implement an exclusive bonding between a smartphone and the tracker. The user confirms the pairing upon initial connection, an exclusive bonding is created and afterwards only that connection between these two devices is seen as legitimate. All other connection attempts are refused. The Pebble *Time*, on the other hand, enables usage of multiple smartphones with one tracker, but demands a physical user confirmation for every new unknown device attempting to connect. The Xiaomi *MiBand* implements a rather simple but nevertheless quite effective way to tackle this problem and basically becomes invisible for all other devices (which basically means it stops advertising its presence) once it is paired and bonded to a smartphone.

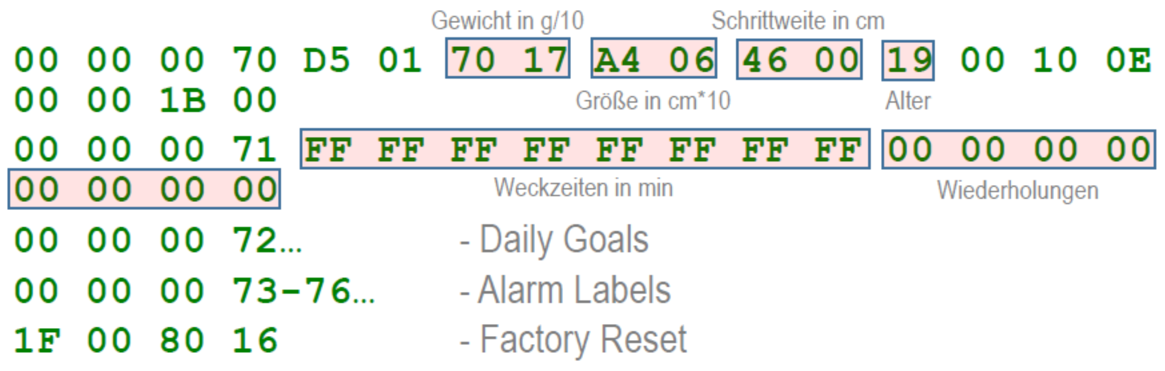
Adequate Authentication

For a security sensitive scenario, it is always recommended to use a robust authentication method of some kind to ensure that access to certain resources can be controlled and reproduced if necessary. Because the data, in the form of user health and behavior data acquired by fitness trackers tested here, can be seen as highly sensitive, robust authentication is a necessity. Commercial and even criminal interest in such data is

well documented by now, and devices handling such data should absolutely implement adequate measures to ensure a secured access.
For our test, we checked whether or not a given product implements an effective authentication procedure, both on the tracker side to verify the access by the app and on the app side to ensure it is communicating with an authentic tracker. As shown in Table 3.1, for 4 products we were able to identify a mechanism that serves the purpose of ensuring the authenticity of both communication partners – tracker and app. Although this method cannot be considered robust in every case, we acknowledged the mere existence and correct function of such a mechanism. Especially the implementation for the Xiaomi device is rather simple and is realized by basically one message consisting of user data (user ID, user alias, weight, height etc.) in a static order and unencrypted. Consequently, with knowledge of the message format, it is theoretically possible to guess a valid user login. Furthermore, we were able to use a replayed authentication message consisting of out-of-date user information to authenticate with the tracker and access its functionality. The three products from Mobile Action, Runtastic and Striiv do not provide a working authentication method, which as a consequence means that the complete tracker functionality is accessible immediately after establishing a connection.

