How can data from fitness trackers be obtained and analyzed with a forensic approach?

1st Florian Hantke
Dept. Computer Science
Friedrich-Alexander-Universität Erlangen-Nürnberg
Erlangen, Germany
florian.hantke@fau.de

2nd Andreas Dewald
Incident Response Division
ERNW Research GmbH
Heidelberg, Germany
adewald@ernw.de

***Abstract*—The use of Internet of Things devices is continuously increasing: People buy devices to make their lives more comfortable by using smart assistants or track sports activities and assess them. Moreover, these devices can support digital investigators with valuable information when it is involved in a crime scene, since its data may provide information about the circumstances of the crime. One group of those devices are fitness trackers, which hold data such as walked steps. Accordingly, analysts can see activities, routines, and inconsistencies. We inspected three different common fitness trackers and developed a tool to analyze them in a standardized and forensically sound way. To collect data, we analyzed the Bluetooth communication, data on the phone, and internet communication. Our tool can analyze the different sources automatically and subsequently presents the results on a self-hosted web application. It is open-source and easily scalable so that developers can implement new extensions to support more than the three analyzed trackers.**

***Index Terms*—Digital Forensics, Internet of Things, Fitness Tracker, Bluetooth, Android, Network**

1. Introduction

Every fifth american wears a device to track their fitness, according to the Pew Research Center [1]. Also in germany, the number of users is increasing every year [2]. These devices collect data, such as GPS, movement, or pulse, and can be used to monitor the user's activity. But not only to track the user's health, the collected fitness data can be used, but also to solve crimes, as various articles show. CNN reported that data obtained from a fitness band was used to reconstruct a timeline of a crime and refute the testimony of the suspect [3]. In another case, data from a Fitbit helped determining the time of death as they showed how the heart rate of the victim spiked significantly, followed by slowly decreasing until it stopped [4].

We presume that the potential of these devices with regard to crime-solving has so far been underestimated. Most approaches found on the internet are not well documented or unreliable. Neither a general approach on how to analyze fitness trackers nor a reliable tool which can analyze fitness trackers in a forensic environment was found. This is why we try to answer the question *Can a program be implemented to investigate common fitness trackers in a standardized and forensically sound way and subsequently view routines and activities of the tracker's user*? This work proposes to research different fitness trackers and gives a general approach on how to analyze them in a forensic way. Based on this knowledge a program is developed, which can use the different methods and extracts and reports the trackers' data. This tool is Open Source and well documented, thus everyone can reproduce the process. In addition, the tool is easily scalable so that people can add more trackers to extend the program.

1.1. Contribution

We present an open-source forensic tool to analyze the Xiaomi Mi Band 2, Fitbit Charge 2 and Huawei Band 2 Pro dynamically in order to view and compare the daily routines and activities of the user and to show inconsistencies.

1.2. Outline

In the next section, we start with a brief look at related work. In section 2 the results of the device analysis are presented. After that, we introduce in section 3 the developed fitness tracker analysis tool, which is evaluated in section 4 afterward. In section 6, we propose our general approach how to analyze fitness trackers forensically and finally summarize and conclude our work in section 5.

1.3. Related Work

Most of the forensics work in this field researches smartwatches. For instance, Baggili et al. analyzed the Galaxy Gear 2 Neo and the LG G and found that both devices, rooted, contained fitness data [5]. However, our goal was to inspect trackers without a Linux based OS and found no forensic motivated work in this regard, but only security motivated. *Breaking Fitness Records without Moving: Reverse Engineering and Spoofing Fitbit* presents an in-depth security analysis of the Fitbit Flex and Fitbit One [6]. The traffic to the cloud and also the Bluetooth communication was analyzed. In this manner, they found a way to request and manipulate various dumps from the tracker including status information and health data. Since

DOI 10.1109/EuroS&PW51379.2020.00072

it was a security issue, Fitbit fixed it and the traffic is encrypted in the latest version. In our work the attempt is to find reliable ways to request data.

In 2016 a team from Canada published work on nine fitness trackers and smartwatches considering their security [7]. Again, they intercepted the communication with the smartphone and the cloud. Afterwards, they compared the found issues and test person's experience with the company's Privacy Policies and Terms of Service to design proposals for IoT companies regarding their Privacy Policies. The approach they used to analyze the devices is similar to our strategy. They used their findings to design policy proposals. We, on the other hand, used our results to develop a forensic tool with the purpose to analyze fitness trackers.

2. Device Analysis

In this work, the term fitness tracker refers to wearable wristbands which are designed to track the user's fitness and activities. They have a small memory capacity and without a remote device, such as an Android phone, they provide just basic functions. It is important to note that this work does not discuss smartwatches, which are wearable watches with a broadly developed OS and the option to install apps.

