Bachelor thesis
Computer Science

Radboud University

Getting access to your own Fitbit data

Author:
Maarten Schellevis
s4142616
maarten.schellevis@student.ru.nl

First supervisor/assessor:
Prof. dr. Bart Jacobs

Second supervisor:
Carlo Meijer

Second assessor:
Dr. ir. Joeri de Ruiter

August 19, 2016

Abstract

This study investigates the possibility of getting direct access to one's own data, as recorded by a Fitbit Charge HR activity tracker, without going through the Fitbit servers. We captured the firmware image of the Fitbit Charge HR during a firmware update. By analyzing this firmware image we were able to reverse-engineer the cryptographic primitives used by the Fitbit Charge HR activity tracker and recover the authentication protocol. We obtained the cryptographic key that is used in the authentication protocol from the Fitbit Android application. We located a backdoor in version 18.102 of the firmware by comparing it with the latest version of the firmware (18.122). In the latest version of the firmware the backdoor was removed. This backdoor was used to extract the device specific encryption key from the memory of the tracker. While we have not implemented this last step in practice, the device specific encryption key can be used by a Fitbit Charge HR user to obtain his/her fitness data directly from the device.

Contents

1. Introduction

An activity tracker is a device or application for monitoring and tracking fitness-related metrics. Measurements collected by activity trackers can be used to gain insight in your body's fitness and overall health. The focus of this report is the Fitbit Charge HR (fig. 1), and from this point onwards it will be called the tracker.

Figure 1: Fitbit Charge HR

This tracker is able to measure heart rate, daily number of steps taken, distance covered, number of calories burned and the number of floors climbed [1]. These values can be read out on the small display of the wrist worn device. Beside these values, additional data such as detailed heart rate history with heart rate zones, active minutes and sleep duration and quality is collected and can be obtained through your personal Fitbit dashboard (www.fitbit.com) or Fitbit app. The internal memory of the device stores minute-by-minute information of the most recent seven days, and 30 days of daily summaries [1].

Before additional information about your activities can be viewed online, the tracker has to send the data to the Fitbit servers. The data is sent from the device to your mobile phone or computer via Bluetooth Low Energy (BLE), and passed through to the Fitbit servers [1]. When the data has successfully arrived on the servers a response is sent to the tracker, either via the computer or mobile phone, to erase the data on the device [2, 3]. This process where the tracker transfers its data to the Fitbit servers is called synchronizing [1]. All data sent from the tracker to the Fitbit servers and vice versa is encrypted and cannot be accessed by the user [2].

Due to this encryption, the user is dependent on the Fitbit servers for analysis of the data. Users are not given the choice to have their data processed elsewhere. The goal of this study is to investigate whether it is possible for the user to get direct access to their data recorded by the activity tracker without interference of the Fitbit servers.

There are several reasons for this to be desirable. Firstly, in the present situation, the user gets a limited amount of data and analysis, but it is possible to get additional information by paying for the "Fitbit Premium" service. This allows the user to get more statistics and the possibility to download the data in an Excel file [4]. Although data directly obtained from the tracker might require processing before it can be used for analysis, for example by applying filters, it would give users the opportunity to perform their own analysis. In this analysis a user can choose the analysis method and also include data from other sources. Thus making it possibly to discover additional patterns in the fitness data or unexpected patterns in the combined data set.

Secondly, the data collected by the activity trackers is health data and may reveal a lot about your personal health and lifestyle. In an advice paper to the European Commission, the Data Protection Working Party (an EU advisory organ [5, Article 29]) explains the presumption that misuse of this category of data could have more severe consequences on the individual's fundamental rights than misuse of other, "normal" personal data [6]. They say that the misuse of sensitive data (e.g. if publicly revealed) may be irreversible and have long-term consequences for the individual as well as his social environment [6]. This presumption is the rationale behind the high level of protection for sensitive data in the EU. In order to minimize the possibility of misuse of their health data, users may want to limit the number of people with access to that data. An additional benefit of keeping the data locally is that it removes the additional risk that comes along with "in the cloud" storage.

A report on an earlier model of the tracker (Fitbit Flex) has suggested that further research into the tracker's firmware might give more insight into the encryption routines and procedures used by the device [2]. The authors mention that it might be possible to obtain the firmware by analyzing the communication during a firmware update of the tracker [2]. In this study, we attempted to obtain the firmware of the device and analyze the security mechanisms present on the activity tracker such as the authentication key management and the data encryption.

In this thesis, we demonstrate that it is possible to obtain the tracker's firmware image and to read out the tracker's memory including the cryptographic key. With the obtained cryptographic key it is possible to decrypt the fitness data coming from the device, and to gain code execution on the tracker. However due to time constraints we did not implement this in practice.

1.1 Side-note: Legal because interoperability

The EU directive 2009/24/EC on the legal protection of computer programs states: "The authorisation of the rightholder is not required where reproduction of the code and translation of its form are indispensable to obtain the information necessary to achieve the interoperability of an independently created computer program with other programs.” [7] However, the following conditions have to be met: the acts are performed by a person having the right to use a copy of the program, the information to achieve interoperability was not yet available, and the acts are confined to parts of the program that are necessary to achieve the desired interoperability [7]. Because this thesis meets all the criteria stated above it therefore remains within the boundaries of the law.

2. Methods & Results

2.1 Obtaining first firmware image

As suggested, firmware might be obtained during an update [2]. The attempt was performed with a freshly unboxed tracker, before any firmware updates were done. The update was done using a Windows PC with Fitbit Connect, the software for Windows. The tracker was connected to this computer using the Fitbit Bluetooth dongle.

We intercepted the firmware from the (normally) secured connection between Fitbit Connect and the Fitbit server by using a program called mitmproxy [8]. This program, running on a second computer serves as a transparent proxy fig. 2, all internet traffic from the first computer passes through the second one while not being specifically configured to do so [8]. The program logs all the internet traffic that passes through.

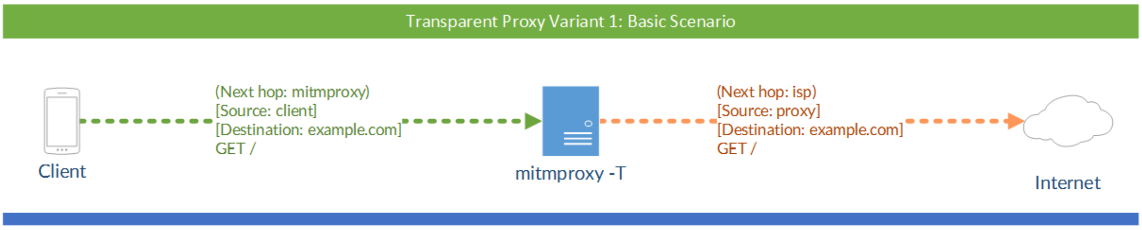

Figure 2: Transparant proxy The image is from the documentationg of the mitmproxy project.