Tamper Protection

As fitness trackers are nowadays also used in highly sensitive scenarios where the information is of benefit not only to the user, but also to health insurance providers, employers and even staff at courts, the potential for intentional manipulation of the tracker data and functions can have far-reaching financial and legal consequences. The aspects of data integrity and data authenticity therefore have to be secured by adequate measures.
In our tests, we checked which measures exist to ensure effective tamper protection, and we evaluated their robustness, i.e. we tested for the possibility of manipulating the data acquired by the tracker or the tracker function itself. Once again, the 4 products from Basis, Microsoft, Pebble and Xiaomi are providing at least a basic protection. For the first three in fact, we were not able to find an effective way of manipulating the data stored on the tracker or the tracker functionality itself. For the Xiaomi device, we were able to manipulate several functions after defeating the rather simple authentication mechanism. Thus, it is possible to trigger uncontrolled vibration, set or delete alarm timers or even perform a factory reset. As these functions are only protected by a rather weak authentication method, we had to slightly downgrade the rating for the Xiaomi product. On the Mobile Action and Striiv products, there is neither authentication nor any other protection against manipulation to be found, which precluded any rating of a protection function. On the Striiv *Fusion* we were able to manipulate user body measurements by changing them to superhuman dimensions. This information is directly used for the calculation of traveled distance and calorie burn and therefore leads to anything but realistic values. The same could be done for the Mobile Action device where we were able to change the user information without having to authenticate in any way. Figure 3.1 shows a listing of commands for the *Q-Band* we were able to identify and execute. Based on our tests, the Runtastic product was also not protected against unauthorized access, thus allowing some tracker functionality to be accessed from an unauthorized smartphone.



```
00 00 00 70 D5 01 70 17 A4 06 46 00 19 00 10 0E
00 00 1B 00
00 00 00 71 FF FF FF FF FF FF FF FF 00 00 00 00
00 00 00 00
00 00 00 72...
00 00 00 73-76...
1F 00 80 16
```

Figure 3.1: Sample list of commands for the *Q-Band* that could be executed without authentication

3.2 Application

This category serves the purpose of determining the overall security level of the mobile app for the tested product. In this step, we check whether the app saves (sensitive) data in an unsecured way on the smartphone and how well the app is protected against standard reverse-engineering approaches.

Secured Local Storage

The applications delivered with the fitness trackers normally store quite a bit of data locally on the smartphone. This may be necessary to temporarily save acquired fitness data until there is a possibility to synchronize with the cloud, to save credentials for offline login in the app or simply to keep statistics at hand. In all these cases, it is necessary to prevent access to this data by other applications, or in some cases by the user himself. By default, Android secures the standard app folders and saves destinations, thereby ensuring the security of the data. For smartphones without root access, this method can be considered reliably safe. However, even on devices without root access, any application has the possibility of using the unprotected storage space for temporary storage, for example, which is then theoretically accessible by any other app on the smartphone.
In our test, we evaluated whether all data stored by the app is secured against unauthorized access by third-party apps or by the user himself, as the access would most certainly invite the possibility of altering sensitive data. Out of all tested products, there is only one product where we were able to identify unsecured storage of critical data: the Xiaomi *MiBand*. The Xiaomi app writes a detailed log for the whole app activity, presumably as a relic from the debug version. This log contains synchronized data, online communication (including URLs and transmitted data) and user information such as alias and body measurements, which are also used for the authentication process. The log is saved to the SD card in plain text, where it is accessible for any other app on the phone.

Code Obfuscation

As already mentioned above, it is necessary to safeguard the tracker and access to the data collected by it. This includes the afore-mentioned tamper protection and authentication. The mechanisms implementing these aspects are rarely designed according to *Kerckhoff's Principle* [Kerckhoff, 1883]and their security is often based on keeping their mode of operation secret (*security by obscurity*). An attacker therefore has the possibility to identify and reconstruct the inner workings through targeted reverse-engineering. The low-hanging fruit for this is the application. It is freely available for all tested products, and with the wide selection of additional readily-available reverse engineering tools, it is child's play to gain access to the source code (or at least a decent representation of it through decompiling). An attacker can then easily identify the relevant functions and figure out how they work. Although code obfuscation cannot completely prevent reverse engineering of such measures, it can increase the time and effort required, and possibly deter the novice attacker. As this protection mechanism has no negative effects on application performance and is extremely easy to apply, we see it as a must-have tool.
As our tests show, however, there are only 3 of the tested apps which implement a sufficient level of code obfuscation – these are the apps from Mobile Action, Pebble and Xiaomi. The products from Basis and Runtastic use obfuscation but only partially and to a degree insufficient to effectively hinder app analysis. The Microsoft and Striiv apps completely forgo code obfuscation and therefore an app analysis for these products turns out to be rather straightforward. Figure 3.2 shows a very small excerpt (around 10%) from the class diagram of an app 'blurred' through code obfuscation. The major effort required for the search of a certain function within code protected this way is obvious. When paired with good obfuscation of the system-API calls, it greatly increases the time and effort for potential attackers.