First, we inspected the Bluetooth Low Energy (BLE) communication for each tracker and the Generic Attributes (GATT) [8] of the it via logs on our rooted Samsung Galaxy S4 and bluepy [9]. Secondly, we analyzed the data stored on the phone via Android Debug Bridge (ADB) [10] - first as normal user, later as root. Lastly, we studied the phones' network traffic by using Burp Suite [11] as a proxy with Burp's CA certificate. The focus was set on GPS, step count, heart rate, acceleration, sleeping hours, account information, such as the profile picture, and timestamps to have a fixed scope. We considered this data to be valuable for an investigation as they can help to understand the course of a crime. In the following sections, we present the individual findings.

2.1. Xiaomi Mi Band 2

Xiaomi, a startup founded 2010 in China reached a market share of 14.6% placing it among the top five wearable companies in 2019, according to the International Data Corporation [12]. Their Mi Band 2 uses its sensors to measure heart rate, count steps, and detect sleep phases, which can be viewed in the Mi Fit application. In contrary to the other two trackers, the Mi Band 2 mostly measures the heart rate on demand. Only if the user triggers an activity, such as running or biking, the heart rate is measured live. Also, if activated in the options, the heart rate is measured every two minutes during sleep.

2.1.1. Tracker device. To communicate via BLE with the Mi Band 2 the user first needs to authenticate before accessing previously recorded data. This was researched by Andrey Nikisaev and the detailed protocol works as shown below [13]:

1) Enable notifications by sending `\x01\x00` to descriptor with handle `0x2902`
2) Send `\x01\x00` + 16 byte key (same for all devices) to authentication characteristic `00000009-0000-3512-2118-0009af100700`
3) Request a random string by sending `\x02\x00` to authentication characteristic
4) Receive a random 16 byte string
5) Send `\x03\x00` + string, encrypted with the initial key *(AES with ECB)* back to authentication characteristic.

Being authenticated, the user can request previously recorded data for up to two weeks. The only way to delete this data is a factory reset in the app, else the data remains on the band even with a new connection. We found a way to receive the data from the band without a phone. The BLE protocol requires enabling the notification of two characteristics initially. Then, we send a timestamp for the requested time. The band will send another timestamp from which it sends data. This is due to the fact that the band cannot store data for a very long time because of the small amount of memory. To answer this, we send `0x02` and subsequently receive data packets from the timestamp until now. A step by step of the protocol looks as follows:

1) *Packet 1 & 2* - Enable notification by activating of the activity characteristic `00000005-0000-3512-2118-0009af100700` and fetch characteristic `00000004-0000-3512-2118-0009af100700`
2) *Packet 3* - Send timestamp to the fetch characteristic in the following format:
`\x01\x01<year><month><day><hour><minute>\x00\x08`
year = `\xe2\x07`; month = `\x06`; day = `\x14`; hour = `\x15`; minute = `\x14`
3) *Packet 4* - Receive `\x10\x01\x01` followed by the actual timestamp
4) *Packet 5* - Send `\x02` to the fetch characteristic
5) *Packet 6 and following* - Receive data packets

Each of the data packets can be split into four data items, where one item equals one minute. The items consist of the index, activity category, acceleration, steps and heart rate. If the Mi Band 2 measured no heart rate, it stores the values 0, 254 or 255. One example packet is shown in Table 1.

2.1.2. Smartphone. In the user area of out test phone, we found various directories related to Mi Fit application. The path `/sdcard/.miband/` included PNGs, which show the recorded path during an activity and their distance. Furthermore, `/sdcard/Android/data/com.xiaomi.hm.health/` contained the user's profile picture in `Millelet/TEMP_PHOTO`, the rest were cache files. Cache data can indicate activity with the app, however, we considered them not valuable for our scope.

The directory `/data/data/com.xiaomi.hm.health` in the root area contains much relevant information. We found one database in the directory `databases` with a great number of tables of which some were empty. This is due to the fact that we did not use every function of the app, such as connecting friends. The interesting tables are:

- `DATE_DATA` - Every row is a single day with a date and a summary of the user's fitness data.

TABLE 1. INTERPRETATION OF PREVIOUS RECORDED DATA.