Mitmproxy is able to log the encrypted internet traffic (HTTPS/TLS). On their website they say: “The basic idea is to pretend to be the server to the client, and pretend to be the client to the server, while we sit in the middle decoding traffic from both sides.” [9] Further information can be found on their website [8].

Before starting the update process the tracker is synchronized and the fitness data on the tracker is sent to the Fitbit servers. The fitness data is transferred in a data object called a megadump [10, 2]. The contents of this megadump were not encrypted with the initial firmware on the device. After the update the contents were encrypted. Additionally the first byte of the megadump changed from 0x2A to 0x2E, indicating that this byte could represent the format of the megadump instead of the type of tracker as earlier theorized [11].

2.2 Description of firmware format

The firmware of the tracker is transmitted over the internet using the JSON data format fig. 3. This JSON object contains a list with two "image" objects. Each image object has three attributes as can be seen in the figure. The "BSL" firmware image has the data key with the longest string value.


```
{
    "images": [
        {
            "data": "MQIAAAAAAQAAADCYAAASAQACAAg
            "deviceMode": "APP",
            "isV2Data": true
        },
        {
            "data": "MQIAAAAAAQAAAICMBAASAgCcAAg
            "deviceMode": "BSL",
            "isV2Data": true
        }
    ]
}
```


Figure 3: JSON firmware object

The firmware image data was not encrypted, since a clear pattern could be observed, but the data was base64 encoded. After decoding the firmware, it was not ready to be analyzed. The raw bytes of the firmware image as they will appear in the trackers memory were still encapsulated.

We have partially reverse engineered the encapsulation format. This encapsulation can be observed in fig. 4. The length and address fields in the headers are represented in little-endian format and the firmware image starts with a 14 byte message header. This header is followed by several "segments".

The firmware image was divided into blocks. Each segment consisted of a segment header and a block of raw firmware image. The block of firmware that is below a segment header is written to the trackers memory using the location address (starting address) in its segment header. This header also contains the length and a checksum of the block of firmware. This checksum(CRC-16-CCITT) is computed over the block of data to detect transmission errors.

Finally at the end of the firmware image was an 11 byte long "footer". This footer appears to start with 2 "random" bytes that could be a checksum and ends with the data length also specified in the first header but now incremented by four.

```
+--------------------------------------------------------------------------------------+
|   "message header" - size: 14 bytes                                                  |
|       unknown constant|??     |encrypted? |??          |data length                  |
|       3102            |0000   |0000       |01000000    |30980000                     |
|                                                                                      |
|   "firmware message data"                                                            |
|       +------------------------------------------------------------------------+   |
|       |     "segment header #1" - size: 16 bytes                               |   |
|       |   ??  |??          |starting address   |segment length |again??    |CRC16  |   |
|       |   12  |01 or 02    |00020008           |00980000       |00980000   |EE36   |   |
|       |   -------------------------------------------------------------------  |   |
|       |     "data of segment #1"                                               |   |
|       +------------------------------------------------------------------------+   |
|         .                                                                            |
|         .                                                                            |
|         .                                                                            |
|       +------------------------------------------------------------------------+   |
|       |     "segment header #last" - size: 16 bytes                            |   |
|       |   ??  |??          |starting address   |segment length |again??    |CRC16  |   |
|       |   12  |03 or 04    |00000000           |00000000       |00000000   |0000   |   |
|       |   -------------------------------------------------------------------  |   |
|       |     "data of segment #last"                                            |   |
|       +------------------------------------------------------------------------+   |
|                                                                                      |
|   "footer" - size: 11 bytes                                                          |
|       unknown |??            |data length + 4                                        |
|       XXXX    |000000000000  |349800                                                 |
+--------------------------------------------------------------------------------------+
```

Figure 4: Format of firmware

We wrote a python script for parsing the headers of the firmware image, producing raw images ready for analysis. Concurrently, we also wrote a script for reconstructing a firmware image after firmware modification, but this script has not been used in this study. After parsing out the headers, all that remains is the firmware image itself. Before we could begin analyzing the code we needed to know the following details first, where the firmware image will be placed in the memory of the tracker (starting address in fig. 4) and what instruction set was used by the processor of the tracker.

The instruction set used was deduced from information on a site which stated that the STM32L151RD microcontroller is used in this type of tracker [12]. This microcontroller contains an ARM® Cortex®-M3 32 bit processor [13], which uses the ARMv7-M instruction set [14]. The datasheet of the microcontroller proved invaluable during the reverse engineering efforts.

2.3 Firmware analysis: Encryption Cipher

We disassembled the firmware image using IDA PRO [15]. Where possible, this program automatically translates the firmware image into assembly source code. In some places the program could not successfully or correctly disassemble the firmware for different reasons for example deviations from standard calling conventions presumably due to optimizations by the compiler. Therefore as well as interpreting the firmware we further corrected and annotated the image manually.

Our first objective was to uncover the encryption algorithm used by the tracker. To find the correct cipher for the encryption algorithm two promising candidates emerged. The first probable cipher is the commonly used AES (Advanced Encryption Standard) [16]. AES is widely known and it was already speculated that it is used in some Fitbit devices [11]. The second candidate is XTEA (eXtended Tiny Encryption Algorithm) [17] which is used by the Fitbit Flex, an earlier tracker model, for authenticating Bluetooth communication between the Fitbit device and smartphone or computer application [2].

The method we used to locate the encryption algorithm was to look for values that are often used in the implementation of the candidate ciphers. We started with searching for the lookup table of the Rijndael S-box used in the AES algorithm. The lookup table was not found in the firmware image. Next, we searched the firmware image for the XTEA delta value (0x9E3779B9) [17]. This value was present in the firmware image and via this route we were able to locate the XTEA cipher subroutine that was present in the trackers firmware.

This cipher was used for data encryption, key derivation as well as in the authentication mechanism. We will discuss and describe some of these mechanisms in more detail in section 2.7 and section 2.5.2 of this report. By analyzing the information flow, we found that the keys used by these mechanisms were in the RAM of the device. All these keys originated from one single key that is stored in the EEPROM of the tracker.

2.4 Obtaining second firmware image: backdoor disappeared

During the project a new firmware update became available [18]. This update contained a critical security update [18], which is defined by Fitbit as: "The security update patches a vulnerability that if exploited could allow attacker-supplied code to gain unrestricted access and potentially go undetected by the customer." [19]

The previous update (fig. 5.a) changed the updating mechanism of the tracker. With the current update (fig. 5.b), the tracker receives an encrypted firmware image. To bypass this new mechanism and obtain the unencrypted firmware image, the request from the initial update (fig. 5.a) was replayed (fig. 5.c). Using this method the latest firmware image in unencrypted form was obtained. In order to utilize the vulnerability present in the firmware image on the tracker, the update was aborted prior to the installation being completed.