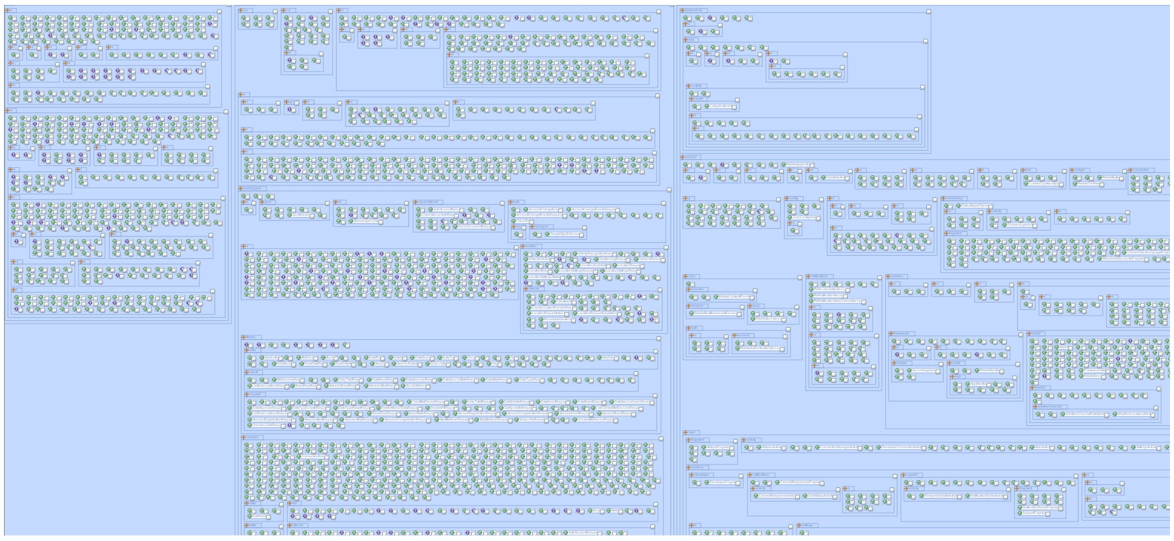


Figure 3.2: Excerpt from a class diagram of an app with strong (although not complete) code obfuscation

Log/Debug Output

Our tests indicated that many applications contain large amounts of debug code and consequently deliver lots of debug log output that often contains information which either helps reverse-engineering or even renders it unnecessary. Measures such as code obfuscation in particular can be rendered useless if the debug output of the app logs contains vital information about app flow and function calls. We checked the tested applications for unnecessary detailed debug/logcat output that delivers potentially valuable information in order to enable reverse-engineering of the app. In our tests there is only one product which can be seen as almost free from debug logcat output: the Mobile Action *Q-Band*. All other products have in some cases extremely detailed debug logs which in turn give many clues as to the mode of action of the app or the code locations where certain functions might be found.