Received data	Interpretation	Explanation
0x39	index of packet	Packet 57
0x11	category	Begin of one minute (57*4)+0
0x1b	acceleration	10%
0x07	steps	7 steps in this minute
0x3f	heart rate	63 bpm
0x60	category	Begin of one minute (57*4)+1
0x2b	acceleration	17%
0xf4	steps	20 steps in this minute
0x41	heart rate	65 bpm
0x60	category	Begin of one minute (57*4)+2
0x12	acceleration	7%
0x00	steps	No steps
0x4b	heart rate	75 beats per minute
0x50	category	Begin of one minute (57*4)+3
0x09	acceleration	4%
0x00	steps	No steps
0xff	heart rate	No measured heart rate

More interestingly, the column *DATA* includes a JSON with a *value* field. This field contains a byte string, which can be base64 decoded and results in a 4320 byte list. 4320 divided by three is 1440, the number of minutes of a day. Every third byte of the byte list can be interpreted as one minute starting from midnight incrementing by one with each group; the first byte seems to be an activity status, the second byte the acceleration, and the third byte the number of steps during the minute. Consequently, it is possible to determine the movement of the user accurate to a minute. Last, the column *DATA_HR* contains a 1440 elements long byte list, which can be interpreted as the heart rate for each minute of the day. If not measured the value is 0, 254 or 255. Heart rate measured during an activity is not included, but can be found in the `TRACKER_DATA` table.

- `DEVICE` - The table contains information about all devices, that were connected with the app, such as the time it was first connected and last synchronized.
- `HEART_RATE` - The table lists all on-demand measured heart rates. It provides a timestamp in Unix time and the id of the device which measured the heart rate as well as the measurement value.
- `MANUAL_DATA` - The table provides data the user added manually in the app, such as sleeping hours.
- `TRACKRECORD` and `TRACKDATA` - Joined together the tables hold information on activity routes, such as the date or distance. Further, we found the start position and the following position changes. An example can be seen in Table 2. Additionally, combining columns *BULKTIME* and *BULKPAUSE* we were able to reproduce the minutes passed since the last position measurement and activity breaks. *BULKHR* contains the pulse and the passed seconds since the last measurement.
- `USER_INFO` - The table provides information about the user, such as the birthday or an avatar.

In the directory *files* we found a log with already known information and *shared_pref* that hold tokens for the huami API, an interface used by the app to store data in a cloud. Finally, the *cache* directory contained the profile pictures, cached API calls and library requests. While we consider libraries as not interesting for our purpose, API calls can be useful for network analysis.

2.1.3. Network. As aforementioned, the Mi Fit application uses the Huami API to communicate with a cloud. On Github the developers recommend to email *partner@huami.com* to create an API account, however, they never answered our e-mail [14]. After we asked via Github, they replied and said that our request has been deprioritized [15].

By using Burp Suite to analyze the application's traffic, we found calls to Huami, Google APIs, and Facebook. Together with the potential API tokens from the *shared_pref* and the recorded API calls, we tried to connect to the Huami API. However, our attempts to get access to the API were denied.

2.2. Fitbit Charge 2

Fitbit reached a market share of 4.1% in 2019. Its Charge 2 measures continual heart rate, steps, walked floors and sleep phases. The heart rate is measured every five seconds in default mode and every second during workouts. To view detailed data the Fitbit app and an account are required. Besides that, Fitbit offers a public Web API which can be used with a registered API key.

2.2.1. Tracker device. As mentioned in related work, the Fitbit synchronizing protocol is already researched and works as seen in Appendix B. The tracker uses the smartphone as a transmitter to send an encrypted mega dump to the Fitbit servers. After the servers evaluated the data, the users can request and view them on their smartphone. We were able to reproduce this behavior and thus we were not able to extract any fitness information using only BLE.

2.2.2. Smartphone. In the user area of the phone we only found binary files that belong to the Fitbit app. The contained no typical magic bytes and we were not able to interpret the format. We found more information in the root area `/data/data/com.fitbit.FitbitMobile/`. In the *cache/* directory we identified Google Maps thumbnails of traveled routes. In addition, the directory holds several subdirectories of which only two contained readable files. The sub-directory *datacache/* held various information about setup and updates, only relevant for the user. Another sub-directory *httpcache/* contained cached network traffic, such as API calls and results, but mostly encrypted. Nevertheless, we found some JSON files which were not encrypted and readable. One of those files held step count data of every minute, obtained from an API call. However, since it is a cache file, it is not clear how reproducible the existence of such files is. During several tests, such a file rarely appeared in the cache files.

In the *databases* directory several interesting SQLite databases were found:

- `exercise_db` - This database holds various tables with data on the exercises. In table *EXERCISE_EVENT* we found timestamps with coordinates and more fitness information. Combined

TABLE 2. THE TABLE SHOWS AN EXAMPLE HOW POSITION DATA IN THE MI FIT DATABASE IS STORED

Data in database	Longitude	Latitude
4947223680, 873280544	49.47223680	8.73280544
381, -11825	49.47224061	8.73268719
0, -7438	49.47224061	8.73261281
-4196, 10204	49.47219865	8.73271485
-762 ,-762	49.47219103	8.73270723
-4959, 6198	49.47214144	8.73276921
-3433, 9536	49.47210711	8.73286457
-3433, 9727	49.47207278	8.73296184
-381, 8773	49.47206897	8.73304957
4196, 3051	49.47211093	8.73308008
0, -6961	49.47211093	8.73301047

with table *EXERCISE_SEGMENT*, which groups the events, we were able to reconstruct the path of an exercise. However, GPS information is only recorded with the phone connected to the internet.