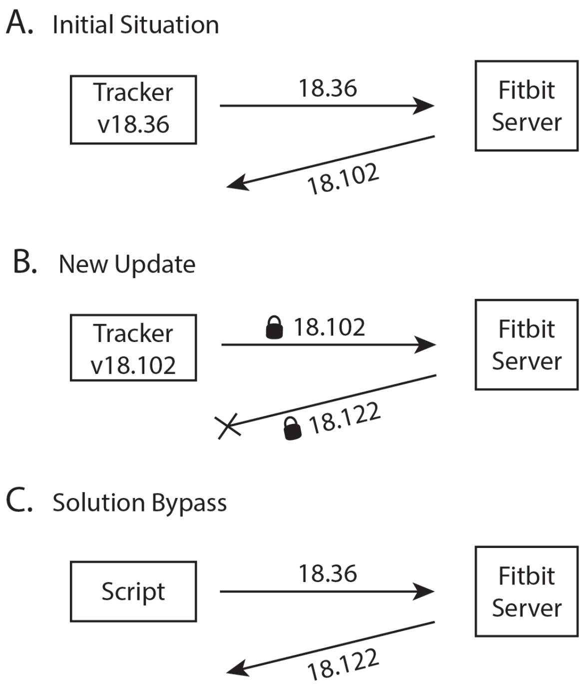

Figure 5: Updating Mechanism

2.4.1 Comparison of firmware images: locating vulnerability

After preparing the new firmware image for analysis as described in section 2.2 of this report, the two firmware images were compared. Differences between the two versions allowed us to find the security vulnerability present in the firmware on the tracker.

The firmware contains a subroutine that processes the different commands that can be sent to the tracker via Bluetooth. We observed that one of the commands in this subroutine had been removed from the firmware image. Reverse engineering of the code uncovered that the command offers the functionality to transfer the tracker's memory contents with Bluetooth. The command (fig. 6) contains the memory address from where the data must be read as well as how many bytes must be read.

```
[__1__] [______2______] [______3______]
[C0 11] [XX XX XX XX] [XX XX XX XX]
1. HEADER
2. memory address
3. length

Use:
Retrieve specified amount of data
from the tracker at specified address
```

Figure 6: Backdoor Command

However, before this backdoor command can be used, the tracker must be in a state where some variables are set to specific values. This can be done by using a specific sequence of commands starting with 0xC0 50, 0xC0 51 and 0xC0 52. These commands are used in the authentication handshake between a tracker and dongle, Android- or iPhone application [20] [21]. This means that being authenticated to the tracker is a prerequisite for the use of the found backdoor command. The next section of this report is about authenticating to the tracker.

2.5 Authentication mechanism

At the start of our investigation we had a limited amount of information about the authentication mechanism [21]. In this section we will describe the steps by which we uncovered more about this mechanism.

First, we will describe how we located the places where the authentication mechanism was used. Second, we will elaborate on the computations used in the authentication mechanism. Finally, we will describe how we obtained the key used to calculate the messages.

2.5.1 Finding the authentication mechanism

To find all the information we needed, we started by figuring out if the authentication mechanism is used by the Fitbit Connect program and/or the Fitbit Android application.

First we examined if the authentication mechanism was used by Fitbit Connect program. Namely, if the authentication handshake can be found in the communication between the tracker and the computer using the dongle. This was performed by viewing at the logs of the Fitbit Connect program. These logs are encrypted by default. A member of the mailing list about Galileo, the project for synchronizing Fitbit trackers using Linux, noted [22] that the encryption of the logs of the Fitbit Connect application can be turned off by adding the line <encryptLogs value="false" /> to the configuration file located at `C:\ProgramData\FitbitConnect\fitbit_connect_config.xml` . With the log encryption turned off the logs could be read. Unfortunately, we were unable to find any indication in the logs that Fitbit Connect uses the authentication mechanism.

Subsequently, we examined if the authentication handshake was performed by the Fitbit Android application. The smartphone used was a Samsung Galaxy S4. We enabled the Bluetooth HCI snoop log in the Developer options on the phone to capture the Bluetooth packets sent and received by the phone. In this log, the characteristic sequence of messages of the authentication handshake was found. This was a strong indicator that the authentication mechanism is contained in the app.

2.5.2 Authentication mechanism description

Next, we tried to find and reverse engineer the authentication mechanism in the Android application. We chose this method because it might be easier to reverse engineer the decompiled Java code instead of the assembly code coming from decompiling the firmware image, since reading assembly code is not always straightforward. We analyzed the Fitbit Android application to study the contents and the computation of the messages used in the authentication handshake. This was done by decompiling and analyzing the Android application using a variety of tools such as JD, apktool and dex2jar [23, 24, 25].

When the tracker is being paired to an Android device with the Fitbit application, the application retrieves an "authSubKey" and a "nonce" from the Fitbit server [2]. This behavior may also happen for the iPhone application but we didn't perform this analysis. While the term nonce is used by Fitbit, strictly speaking this term is not correct since the nonce is coupled to the key used and is used more than once. From this point this "nonce" will be called "authentication key salt".

The content (fig. 8) and the computation of the messages of the authentication handshake (fig. 7) will be described now. Figure 8 contains the message format, this is a completed version of the information found on [21]. With this handshake the application authenticates to the tracker and vice versa. The handshake consists of three messages. We discovered that the authentication key was used by the smartphone to compute a CMAC with XTEA in CMAC-mode (Cipher-based Message Authentication Code) [26].

The first message(fig. 8.a) is sent from the smartphone to the tracker (fig. 7.a). This command message contains a random value and the authentication key salt that was received during the pairing process.

The tracker then derives the authentication key using the authentication key salt, this process will be explained later in this report. The computation of the CMAC is initialized with the authentication key. The input values used to compute the CMAC are the random value that is given by the application and the counter, a number in the tracker that is incremented each authentication attempt. This CMAC and the counter are sent in the second message of the handshake (fig. 7.b).

Upon receiving this message, the smartphone application checks if the CMAC provided by the tracker is valid. After that it computes a CMAC of the counter and sends this CMAC to the tracker in the final authentication message (fig. 7.c). Finally the tracker sends an acknowledgement that the authentication has succeeded.