3.3 Online Communication

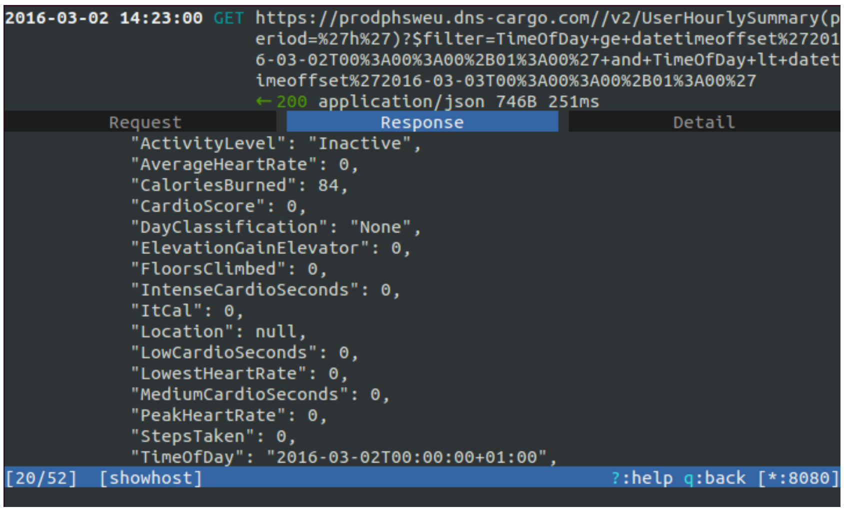
This category deals with the analysis of all incoming and outgoing traffic to and from the app. For the tests in this category, we checked whether the connections are encrypted and whether there is a potential of reading or even manipulating data via a man-in-the-middle without the server noticing. Here again, the danger of manipulation by third parties as well as by users themselves is also to be examined.

Encrypted Communication

Data transmission over the Internet should definitely be secured with up-to-date encryption nowadays. The activity and fitness data acquired and stored by fitness trackers is no exception for this. Although encryption alone is not sufficient today to ensure a high level of security, it is definitely a must-have for an adequate security concept.
In this test, we checked whether or not all important connections, especially those for the transmission of sensitive data (for user login, data synchronization etc.) are encrypted. According to our tests, there is no product that completely forgoes encryption of Internet communication. At least for the important communication such as user login and data synchronization all products use secured HTTPS connections. For 4 products, namely Basis, Microsoft, Pebble and Runtastic, absolutely no unsecured connection could be observed during testing. For the rest of the products we observed a few HTTP connections, but these were never used for the transmission of critical information: The Mobile Action app, for example, checked for a new firmware via HTTP, the Xiaomi app communicated with a few third-party domains in a unsecured manner and the Striiv app used some unsecured connections for rather uncritical purposes. All in all, we can state a quite high level of security for this category.

Tamper Protection

Although a connection is secured via HTTPS, there are still ways to infiltrate such a connection and manipulate its content. With the help of our test setup (see section 2.1) we analyzed all encrypted connections initiated or accepted by the tested product and checked whether or not we were capable of reading and/or manipulating the content. Although the necessity for e.g. SSL-Pinning[Pinning] can be estimated rather low for other applications, in our case and with consideration of the user himself as an attacker, it can become quite vital to safeguard against certain attacking scenarios. For our tests, we tried to read the content of the secured connections. If this is possible, we have to assume that there is at least a theoretical chance that an attacker would be able to manipulate the content. As shown in Table 3.1, there are only two products with sufficient protection from such a scenario. The two products from Basis and Pebble are able to recognize the man-in-the-middle before logging in and refusing the connection. All other products allowed us to read the content of the secured connections and in some cases a manipulation could be performed as well. Figure 3.3 and 3.4 show two excerpts from an intercepted authentication request and the content of a data synchronization. As we have to install a custom root certificate for this kind of attack, the practical threat for Android devices is actually rather low. But because it is not entirely inconceivable, especially when considering the user as the attacker, we have to slightly downgrade the rating for the unprotected products at this point.