- `fitbit-db` - This database offers information about the user's profile and devices. Furthermore, we discovered the table *TIME_SERIES_OBJECT*, which holds certain data types at a specific time, whether summarized for one day or of a 15-minute interval. The data types, as we interpreted them can be found in Appendix A. Unfortunately, both groups are not reliable as much data is missing.
- `heart_rate_db` - It contains the average heart rate for each day and further information about heart rate zones, such as *cardio* or *fat burn*.
- `sleep` - It holds information on the users' sleep cycle. The table *SLEEP_LOG* contains the start and duration about recorded sleep, as well as minutes spent asleep, minutes spent awake and more metadata. In addition, *SLEEP_LEVEL_DATA* and *SLEEP_LEVEL_DATA_SUMMARY* show exactly when and how often the user entered a sleep phase.

In the last two directories *files* and *shared_prefs* we found the user's e-mail address, a URL to the user's profile picture and information about the smartphone device model. Moreover, we discovered two files *oauth2_authinfo_credentials.json* and *0ADF345A1316trackerAuthCredentials.json*, which presumably contain an access token for the API and keys to communicate with the tracker. However, both files appear to be encrypted as we were not able to read them. We found no key in the data to decrypt them. One could reverse engineer the apk and search the code for a key, yet this was out of our scope.

2.2.3. Network. Fitbit offers a public Web API, which works with OAuth2. Unfortunately, all discovered credentials were encrypted so we used a new created account to test the API.

While most API calls worked without problems, some issues were encountered. First, the API offers a method to download the data of an activity as a *.tcx* file, including GPS, however it requires a log-id. This can be found in the Fitbit Web Interface. Although the documentation says different, we were not able to request this ID by only using the API. The next issue was that time series calls did not work properly and to obtain all intraday data we must request each day and data type individually. Due to Fitbits request limit, we are limited to request only 74 days per hour. Note that this limitation was not detected in the app.

We also analyzed cached files and found API calls that were not documented officially, yet most of them were deprecated. The only interesting call we found offers the time at which the user was sitting: *https://api.fitbit.com/1/user/[user-id]/sed/date/[date].json*.

2.3. Huawei Band 2 Pro

The last tracker we analyzed is the Band 2 Pro from Huawei, who reached a market share of 8.4% in 2019. The band measures the heart rate every 30 minutes in default mode and every second in sport mode. Furthermore, it measures steps and sleep phases and during sport mode the traveled distance and position via GPS. To connect the Band 2 Pro with the phone, Huawei recommends two apps: Huawei Health and Huawei Mobile Services. Huawei Health app is used to track the Band 2 Pro data and Huawei Mobile Services is needed for the Huawei ID, the required user account.

2.3.1. Tracker device. The analyzed Bluetooth log shows that the Band 2 Pro uses two characteristics to transmit data - `0x0039` to send and `0x003a` to receive data. In order to enable the communication, the client must activate notifications. The recorded messages consist of byte strings, where we didn't recognize a pattern, so we presume that the communication is encrypted. Further, in contrast to the two previous trackers, we were not able to connect the tracker with our laptop via Bluetooth as the tracker refused the connection. We tried to connect with the tracker in different states, connected to the phone, disconnected and after a reset, nevertheless we were not successful.

2.3.2. Smartphone. We analyzed both Huawei apps and describe first the Mobile Service app, followed by the Health app. In both apps, we found data in the user area which were unfortunately redacted and not valuable for our purpose. In the following we will only describe the root area.

Data from the Mobile Service app was found in the directory `/data/data/com.huawei.hwid/`. The directory *app_webview* contained a cookie database, which stores several expired session keys.

In the *databases* directory we discovered the *dns.db* database, that contains DNS entries to several Huawei services. Another large database goes by the name *sns_e_d3790ff2a4e4e80f094f955a8ab938a1.db*, however it

is encrypted. For this database we found related files in the shared preferences, as the names indicate: *sns_dbKey_d3790ff2a4e4e80f094f955a8ab938a1.xml; sns_secret.xml*; *snsSp_d3790ff2a4e4e80f094f955a8ab938a1.xml*. The secret file holds an IV Key, Secret Key, and Salt and the dbKey file holds a twice base64 decoded 64 bytes long strings. However, we tried to decrypt the database with this information and different standard decryption methods and none led to a result.