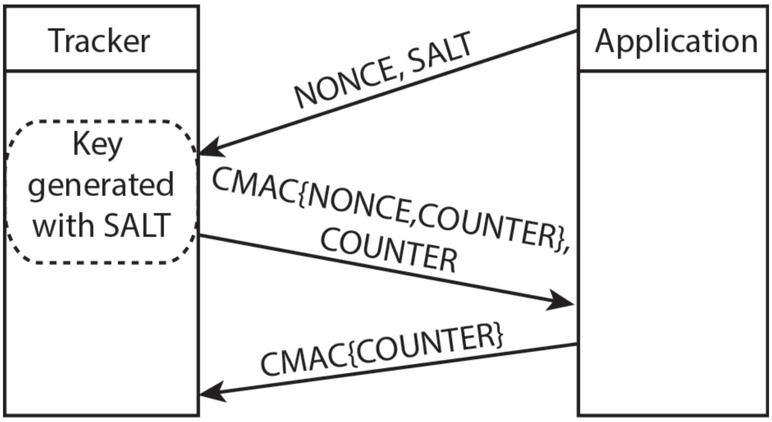

Figure 7: Authentication Handshake

```
A.  [__1__] [_____2_____] [_____3_____]
    [C0 50] [XX XX XX XX] [XX XX XX XX]
    1. HEADER - FBAirlinkOpcodeClientAuthStart
    2. random number
    3. authentication key salt

    Use: Starts the authentication handshake,
    sent to device

B.  [__1__] [___________2___________] [_____3_____]
    [C0 51] [XX XX XX XX XX XX XX XX] [XX XX XX XX]
    1. HEADER - FBAirlinkOpcodeClientAuthChallenge
    2. CMAC of the random number and the counter,
       the key used for the CMAC is derived using
       the authentication key salt
    3. counter (increments with each challenge)

    Use: response to Auth Start,
    sent from device

C.  [__1__] [___________2___________]
    [C0 52] [XX XX XX XX XX XX XX XX]
    1. HEADER - FBAirlinkOpcodeClientAuthChallengeResponse
    2. CMAC of the counter

    Use: This should authenticate the handshake,
    sent to device
```

Figure 8: Authentication Bluetooth Commands

2.5.3 Obtaining the Authentication Key from the Android app

In addition to the authentication mechanism, the authentication key is required to authenticate with the tracker. As stated previously the Android application retrieves an authentication key when pairing to a tracker [2]. This one-time event indicates that the authentication key is stored by the Android application. In this section we will describe how we obtained the key from the Fitbit Android application.

We observed that the Fitbit Android application contains logging statements. The logging mechanism of the application could be activated by changing the instance of the com.fitbit.config.Buildtype class from RELEASE to DEBUG. We installed the modified application on the Android device. With this modified application we started an update of the tracker firmware. We interrupted this process before the firmware was successfully installed on the tracker by disabling the Bluetooth connection on the phone. During this firmware update attempt, the Android application authenticated with the device. After these steps we could successfully obtain the authentication key from the log file.

2.6 Reading out the memory: Obtaining the encryption key

Our goal was to authenticate to the tracker in order to read out the tracker's memory using the backdoor command we found earlier (2.4.1). With the knowledge of the authentication mechanism and the authentication key in our possession, we started the implementation of the authentication mechanism.

We chose to extend the existing Galileo software [10] since this already contains a lot of what we need, such as interaction with the Bluetooth dongle, discovery of devices, etcetera. We did not implement checking the validity of the CMAC that is provided by the tracker as this is not strictly necessary at this point. Furthermore, we added the functionality to read the memory of the device using the found backdoor command. Several types of memory banks can be obtained e.g. the Flash memory Banks, the Data EEPROM Banks, the System memory Banks, the Option Byte Banks and the entire RAM [13].

This backdoor command provides the user with the entire memory of the tracker. Because this includes all the firmware as well as all the data, including the keys, the user in principle has the information that is needed to create a modified firmware image that will be accepted by the device as an update and thereby gain code execution.

2.7 Computation of the authentication key

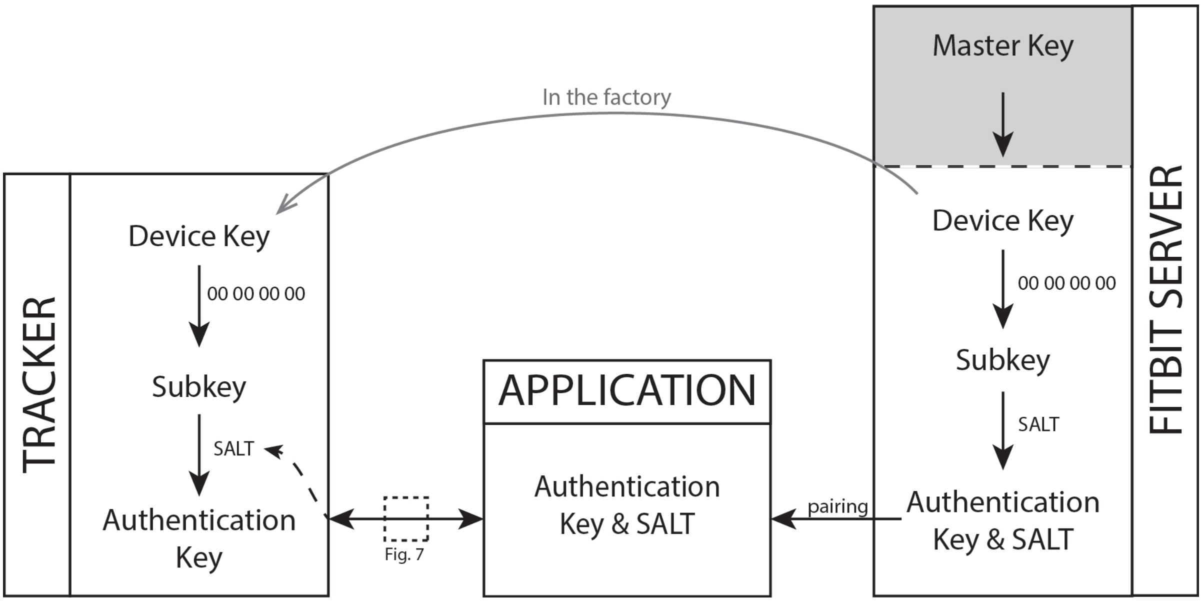

Figure 9: Hierarchy of Encryption Keys The grey areas in the figure indicate speculation.

The keys that are used by the tracker are related. In fig. 9 it can be observed that the 'authentication key' is derived from the 'device key' in two key derivation steps. In the first step a constant is used as the input for the derivation. In the second key derivation step for the authentication key, the tracker uses the salt as the input. The salt is received from the Android application in the first message of the authentication handshake(fig. 7.a). Each key has a different context. More specifically, the 'Device Key' is known by the tracker and Fitbit, while the authentication key is also available to the Android application. Fitbit calls a key that is derived from another key a 'subkey' (for example "authSubKey"). We hypothesize that the device key itself could also be computed using a master key. The input for such a computation could for example be the serial number of the device.