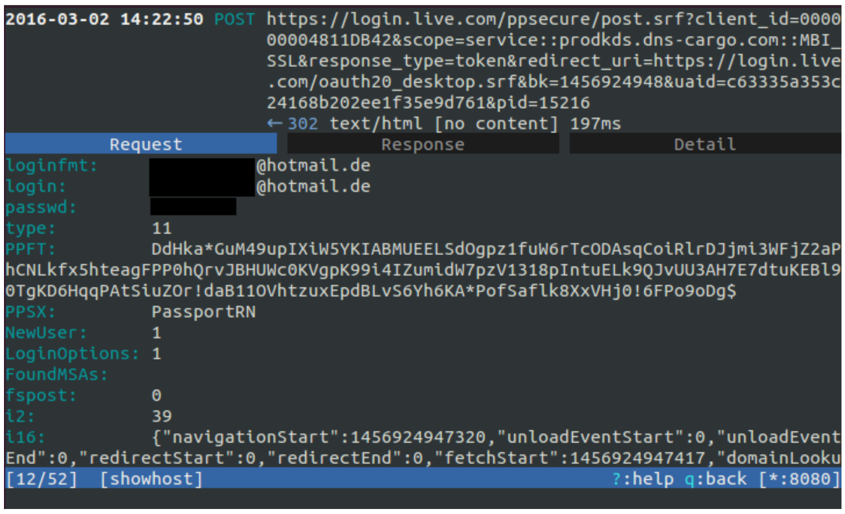


```
2016-03-02 14:23:00 GET https://prodphsweu.dns-cargo.com//v2/UserHourlySummary(p
                        eriod=%27h%27)?$filter=TimeOfDay+ge+datetimeoffset%27201
                        6-03-02T00%3A00%3A00%2B01%3A00%27+and+TimeOfDay+lt+datet
                        imeoffset%272016-03-03T00%3A00%3A00%2B01%3A00%27
                        ← 200 application/json 746B 251ms
          Request                 Response                 Detail
            "ActivityLevel": "Inactive",
            "AverageHeartRate": 0,
            "CaloriesBurned": 84,
            "CardioScore": 0,
            "DayClassification": "None",
            "ElevationGainElevator": 0,
            "FloorsClimbed": 0,
            "IntenseCardioSeconds": 0,
            "ItCal": 0,
            "Location": null,
            "LowCardioSeconds": 0,
            "LowestHeartRate": 0,
            "MediumCardioSeconds": 0,
            "PeakHeartRate": 0,
            "StepsTaken": 0,
            "TimeOfDay": "2016-03-02T00:00:00+01:00",
[20/52]  [showhost]                                  ?:help q:back [*:8080]
```

Figure 3.3: Authentication request (screenshot from *mitmproxy* UI)

3.4 *Apple Watch*

As we already mentioned above, the tests for the Apple product had to be performed in a slightly different manner, as not all aspects could be easily tested in ways analogous to the Android tests. Consequently, Table 3.2 showing the results for the *Apple Watch* is considerably shorter.



```
2016-03-02 14:22:50 POST https://login.live.com/ppsecure/post.srf?client_id=0000
                         00004811DB42&scope=service::prodkds.dns-cargo.com::MBI_
                         SSL&response_type=token&redirect_uri=https://login.live
                         .com/oauth20_desktop.srf&bk=1456924948&uaid=c63335a353c
                         24168b202ee1f35e9d761&pid=15216
                         ←302 text/html [no content] 197ms
          Request                    Response                    Detail
loginfmt:     [redacted]@hotmail.de
login:                  @hotmail.de
passwd:       [redacted]
type:         11
PPFT:         DdHka*GuM49upIXiW5YKIABMUEELSdOgpz1fuW6rTcODAsqCoiRlrDJjmi3WFjZ2aP
hCNLkfx5hteagFPP0hQrvJBHUWc0KVgpK99i4IZumidW7pzV1318pIntuELk9QJvUU3AH7E7dtuKEBl9
0TgKD6HqqPAtSiuZOr!daB11OVhtzuxEpdBLvS6Yh6KA*PofSaflk8XxVHj0!6FPo9oDg$
PPSX:         PassportRN
NewUser:      1
LoginOptions: 1
FoundMSAs:
fspost:       0
i2:           39
i16:          {"navigationStart":1456924947320,"unloadEventStart":0,"unloadEvent
End":0,"redirectStart":0,"redirectEnd":0,"fetchStart":1456924947417,"domainLooku
[12/52]  [showhost]                                       ?:help q:back [*:8080]
```