Next, we analyzed the files related to the Huawei Health app. We inspected the databases in directory *databases*. The file *Device.db* maintains the data of all connected devices with the time they were added and device addresses. The largest database in the directory is *hihealth_0003.db*, yet it is encrypted. The databases *com_huawei_health*.db*, where * stands for a unique number, are also encrypted. We assume that the number stands for the stored information type, as later in the log files the types have the same numbers. However, the databases are encrypted and no keys were found in the app data.

We found log files in the folder *com.huawei.health/* in four sub-directories. Each sub-directory, except for one, contained log files. *com.huawei.health_DaemonService/* appears to log a conversation with a server with information, such as the HTTP response status and the amount of sent and received data. The log shows that the class `HiH_HiHealthSaveData` stores several requested data types. The types 4002 and 4003 are the daily summarized counted steps and distance, as they match the data we see in the app. The log files in *com.huawei.health* store information on the users' interaction with the app. For instance, we found traces that the user requested counted steps to show them in a graph. When steps of a day were requested, the log provides all rendered data. Thus, we are able to parse all steps, which were viewed by the user. In Figure 1 an example log is shown in which we can see the values *step* and timestamps. However, only the step, calories, and distance data were found, no positions or heart rate.

Figure 1. The Huawei log shows data which were requested by the user

```
2018727-11:11:19:779|SCUI_FitnessDetailInteractor
    |80002323|getFitnessDataDetail datas=[HiHealthData{
    type=4, day = 2018-07-26 07:59:00, values = step
    =74.0 end_time=1532584800000 trackdata_deviceType
    =47 client_id=5 altitude_offset=0.0 start_time
    =1532584740000 calorie=1815.0 distance=60.0},
    HiHealthData{type=4, day = 2018-07-26 10:02:00,
    values = step=184.0 end_time=1532592180000
    trackdata_deviceType=47 client_id=5 altitude_offset
    =0.0 start
2018727-11:11:19:779|SCUI_FitnessDetailInteractor|_time
    =1532592120000 calorie=5201.0 distance=132.0},
    HiHealthData{type=4, day = 2018-07-26 10:42:00,
    values = step=49.0 end_time=1532594580000
    trackdata_deviceType=47 client_id=5 altitude_offset
    =0.0 start_time=1532594520000 calorie=1003.0
    distance=32.0},
...
2018727-11:11:19:779|SCUI_FitnessDetailInteractor|
    ltitude_offset=0.0 start_time=1532618220000 calorie
    =7249.0 distance=226.0}, HiHealthData{type=4, day =
     2018-07-26 18:31:00, values = step=143.0 end_time
    =1532622720000 trackdata_deviceType=47 client_id=5
    altitude_offset=0.0 start_time=1532622660000
    calorie=4416.0 distance=93.0}]
```

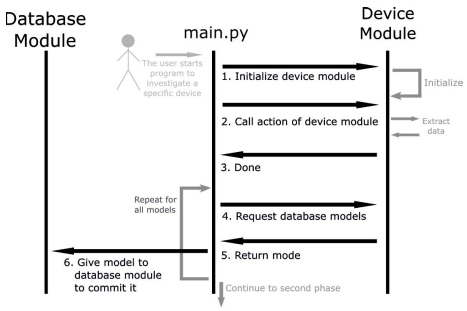

Figure 2. The figure shows the program flow of the first phase

2.3.3. Network. In the log files we discovered that the Health app communicates with *https://hicloud.com*. Unfortunately, Burp Suite recorded no communication belonging to the Huawei apps. The explanation for this behavior was found in the Health log file where an exception said: "don't use proxy". To bypass this challenge one could reverse the app, change the code and recompile it. This was out of our scope and so we tried various standard requests to the API which all led to a dead end.

3. Implementation

In the previous section we presented the data that we found on the fitness trackers and the other sources. Now, we introduce the tool[1] that we developed on the basis of these findings to investigate the trackers.

The tool is programmed in Python 3 and all requirements are listed in *requirements.txt*. The analysts can start the tool by specifying the device and the method they want to use. For instance, one could start `python3 main.py --device MiBand2 --method app` to extract and afterward analyze the data which is stored on the phone in question. The tool is divided into two phases: the *Extracting Phase* and the *Presenting Phase* that apply the appropriate aforementioned methods for the specified devices.

Figure 2 demonstrates the first phase. In the first phase the tool initializes the device module for the requested device and method. This module is different for each device and method combination. It obtains the data from the selected source and verifies the correctness of the data (if possible). Next, it analyzes and interprets them and stores them in a SQLite database. The device modules are independent, so that anyone can develop further modules in future work. For that purpose, the modules are well documented and easy to understand.