Now we will describe the subkey computation using fig. 10. The 16 byte subkey is a concatenation of two 8 byte CMAC's. As can be seen in the figure each XTEA CMAC is initialized using the key, updated with the little endian representation of the number 1 or 2 (depending position CMAC),

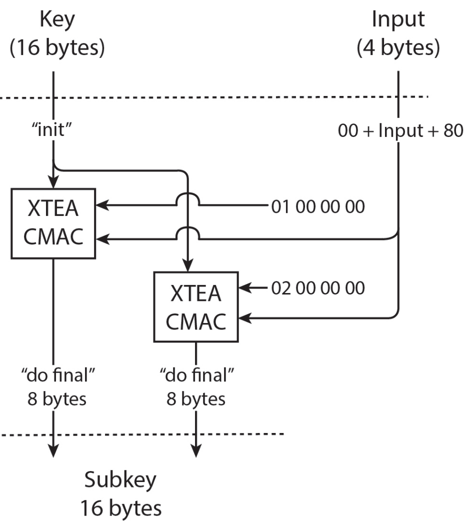

Figure 10: Subkey Computation

updated with the input and finally the CMAC is computed. We verified this computation using the data we obtained from the tracker.

3. Discussion

3.1 The chosen approach: Firmware analysis

In this report the firmware of the Fitbit Charge HR was analyzed, this approach has several advantages. First of all, this evades invasive procedures with tools that might damage the device. Using this method, no physical access to the electronics of the device was needed. Secondly, the several steps of the key retrieval can be automated in a script, making it easier to repeat the process by other users. Such a script would make it possible for other people to apply the findings of this paper to their own device. This would allow them to obtain their fitness data directly from their Fitbit activity tracker.

3.2 Attack authentication mechanism

In this study the possible weaknesses of the authentication mechanism were not studied. Weaknesses in the authentication mechanism could be used by an adversary to gain authentication with the tracker. In this section we will discuss potential ways to authenticate and the potential consequences.

It is unclear for which actions authentication is required. Therefore, the consequences of authentication are not fully understood. The Android application uses authentication for synchronizing the fitness data recorded by the tracker and for updating the tracker's firmware. However, we do not know whether this authentication is always a requirement for a firmware update.

Replaying a previously sent message authentication code (MAC) is not a valid option. The counter is increased on every authentication attempt (fig. 7). As a result each authentication attempt, challenge and thus the response are different and replaying the MAC does not have the desired effect.

Interestingly, the key used to authenticate is equal for both the Fitbit application and the tracker. This may implicate that the tracker can be used as a signing oracle for its own challenge, since the challenge sent by the tracker is sequential and hence completely predictable. Although we have not thoroughly researched this possibility extensively, to the best of our knowledge, it is infeasible to find two messages of different lengths that yield the same output once processed by the CMAC function, given that the key is not known. Therefore, the MAC sent by the tracker cannot be used as a response to the its own challenge since the nonce is also included in the computation of this MAC (fig. 7).

An attacker might be able to relay the communication between the application and a tracker, let the application authenticate and then use this authentication to send his/her own command instead of the command intended by the application to the device. If this is possible, and if the application can be triggered into authenticating the tracker without interaction with the user, this provides the attacker with an opportunity to send commands to the trackers of people walking by without being noticed. Using this mechanism the attacker could possibly turn the tracker into a surveillance device by updating the tracker with altered firmware.

3.3 Explanation for backdoor

We could not find any indication in the firmware image as to why this backdoor was present in the firmware. The most likely explanation is that it was used during the development of the device, but was not removed by the developers.

3.4 Separation product and service

In our opinion a separation of monitoring devices and the fitness data processing would be preferable. It is evident that the firmware is an essential part of the tracker. The analysis of the fitness data can however be considered as a separate product. In the current situation, in the case of Fitbit, the tracker and the data analysis are coupled by technical means, i.e. encryption. This could be seen as a form of abuse of its dominant market position, since it restricts the market of health data processing services [27] [28]. However, it remains debatable if this really applies.

In 2018 a new European data regulation will introduce the right of data portability [29]. This enables a user to transfer his/her data from one controller to another. With the introduction of this new regulation it becomes more obvious that separation of the tracker and the data analysis is necessary. As currently this portability cannot be performed for the Fitbit data, due to the encryption. It is likely that with the start of the new regulation changes are required for the Fitbit data processing, however it is not yet clear how these will be implemented in practice.

3.5 Bluetooth address of Tracker

The Activity tracker identifies itself using a Bluetooth MAC (media access control) address. This address is fixed for the activity tracker [2] [30]. This fixed address is used as an identifier for the tracker. This enables other parties to monitor trackers in the vicinity, resulting in the unauthorized disclosure of personal information. In Bluetooth 4.0 a feature called LE Privacy was added to solve this problem [31]. With this feature, the MAC address is changed at regular intervals.

In earlier research [30] Fitbit was notified about the of this situation. Fitbit responded that their wearable devices could support LE privacy, however the fragmented Android ecosystem prevented them from implementing it since some devices do not support this feature [30].

According to the EU Data Protection Directive the controller, in this case Fitbit, must implement appropriate measures to protect personal data against unauthorized disclosure or access [5].

Since the Bluetooth address is an identifier for the device, it can be used to observe if the user wearing the device is present. The location of the user is personal data and therefore LE privacy should be implemented and enabled by default. In case an Android device does not support LE privacy, the user, after being properly informed of the situation, can be given the opportunity to disable the LE Privacy on their tracker.

4. Further Research

Several steps are needed before a user, with his/her fitness data locally, can use the tracker to the same extent as a user with his/her data synchronized to the Fitbit server.

- A tool could be made to synchronize the tracker locally. The final step of this synchronization process is sending a response to the tracker that it can erase the data from its memory. To create this response, some of the information from the megadump is probably necessary. In order to obtain this information, the megadump should be decrypted and to some extent parsed. Initial efforts to parse the megadump are described in [32]. With the parsed information, an appropriate response could be generated and send back to the tracker.

- Several steps are needed to create statistics after decrypting the data of the megadumps coming from the tracker. The format of the decrypted data could be described. With the description, the data can be fully parsed and converted to a more common format which will ease the next step. The data probably needs to be filtered and sanitized before it can be used any further. Afterwards the data could be used to generate statistics, figures or diagrams. New tools can be created or existing tools could be adapted for this purpose.

For future research it would be interesting to see if the backdoor command found in this research, which can be used to read the memory of the tracker, is present in earlier versions of the tracker's firmware. This could provide users who bought a new tracker, with a relatively easy method of finding the key used for the encryption of their data.

It would be good to investigate if the backdoor command is also present in the firmware of other types of devices by Fitbit. Additionally,investigating if the other findings are applicable to other products from Fitbit is desirable.