Figure 3.4: Synchronization (screenshot from *mitmproxy* UI)

	Apple *Watch* and *iOS 9.2*
Tracker	
Controlled Visibility	✔
Bluetooth Smart LE Privacy	≈
Controlled Connectivity	
Online Communication	
Encrypted Connections	≈
Tamper Protection	≈

✔Yes ≈Partially ✘No

Table 3.2: Summary of the test results for the *Apple Watch*

The "Airplane Mode" is a way to deactivate both Wi-Fi and Bluetooth on the *Apple Watch*. It also seems that the watch uses Bluetooth Smart LE Privacy, because it has a different MAC address every time it starts up. After activating the Airplane Mode, the watch does not use Bluetooth as intended. After deactivating the Airplane Mode the watch starts to send packages again, but unfortunately it uses its built-in and unaltered MAC address of the Bluetooth adapter.

The online communication is also not perfect. In fact, the smartphone communicates almost exclusively via TLS, but there are some unencrypted HTTP connections. Especially the presumed update files for the WatchOS 2.1 update were transferred unencrypted. This is no problem if there are other security mechanism like a signature or a secured checksum. Although attackers may prevent the update, they would not be able to put their manipulated version past the hardware.

Based on our results, we were able to conclude that nearly all encrypted connections were secured, provided our root certificate was not installed. We cannot rule out the possibility that these connections are linked to the Apple Watch, but we cannot say it with absolute certainty. The HTTP POST data to gsp-ssl.ls.apple.com, for example, includes text snippets such as "com.apple.GeoServices" and "com.apple.NanoWeatherKit" as well as "Watch1.1" and "2.1.13S6618". According to internal data, the current version of the *Apple Watch* is 1.1 whereas "13S661" is tied to WatchOS 2.1. It is a certain part of the answer which points to a larger problem: The response to the POST includes, among other things, the current user location and provides details such as the street number. Potential attackers using a man-in-the-middle approach would be able to read the encrypted connection and extract this kind of information. There are also other vulnerable hosts like gsp10-ssl.ls.apple.com. Under normal circumstances, a vulnerable TLS connection does not automatically represent a critical problem if no private or important data is transferred. In this case, however, the data transferred was the detailed user location. After activating the profile with our root certificate, we were able to read almost every connection. For this analysis, we did not use mitmproxy but a self-made application.
As can be seen in Table 3.2, the question involving controlled connectivity cannot be answered easily. It is not trivial to link an already linked Apple Watch to another account. Even if the watch is reset to factory settings, it refuses to link to another account. This is supposedly part of a theft protection mechanism from Apple. But in our tests, we could not determine whether this mechanism is also built into the Apple Watch or only part of the smartphone app and Apple's server. For theft protection, this usually doesn't matter, as duped customers would use the original apps with a stolen smartwatch, whereas an attacker with a malicious app does not need to. Without a clear answer to this question no entry was added to table 3.2.

All in all, we can say that, based on our results, the *Apple Watch* implements a strong level of security, yet suffers from minor problems with potentially greater impact in case of the insufficiently-protected user address information.

4 Implications for User Privacy

This chapter focuses on user privacy implications of wearing a fitness tracker. As can be seen in Table 3.1 we did not include this criterion directly in the evaluation of the products tested, as it seems rather absurd to validate a product regarding its impact on user privacy, when its sole purpose is to track every move of the user 24/7. Every user is aware of the major impact such a tracker may have on privacy. But nowadays these devices, which actually provide a scope of functions similar to a smart watch, are capable of so many things that a large share of users may not be completely aware of what they are actually revealing to the manufacturer of their fitness tracker or smart watch.