In the second phase, the results are presented as an interactive HTML report implemented in the report module by using Flask. An example is shown in 3 and in more detail in appendix C. The investigator can choose a source and view different reports:

- *User Report* - Here the analyst can see user data, for instance, the profile picture or the name that was collected from the API or the app. Thus, they can quickly check whether it is the suspected person who owns and uses the device.

1. https://github.com/FHantke/FitnessTrackerAnalysisTool

- *Data Report (3)* - Here the analyst can see information about the heart rate, step count, acceleration and sleep hours in diagrams. Since the amount of data can easily reach hundreds of thousands of records, if measured every second, which is at the expense of render speed, we decided that the investigator can choose a time range of interest and the accuracy of the chart. In addition to the main chart, a calendar shows the step activity over the course of one year. Steps were chosen, since, in the initial research, the step data was most reliable in terms of availability and visibility of activity. Moreover, the user's sleeping hours are plotted in a box-plot diagram, which shows the distribution of the start and the end of the sleep. Accordingly, the investigator can deduce a sleep rhythm and recognize inconsistencies. After the charts have rendered, the user can easily zoom in, view more specific time ranges or export the current chart view as PDF or SVG.
- *Map Report* - Here the analyst can see the recorded GPS routes on a Google Maps view. They can select one route to get a detailed view with more information on every single recorded coordinate. For instance, they can see the timestamp and heart rate for a specific point in a track. Further, it is possible to use Street View and follow the path of the user from the first-person perspective.
- *Compare Report* - Here the analyst can compare data from selected sanalyses to make similarities and differences of the data visible.
- *Custom Attributes Report* - Every device is different, thus we decided to give developers the opportunity to report specific information about the device.
- *Logs* - In the log view the various logs can be searched and sorted. They contain the process of the investigation with timestamps, as well as problems and errors.

Figure 3. The activity data page of the report displays fitness data on a timeline. Additionally, it shows active days and the average sleeping hours.

4. Evaluation

The evaluation is based on data we collected using each tracker for one month. We saved the data shown in the apps for later use as control data. First, we evaluated the forensic soundness of the tool by considering the four criteria defined by Rodney McKemmish: Meaning; Errors; Transparency; Experience. Furthermore, completeness, as McKemmish calls it one of the two most critical properties, is considered separately [16]. Secondly, we checked the robustness of the tool.

To maintain the Meaning, the report module does not change data and the device modules may only interpret data while storing the data. For instance, raw byte values are represented and stored as a decimal number or an ASCII character. Further, our tool checks file hashes, when downloading data from a phone before and after the execution to avoid the possibility of broken data. In case of a wrong hash, it would restart the download. When working with APIs, SSL certificates are verified to ensure integrity, which is part of meaning. Assuring integrity, of course, means also to follow a solid chain of custody. Every interaction with the devices must be documented precisely. Errors that occur during an investigation need to be identified and clarified as McKemmish says. Thus every exception in our tool is well documented and displayed in the logs. To be as transparent as possible, our entire tool is made open source. Also, every step of the tool is reported in the log so that the processes can easily be reproduced. The last criterion, which is called Experience of an investigator, cannot be influenced by this tool. It is the task of the analysts to judge themselves correctly. Nevertheless, we tried to make the reports as structured as possible so that even an unqualified person can see and understand correlations.

Beyond the immutability of evidence and completeness of documentation, we also understood by completeness that one should consider whether the tool extracts and reports all data in the scope of this work. We compared the data our tool collected with the information shown in the Android apps. Table 3 shows the evaluation of the completeness of the different devices.

The question is now, is the tool is forensically sound? We use several techniques and methods to maintain reliability, so that one part of forensic soundness is guaranteed. The second part, completeness, was considered for every device and its methods individually. There, it can be seen that most methods do not obtain all the information on which the focus was set and sometimes not even all of the data, but just for a period of time. Nevertheless, the found data in this period of time is complete and correct as several checks with our collected data show. As a result, we consider the tool forensically sound as long as the investigators know what data and what period of time were obtained. This is displayed in the report.

Next, we considered the robustness of the tool. First, we tested the normal use with data collected during the entire research. The tool works as intended - it extracts the data and displays it correctly. When requesting too many data, the tool takes a few seconds to execute, but always provides the report eventually. Afterwards, we used the program incorrectly and attempted to force different failures or data manipulation. With our methods, we covered

TABLE 3. EVALUATION OF COMPLETENESS OF DEVICE MODULES

Device	Method	Found data	Completeness
Xiaomi	BLE	Steps	Only the last seven days
		Heart Rate	Only the last seven days
		Acceleration	Only the last seven days
Xiaomi	App	GPS	Yes, if the phone is rooted
		Steps	Yes, if the phone is rooted
		Heart Rate	Yes, if the phone is rooted
		Acceleration	Yes, if the phone is rooted
		Sleeping hours	Yes, if the phone is rooted
		Account data	Yes
Fitbit	App	GPS	Yes, if the phone is rooted
		Steps	Yes, if the phone is rooted
		Account data	Yes, if the phone is rooted
Fitbit	Cloud	Steps	Yes, with user credentials
		Heart Rate	Yes, with user credentials
		Sleeping hours	Yes, with user credentials
		Account data	Yes, with user credentials
Huawei	App	Steps	If the user requested data

a wide spectrum of issues, such as changing the database or failed imports. The results show, the program reacts every time as it was intended to and reports the incorrect behavior. In conclusion, it can be said that the program passed all the tests, even if it can become slow when the amount of data is too large.