At the moment, no validation of the authenticity, either by means of cryptographic signatures, or otherwise, seems to be present in the firmware image of the initial update (fig. 5.a). This offers the possibility to update the tracker with modified firmware. Users could use this to experiment with adding new functionality to the device.

As discussed, the authentication mechanism could be further analyzed. The authentication mechanism could be studied to locate possible security weaknesses. Weaknesses in the authentication mechanism could be used by an adversary to gain authentication with the tracker. To fully understand the consequences of authentication, the actions for which authentication is required should be investigated.

In this report a technical approach was used to get access to the data recorded by the tracker. It would be very interesting to investigate if it is possible to reach the same goal using a very different approach. The benefit of this alternative approach is that it could work for all tracker models. Instead of through technical means a user could use his/her ”Right of access” as described in [5]. This right states that the user could obtain the data undergoing ”processing” from the controller in an intelligible form. The definition of ”processing” used in [5] includes storage. In the case of a Fitbit activity tracker this would mean that if a user has his/her data recorded by the device but does not want to synchronize the data at the expense of his privacy, he/she can argue Right of access to his/her data stored on the tracker. In order to obtain the data from the tracker in an intelligible form, a user needs the device specific encryption key and instructions on how to decrypt the data and both should be provided by Fitbit.

5. Conclusions

We have reverse-engineered the cryptographic primitives used by the Fitbit Charge HR activity tracker and recovered the authentication protocol and the device specific encryption key. Furthermore, we found several serious vulnerabilities in version 18.102 of the Fitbit Charge HR firmware. Of which the most serious is the backdoor in the Bluetooth command interpreter, enabling an attacker to extract the device-specific key. This allows the attacker to decrypt the data transmitted to the Fitbit servers, or worse, gain code execution on the tracker by creating a custom crafted firmware image. Fortunately, we found that this backdoor was removed by Fitbit in its latest version of the tracker firmware. However, we have not verified the existence nor the absence of the backdoor on other models. The possibility of a relay attack on the authentication protocol was also discussed. Finally, we have made it possible for the Fitbit Charge HR user to obtain his/her raw fitness data directly from the device, without interference of the Fitbit servers. However, we have not attempted to process this raw data.

A. Protection of personal data in the EU

Fitbit sells its products in the European Union (EU). The EU has adopted the Data Protection Directive, which regulates the processing of personal data within the European Union [5]. However, Fitbit states in their privacy policy that the service they provide operates entirely from the United States. Therefore, they state that all the personal information that you provide to Fitbit is processed in the US, and is subjected to the US law [33].

Up until 2015, the data from Fitbit was sent legally to the US using the framework for transatlantic data flows called the Safe Harbor Privacy Principles [34]. In 2000 the European Commission determined that these principles provide an "adequate" level of privacy protection [34]. However, on 6 October 2015 the European Court of Justice ruled that the Safe Harbor framework is invalid, and a new law called the EU-US Privacy Shield went in effect on 12 July 2016 [35]. Despite the improvements of the Privacy Shield compared to the Safe Harbor decision, the European Data Protection Working Party [5] still has concerns regarding both commercial aspects and the access by US authorities to data transferred from the EU [36].

Several aspects of the Data Protection Directive and in what manner Fitbit complies with these aspects are discussed below.

A.1 Monitoring the application of European privacy law

The Data Protection Directive determines that each European member state has its own supervisory authority [5]. The tasks of a supervisory authority (or authorities) are advising, informing and obviously supervising [37]. More specifically, the authority is responsible for monitoring the application of the laws of a member state according to the Data Protection Directive [5]. Additionally, they are consulted when new administrative measures or regulations are made [5]. The supervisory authority has the power to investigate and intervene with the data processing happening in its member state. It also has the power to engage in legal proceedings when the law implemented in the Data Protection Directive has been violated [5]. The supervisory authority of the Netherlands is called "Autoriteit Persoonsgegevens" [37].

A.2 Record of data processing

The Data protection directive (Article 18) states that a company must notify the supervisory authority of a EU member state before carrying out any operation on personal data in that country [5]. One of the examples of operations is collection. This notification contains e.g. the purpose of processing the data, the recipients of the data, a general description of the measures taken to ensure security of processing and a description of the subjects involved. This information is kept in a public register in each EU member state [5].

In the Netherlands, the country where the tracker was purchased, the supervisory authority, has no record in its public register for Fitbit [38].

A.3 Valid Consent?

Processing of personal data concerning health or sex life is prohibited according to article 8 of the Data Protection Directive [5]. There are however exceptions possible e.g. when the data subject has given his explicit consent to the processing of those data [5].

At the end of 2015, the "Autoriteit Persoonsgegevens" (AP), the Dutch supervisory authority published a report on the Nike+ Running app and data processing [39] [40]. According to the AP, the Nike+ Running app illustrates the ongoing trend that electronic devices, such as smartphones, are used increasingly often to monitor health and lifestyle [40]. The companies selling these health monitoring devices, process the collected health data. The AP discusses "express consent" given by the user and concludes that Nike did not inform its users sufficiently about the types of personal data it collected and processed and for what purposes. Therefore, they could not rely on this exception to the processing of personal health data.

This raises the question in which situation consent by the user is valid. The Data Protection Working Party has published "The Opinion" [41], a document regarding the definition of consent. This Opinion provides examples on valid and invalid consent, focusing on its key elements e.g. "freely given", "informed" and "specific". In the next section, these key elements regarding the Fitbit privacy policy and the given consent by the user will be discussed.

A.3.1 "Freely given"

The first element of consent to be discussed is "freely given". The Opinion clarifies: "Consent can only be valid if the data subject is able to exercise a real choice, and there is no risk of deception, intimidation, coercion or significant negative consequences if he/she does not consent." [41]

A consumer does have the choice to not buy or use an activity tracker. However, the moment when the user is expected to give consent is after the purchase of the tracker. This leads to the situation where the user wants to start using his tracker but first is required to give his consent. Not accepting the terms puts the user in the situation where he cannot use the tracker, since having the fitness data from the tracker processed elsewhere is not possible. The user could try to return the product, which takes time and effort, or not use it at all, essentially throwing away the money spent. Additionally the decision of the user can be influenced by the expectation of starting to use the product. These contextual elements make that the consent asked from the users is not entirely "freely given".

A.3.2 "Informed"

The Opinion states there is an obligation to provide information to the users. The information must be given directly to the users and it is not enough for information to be "available" somewhere. The information must be clearly visible, prominent and comprehensive [41]. When a user creates an account using the Fitbit Connect software, he must accept the terms of service and the privacy policy. Both are not displayed directly but are referred to using hyperlinks.

The user also should be given clear, accurate and full information of all relevant issues, such as the nature of the data processed [42]. This also includes an awareness of the consequences of not consenting to the processing in question [42]. In the privacy policy, the nature of the data processed and the consequences of not consenting are not discussed [33].