Let us first take a brief look at the product applications. Which permissions do they require and what implications do these have for the user? Table 4.1 shows an incomplete listing of the most sensitive permissions the tested products require when being installed. Of course, many of these are necessary to be able to provide the full range of functions to the user – a tracker can only show an incoming text message on its display if it has the permission to read text messages. But at this point, the sweeping implications of these granted permissions may not be clear to every user. With the text-related permissions to WRITE, SEND, RECEIVE and READ, the app has permanent unrestricted access to the complete text traffic on the phone and is even allowed to write and send text messages autonomously. And as most apps have the permission to automatically run in background (with the RECEIVE_BOOT_COMPLETED permission), this can be done at any time. With the CALL_PHONE and PROCESS_OUTGOING_CALLS permissions, apps are additionally allowed to initiate or end phone calls and even make them secretly. Along with the access to user location (GPS or connected networks) at any time, the ability to read the user's calendar (and manipulate it) and to know each of his contacts, such an app accumulates a critical mass of accessible data, above and beyond it is already collecting a plethora of sensitive private data 24 hours a day, making it extremely powerful indeed.

We have to give credit to the manufacturers, however, who, according to our tests are obviously refraining from further excessive data collection on user behavior, statistics, etc. At least our examinations revealed no suspicious or unusual signs. On the other hand, it is worth mentioning that these products are already collecting such masses of data legitimately, so that it is probably unnecessary to obtain more data, as the information content would hardly increase above and beyond what manufacturers know anyway. Many users are simply not aware of the information actually contained in their data. Of course anyone can check the specifications of their tracker, find out precisely which sensors are integrated in a specific device and which raw data is acquired with these sensors. But it is actually not the raw data like pulse, step count, calorie burn or location, each on its own, but rather the combination of all this data that represents the real value. The theoretically unlimited number of possibilities to combine, analyze and derive information from the raw data with modern Big Data analysis and data mining systems is the actual threat to the users' privacy. Nowadays, it is possible to detect certain user states, which the user may not necessary wish to reveal; whether he recently quit smoking or if he is drunk at the moment [Kawamoto et al., 2014] are just two examples. Even relationships between different users can be found, e.g. if there is a working relationship between them [Tsubouchi et al., 2013], by evaluating their fitness data. It is quite safe to assume that these examples only represent the tip of the iceberg, and manufacturers of fitness devices certainly know things about users that even they themselves are not aware of.

App Permission	Basis *Peak*	Microsoft *Band 2*	Mobile Action *Q-Band*	Pebble *Time*	Runtastic *Moment Elite*	Striiv *Fusion*	Xiaomi *MiBand*
ACCESS_COARSE_LOCATION	✘	✔	✘	✔	✔	✔	✔
ACCESS_FINE_LOCATION	✘	✔	✘	✘	✘	✔	✔
ACCESS_GPS	✘	✘	✘	✔	✘	✘	✘
ACCESS_ASSISTED_GPS	✘	✘	✘	✔	✘	✘	✘
CALL_PHONE	✔	✔	✔	✔	✘	✘	✔
PROCESS_OUTGOING_CALLS	✘	✔	✘	✘	✘	✘	✘
READ_CONTACTS	✔	✔	✔	✔	✔	✔	✘
READ_CALENDAR	✘	✔	✘	✔	✘	✔	✘
READ_SMS	✘	✔	✘	✔	✘	✔	✘
READ_EXTERNAL_STORAGE	✔	✘	✔	✔	✔	✔	✔
RECEIVE_BOOT_COMPLETED	✔	✔	✔	✘	✔	✔	✔
RECEIVE_SMS	✘	✔	✔	✔	✘	✔	✘
SEND_SMS	✘	✔	✘	✔	✘	✘	✘
WRITE_SMS	✘	✘	✘	✔	✘	✘	✘
WRITE_EXTERNAL_STORAGE	✔	✔	✔	✔	✔	✔	✔