5. Proposed Method for Fitness Tracker Forensics

As we mentioned earlier, our motivation for this work was that we believe fitness trackers can provide great information about a crime scene. To get the best result, professional and careful handling and a solid chain of custody is important. Thus, we propose to divide the procedure into four steps: Preparation, Extraction, Analysis and Reporting. Investigators should also keep in mind that the user could falsify data, for instance by giving the tracker to a friend. The collected data should never be used by its own. The following sections describe the steps and clarify how our tool can be used for this purpose.

5.1. Preparation

When a fitness tracker is involved as a potential witness, we recommend to put the phone and the tracker into separate Faraday bags to block further communication. Otherwise, a tracker could send mistakenly collected data to the phone and show activity, even after the seizure. This could lead to inconsistent data during the analysis. Also, every interaction with the devices from now on must be documented for the final report and the chain of custody. In the same way, investigators must take the law and regulation in account on which the analysis is based. Since a profile picture or movement of a person is private information, special care must be given to comply with the law. Furthermore, it may be possible to request user credentials for the fitness tracker account with the help of a warrant for the purpose of extracting information from an API.

5.2. Extraction

In this step, the analyst should use our tool to extract information from the different sources. Due to volatile memory, the tracker should be processed first and as soon as possible. For this, the analyst must connect the tracker via Bluetooth to a workstation and run our tool. Next, the investigator should analyze the phone. Therefore, the phone must be connected to the workstation via USB and our tool extracts the data. To extract valuable information from a phone, we have seen that the phone must be rooted for most trackers. The rooting process can cause problems as discussed in paper from Vidas et al. [17]. Rooting a phone often is accomplished by using a software flaw in the phone's Android version. This generally changes internal data as it installs tools with extended privileges and brings with it the chance of damaging the device. Modifying the data is at the expense of integrity and can lead to refusal of the evidence extracted from the rooted phone in court. Thus, we suggest to first copy all possible evidence from the unrooted phone, before starting the rooting process. If the tracker's manufacturer offers an API it can also be used with our tool, however our research showed that access to the tracker users account is needed. This can be done with the permission and cooperation of the tracker owner, for instance, if the owners innocence is to be proven.

5.3. Analysis

During the analysis, the investigator tries to answer the initial questions of the investigation by interpreting the collected data. They can use the different views and graphs in our tool for this purpose. The analyst should cross-check and synchronize all the findings in the different sources, for instance, the phone and the tracker. The findings can show differences in the data which should be considered. This can, for example, happen when the tracker was not connected to the phone when it was confiscated.

5.4. Reporting

Reporting is relevant to demonstrate that the findings are accurate and to present them in an understandable way. Therefore, the investigator should extract the graphs from our tool and use them in the report. In the end, the report must also hold all interactions with the devices to justify the decisions and results.

6. Conclusion & Future Work

We analyzed three common fitness trackers to find data which could support court hearings with valuable information. On the basis of these results we developed a tool to investigate the different sources and proposed an approach to follow.

However, our methods that we used have a few limitations. First, we analyzed only three trackers while there exist many more on the market. For some trackers, we didn't find all the requested information, mostly due to encryption or the absence of this information. Further, to obtain valuable information from an Android phone it must be rooted.

To build upon this work, one could analyze more trackers and other operating systems to extend our tool, as

it was intended. On the other hand, one could reverse the apps to break the encryption and cloud communications and add the functionality to our tool. With this method, a researchers may also overcome the limitation of rooting a phone as the communication with a cloud, if read only, does not modify the evidence. As it can be seen, plenty of future work is possible in this field and the potential for growth is promising.

Although the tool has limitations, we think the resulting graphs and tables can be used in forensic work, as we evaluated it to be forensically sound. We believe that this work gives a good idea of how beneficial the analysis of fitness trackers for judicature procedure can be and that we will see them more often in legal cases in the future.

References

[1] Pew Research Center, "About one-in-five Americans use a smart watch or fitness tracker," 2020, https://www.pewresearch.org/fact-tank/2020/01/09/about-one-in-five-americans-use-a-smart-watch-or-fitness-tracker/, Accessed: 2020-04-19.