A.3.3 "Specific"

The last element of consent that will be discussed here is "specific". The Opinion states: "To be valid, consent must be specific. In other words, blanket consent without specifying the exact purpose of the processing is not acceptable. To be specific, consent must be intelligible: it should refer clearly and precisely to the scope and the consequences of the data processing. It cannot apply to an open-ended set of processing activities. This means in other words that the context in which consent applies is limited. Consent must be given in relation to the different aspects of the processing, clearly identified. It includes notably which data are processed and for which purposes."

Now we will look at how "specific" the consent is, that is, how clear and precise the scope and consequences of the data processing are explained in the privacy policy of Fitbit.

Specificity of the Collected data

The information provided about the data collected seem to be not specific enough given the information in the Opinion. The Fitbit privacy policy lacks clarity as well as precision, on what is collected. For example under the section that explains what data is collected "When You Activate a Fitbit device" [33], it says that the user is asked to enter information about himself, such as height, weight and gender. Also noted in this section is that "Depending upon the specific Device you use, it can collect the number of steps you take, your weight, measure your sleep quality and transmit this data to Fitbit." Followed by a link to a webpage to see the full list of data that the device collects. The link leads to a 404 not found webpage [43].

Specificity of the data processing

Furthermore, the policy is not specific about all processing that is performed and for which purpose. The policy states: "Fitbit uses your data to provide you with the best experience possible, to help you make the most of your fitness, and to improve and protect Fitbit." After this statement they continue to list an open-ended set of processing activities in the form of a list of examples.

The privacy policy also contains a section on the sharing of data. In the section "Data That Could Identify You", Personally Identifiable Information (PII) is defined as data that could reasonably be linked back to you. One of the circumstances under which Fitbit will share PII is notable: "If it is necessary in connection with the sale, merger, bankruptcy, sale of assets or reorganization of our company, your PII can be sold or transferred as part of that transaction as permitted by law. The promises in this Privacy Policy will apply to your data as transferred to the new entity."

The first reason why this sentence is notable is because it contradicts a previous statement in the policy that says: "First and foremost: We don't sell any data that could identify you." The second reason is that it is unreasonable to expect knowledge from the average user of the tracker concerning the laws involved in sale, merger, bankruptcy, sale of assets or reorganization of companies in the United States of America. Moreover, current laws might change in the future making the situation even more complex.

A.4 Consequences of not consenting to the data processing

Here we will discuss the consequences of not accepting the terms and the privacy policy of Fitbit. If the user does not accept the terms and the policy, he/she cannot make a Fitbit account. To update the firmware of the tracker, you need to login using your Fitbit account. This means that a user who does not give his consent cannot update his/her trackers firmware and thus has his/her personal data at risk. This is because the fitness data coming from the tracker is not protected in any way with the out-of-the-box firmware. Additionally, the device is vulnerable to malware in the form of firmware modified by an attacker, since the security updates are not applied. Even if the user updates the tracker later on, a malware infected tracker could mimic the update process leaving the user unaware of the data breach.

Users cannot simply update the tracker without synchronizing it, meaning that they will essentially use the tracker only for viewing the information directly on the device. This is not possible because the tracker synchronizes the fitness data with the Fitbit server before the firmware update process starts. This makes having the latest, most secure firmware without sending the fitness data impossible.

A.5 Reform of European data protection regulation

On 8 April 2016, the European Council adopted a new regulation on the European level that will replaces the current national regulations coming from the Data Protection Directive [44]. This General Data Protection Regulation will take effect on 25 May 2018 [44].

The new rules in this regulation will strengthen the existing rights of consumers [45]. These include: easier access to your own data, a right to data portability, a clarified right to be forgotten and the right to know when your data has been hacked [45]. The new rules aim to make it easier for companies to do business in the EU by establishing a single set of rules and having only one supervisory authority [45]. Additionally all companies offering services in the EU, even non-EU-based ones, will have to apply to the same rules [45].

Acknowledgements

I would like to thank my supervisors, Bart Jacobs en Carlo Meijer, for helping me with my research. Especially Carlo for his help and enthusiasm and for retrieving the authentication key from the Android application. Furthermore thanks to Benoît Allard for his work on the Galileo project and for the initial information he provided. I also thank my sister, Rosa L. Schellevis, and my mother, Victoria Schellevis-Mol, and Jasper P. Ko for supporting me and providing feedback on the text my report.

References

[1] *Manual Fitbit Charge HR*. v1.2. Fitbit. URL: https://staticcs.fitbit.com/content/assets/help/manuals/manual_charge_hr_en_US.pdf.

[2] Britt Cyr et al. "Security Analysis of Wearable Fitness Devices (Fitbit)". Massachusetts Institute of Technology. 2014. URL: http://courses.csail.mit.edu/6.857/2014/files/17-cyrbritt-webbhorn-specter-dmiao-hacking-fitbit.pdf.

[3] Madhusudan Banik Mahmudur Rahman Bogdan Carbunar. "Fit and Vulnerable: Attacks and Defenses for a Health Monitoring Device". 2013. URL: http://arxiv.org/abs/1304.5672.

[4] *What is Fitbit Premium?* Fitbit Help. URL: https://help.fitbit.com/articles/en_US/Help_article/1365 (visited on 08/15/2016).

[5] Council of the European Union European Parliament. "Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data". In: *Official Journal of the EC* 281 (1995). URL: http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:31995L0046&rid=3.

[6] Article 29 Data Protection Working Party. *Advice paper on special categories of data ("sensitive data")*. Advice Paper. 2011. URL: http://ec.europa.eu/justice/data-protection/article-29/documentation/other-document/files/2011/2011_04_20_letter_artwp_mme_le_bail_directive_9546ec_annex1_en.pdf.

[7] Council of the European Union European Parliament. "Directive 2009/24/EC of the European Parliament and of the Council of 23 April 2009 on the legal protection of computer programs (Codified version) (Text with EEA relevance)". In: *Official Journal of the EC* 111 (2009). URL: http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32009L0024&rid=1.

[8] *mitmproxy*. URL: https://mitmproxy.org/.

[9] *How mitmproxy works mitmproxy 0.17.1 documentation*. URL: http://docs.mitmproxy.org/en/stable/howmitmproxy.html#the-mitm-in-mitmproxy (visited on 08/16/2016).

[10] Benoît Allard. *Galileo*. URL: https://bitbucket.org/benallard/galileo.

[11] *benallard / galileo / wiki / Megadumpformat - Bitbucket.* URL: https://bitbucket.org/benallard/galileo/wiki/Megadumpformat (visited on 08/15/2016).