✔Yes ✘No

Table 4.1: Permissions granted to product applications

5 Summary and Outlook

In our second test of fitness tracking devices, we examined 7 Android products and the *Apple Watch* in terms of their overall level of security. The products from Basis, Pebble, Microsoft and Apple made a favorable showing. These devices do not reveal any real weaknesses and generally offer a high level of security. All other products show at least some minor weaknesses regarding authentication and tamper protection. What is striking about the test results is the fact that none of the products show major flaws in terms of secure Internet communication. All the products protect the important aspects of user authentication and data synchronization when communicating via secure HTTPS connections. As our extended man-in-the-middle test demonstrated, however, on all the products except for Basis and Pebble we managed to sneak in and monitor the connection. In this respect we might also add that we were only able to do so by installing our own root certificate, which is not easily possible for an attacker under Android, and therefore was not considered a severe flaw. However, we identified quite a number of flaws regarding local communication, i.e. user authentication on the tracker side and protection of the tracker functionality. Overall the detected flaws are sufficient to question the use of fitness trackers for purposes which can have serious financial and/or legal consequences for the user.

Also worth mentioning is the product from Xiaomi, which based on our test evaluation (see Table 3.1) revealed many weaknesses from an objective viewpoint, yet this device, by far the most economical in the test, leaves an overall positive subjective impression, as it already includes security features by virtue of its design. The practical implementation of the security concept, however, leaves something to be desired, which is why it did not earn a high score overall.

The product from Striiv, *Fusion*, earned a poor rating. The equivalent product, Acer *Leap*, was already evaluated in last year's test [Clausing et al., 2015], and revealed glaring flaws. As this year's test indicates, Striiv *Fusion* suffers from all the same deficiencies – the security design of the basic device has apparently not been improved at all. Furthermore, based on experiences with the Acer *Leap* from last year's test, we were able to perform greater in-depth analysis of the Striiv *Fusion* and found even more vulnerabilities and possibilities for manipulation.

For future tests, we plan to more clearly illuminate the ramifications on user privacy, and we may additionally tighten the test specifications for online communication. We also plan to focus on the tracker firmware and its potential for manipulation. The scenario of the user as the primary attacker will also have greater influence on our test methodology, i.e. a way of manipulating the product app will be evaluated.

Bibliography

[Clausing et al., 2015] Clausing, E., Schiefer, M., Lösche, U., and Morgenstern, M. (2015). Internet of things - security evaluation of nine fitness trackers. Online. last access March 2nd, 2016. Available from: https://www.av-test.org/fileadmin/pdf/avtest_2015-06_fitness_tracker_english.pdf.

[Dörner, 2015] Dörner, S. (2015). Diese Krankenkassen bezuschussen die Apple Watch. Online. last access March 2nd, 2016. Available from: http://www.welt.de/wirtschaft/article144818188/Diese-Krankenkassen-bezuschussen-die-Apple-Watch.html.

[Fleischer, 2016] Fleischer, J. (2016). Fitness trackers can be used against you in a court of law. Online. last access March 2nd, 2016. Available from: http://www.wsbtv.com/news/news/local/fitness-trackers-can-be-used-against-you-court-law/nqHp4/.

[Gardner, 2016] Gardner, L. (2016). Fitness tracker data used in court cases. Online. last access March 2nd, 2016. Available from: http://www.news4jax.com/news/investigations/fitness-tracker-data-now-used-as-evidence-in-court-cases.

[Kawamoto et al., 2014] Kawamoto, K., Tanaka, T., and Kuriyama, H. (2014). Your activity tracker knows when you quit smoking. ACM, New York, NY, USA.

[Kerckhoff, 1883] Kerckhoff, A. (1883). La cryptographie militaire. Journal des sciences militaire.

[mitmproxy, 2014] mitmproxy (2014). mitmproxy. Online. last access March 2nd, 2016. Available from: https://mitmproxy.org/.

[Pinning] Pinning. Certificate and public key pinning. Online. last access March 2nd, 2016. Available from: https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning.

[Tsubouchi et al., 2013] Tsubouchi, K., Kawajiri, R., and Shimosaka, M. (2013). Working-relationship detection from fitbit sensor data. Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication. ACM.

[Vers, 2015] Vers (2015). Mehrere Krankenkassen bezuschussen Fitness-Armbänder. Online. last access March 2nd, 2016. Available from: http://fitnessarmband.eu/krankenkassen-bezuschussen-fitness-armbaender/.