[12] *Deep Dive Teardown of the Fitbit Charge HR FB405BKS-EU - Bluetooth 4.0 / NFC Sync Fitness Watch.* URL: http://www.techinsights.com/reports-and-subscriptions/open-market-reports/Report-Profile/?ReportKey=10707 (visited on 08/15/2016).

[13] *Ultra-low-power 32-bit MCU.* STM32L151xD and STM32L152xD. Rev. 10.0. STMicroelectronics. 2015. URL: http://www.st.com/resource/en/datasheet/stm32l151rd.pdf.

[14] *Cortex-M3 Processor - ARM.* URL: http://www.arm.com/products/processors/cortex-m/cortex-m3.php (visited on 08/15/2016).

[15] *IDA Pro.* URL: https://www.hex-rays.com/products/ida/.

[16] NIST FIPS Pub. "197: Advanced encryption standard (AES)". In: *Federal Information Processing Standards Publication* 197 (2001), pp. 441–0311. URL: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[17] Roger M Needham and David J Wheeler. "Tea extensions". In: *Report, Cambridge University, Cambridge, UK (October 1997)* (1997). URL: http://www.club.cc.cmu.edu/~ajo/docs/xtea.pdf.

[18] *What's changed in the latest tracker update?* Fitbit Help. URL: https://help.fitbit.com/articles/en_US/Help_article/1372 (visited on 08/16/2016).

[19] *How do I interpret the severity of a Fitbit security update?* Fitbit Help. URL: https://help.fitbit.com/articles/en_US/Help_article/2016 (visited on 08/16/2016).

[20] *benallard / galileo / wiki / Communicationprotocol - Bitbucket.* URL: https://bitbucket.org/benallard/galileo/wiki/Communicationprotocol (visited on 08/15/2016).

[21] *Reverse Engineering Fitbit BLE Protocol.* URL: https://pewpewthespells.com/blog/fitbit_re.html (visited on 08/15/2016).

[22] *Restore unencrypted logs - galileo - FreeLists.* URL: http://www.freelists.org/post/galileo/Restore-unencrypted-logs (visited on 08/15/2016).

[23] *JD-GUI.* URL: http://jd.benow.ca/.

[24] *Apktool*. URL: https://ibotpeaches.github.io/Apktool/.

[25] *dex2jar*. URL: https://sourceforge.net/projects/dex2jar/.

[26] Morris J Dworkin. "SP 800-38B". In: *Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication, National Institute of Standards & Technology, Gaithersburg, MD* (2005). URL: http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf.

[27] *Antitrust: Overview Competition - European Commission*. URL: http://ec.europa.eu/competition/antitrust/procedures_102_en.html (visited on 08/16/2016).

[28] "CONSOLIDATED VERSION OF THE TREATY ON THE FUNCTIONING OF THE EUROPEAN UNION". In: *Official Journal of the European Union* (2012). URL: http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:12012E/TXT&from=EN.

[29] Council of the European Union European Parliament. "Regulation 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)". In: *Official Journal of the European Union* (2016). URL: http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679&from=EN.

[30] Andrew Hilts, Christopher Parsons, and Jeffrey Knockel. *Every Step You Fake: A Comparative Analysis of Fitness Tracker Privacy and Security*. Open Effect Report, 2016. URL: https://openeffect.ca/reports/Every_Step_You_Fake.pdf.

[31] Martin Woolley. *Bluetooth Technology Protecting Your Privacy*. Bluetooth Blog. Apr. 2, 2015. URL: http://blog.bluetooth.com/bluetooth-technology-protecting-your-privacy/ (visited on 08/16/2016).

[32] *hiptopjones/fitbit*. GitHub. URL: https://github.com/hiptopjones/fitbit (visited on 08/17/2016).

[33] *Fitbit Privacy Policy*. URL: https://www.fitbit.com/nl/legal/privacy (visited on 08/16/2016).

[34] European Commission. "2000/520/EC: Commission Decision of 26 July 2000 pursuant to Directive 95/46/EC of the European Parliament and of the Council on the adequacy of the protection provided by the safe harbour privacy principles and related frequently asked questions issued by the US Department of Commerce (notified under document number C(2000) 2441) (Text with EEA relevance.)" In: *Official Journal of the EC* (2000). URL: http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32000D0520&rid=1.

[35] *European Commission - PRESS RELEASES - Press release - European Commission launches EU-U.S. Privacy Shield: stronger protection for transatlantic data flows.* URL: http://europa.eu/rapid/press-release_IP-16-2461_en.htm (visited on 08/15/2016).

[36] Article 29 Data Protection Working Party. *Article 29 Working Party Statement on the decision of the European Commission on the EU-U.S. Privacy Shield.* 2016. URL: http://ec.europa.eu/justice/data-protection/article-29/press-material/press-release/art29_press_material/2016/20160726_wp29_wp_statement_eu_us_privacy_shield_en.pdf.

[37] *Taken en bevoegdheden.* URL: https://autoriteitpersoonsgegevens.nl/nl/over-de-autoriteit-persoonsgegevens/taken-en-bevoegdheden (visited on 08/15/2016).

[38] *Wbp Meldingenregister.* URL: https://www.collegebeschermingpersoonsgegevens.nl/asp/orsearch.asp (visited on 08/15/2016).

[39] *Nike modifies running app after Dutch DPA investigation.* URL: https://autoriteitpersoonsgegevens.nl/en/news/nike-modifies-running-app-after-dutch-dpa-investigation (visited on 08/15/2016).

[40] *Onderzoek naar de verwerking van persoonsgegevens in het kader van de Nike+ Running app door Nike Inc.* College bescherming persoonsgegevens, 2015. URL: https://autoriteitpersoonsgegevens.nl/sites/default/files/atoms/files/onderzoek_nike_running_app_november_2015_1.pdf.

[41] Article 29 Data Protection Working Party. *Opinion 15/2011 on the definition of consent.* 2011. URL: http://ec.europa.eu/justice/data-protection/article-29/documentation/opinion-recommendation/files/2011/wp187_en.pdf.

[42] Article 29 Data Protection Working Party. *Working Document on the processing of personal data relating to health in electronic health records (EHR)*. Feb. 15, 2007. URL: http://ec.europa.eu/justice/data-protection/article-29/documentation/opinion-recommendation/files/2007/wp131_en.pdf.

[43] *Product Specifications Page*. URL: https://www.fitbit.com/comparison/trackers (visited on 08/16/2016).

[44] *Reform of EU data protection rules - European Commission*. URL: http://ec.europa.eu/justice/data-protection/reform/index_en.htm (visited on 08/15/2016).

[45] *European Commission - PRESS RELEASES - Press release - Agreement on Commission's EU data protection reform will boost Digital Single Market*. URL: http://europa.eu/rapid/press-release_IP-15-6321_en.htm (visited on 08/16/2016).