[2] Bitkom, "Zukunft der Consumer Technology - 2019," 2019, https://www.bitkom.org/sites/default/files/2019-09/190903_ct_studie_2019_online.pdf, Accessed: 2020-04-19.

[3] A. Watts, "Cops use murdered woman's Fitbit to charge her husband," CNN, 2017, https://edition.cnn.com/2017/04/25/us/fitbit-womans-death-investigation-trnd/index.html, Accessed: 2020-04-19.

[4] C. Hauser, "Police Use Fitbit Data to Charge 90-Year-Old Man in Stepdaughter's Killing," The New York Times, 2018, https://www.nytimes.com/2018/10/03/us/fitbit-murder-arrest.html, Accessed: 2020-04-24.

[5] I. Baggili and J. Oduro and K. Anthony and F. Breitinger and G. McGee, "Watch What You Wear: Preliminary Forensic Analysis of Smart Watches," 10th International Conference on Availability, Reliability and Security, 2015, pp. 303–311.

[6] H. Fereidooni and J. Classen and T. Spink and P. Patras and M. Miettinen and A. R. Sadeghi and M. Hollick and M. Conti, "Breaking Fitness Records without Moving: Reverse Engineering and Spoofing Fitbit," CoRR, 2017.

[7] A. Hilts and C. Parsons and J. Knockel, "Every Step You Fake: A Comparative Analysis of Fitness Tracker Privacy and Security," Open Effect Report, 2016.

[8] Bluetooth SIG, GATT Specifications, https://www.bluetooth.com/specifications/gatt, Accessed: 2020-02-15.

[9] IanHarvey, Python interface to Bluetooth LE on Linux, https://github.com/IanHarvey/bluepy, Accessed: 2020-02-15.

[10] Google Developers, Android Debug Bridge, https://developer.android.com/studio/command-line/adb, Accessed: 2020-02-15.

[11] PortSwigger Ltd., Burp Suite, https://portswigger.net/burp, Accessed: 2020-02-15.

[12] International Data Corporation, "Worldwide Wearables Shipments Surge 94.6% in 3Q 2019 Led by Expanding Hearables Market, Says IDC," 2019, https://www.idc.com/getdoc.jsp?containerId=prUS45712619, Accessed: 2020-04-19.

[13] A. Nikishaev, "How I hacked my Xiaomi MiBand 2 fitenss tracker - a step-by-step Linux guide," Medium, 2018, https://medium.com/machine-learning-world/how-i-hacked-xiaomi-miband-2-to-control-it-from-linux-a5bd2f36d3ad, Accessed: 2020-04-19.

[14] Huamitech, Huami Web API Documents for Developers, https://github.com/huamitech/rest-api/wiki, Accessed: 2020-02-15.

[15] Jimzhanghm, Application Registration, https://github.com/huamitech/rest-api/issues/2, Accessed: 2020-02-15.

[16] R. McKemmish , "When is Digital Evidence Forensically Sound?," Advances in Digital Forensics IV, Springer US, 2008, pp. 3–15.

[17] T. Vidas and C. Zhang and N. Christin, "Toward a general collection methodology for Android devices," Digital Investigation, 2011.

Appendix

1. Fitbit Charge 2 - Protocol

The Fitbit Charge 2 synchronizes its data first with the cloud. Only then, the app can handle this data by requesting it from the cloud.

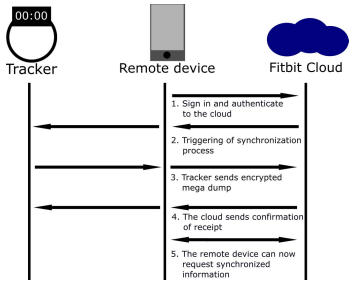

Figure 4. The figure indicates the synchronization protocol of Fitbit

2. Fitbit Charge 2 - Object Types

The Fitbit database holds various object types, which can be interpreted. Table 4 shows the interpretation of these object types and whether they contain the correct value or not. Not all of object types are reliable.

TABLE 4. THE TABLES SHOWS INTERPRETED OBJECT TYPES AND THEIR CORRECTNESS

Type	Meaning	Correct Values
1	Weight summary of one day	False
3	Contains only 0.0 for each day	False
4	Step count summary of one day	False
5	Calorie summary of one day	False
6	Floor summary of one day	False
8	Distance summary of one day	False
11	Intraday step count every 15 minutes	True
12	Intraday calories every 15 minutes	True
16	Active minutes of one day	False
17	Intraday active minutes every 15 minutes	True
18	Intraday heart rate every 15 minutes	True
19	Average heart rate minutes of one day	True

3. Data Report

The figures show the different graphs one can analyze in the data report.

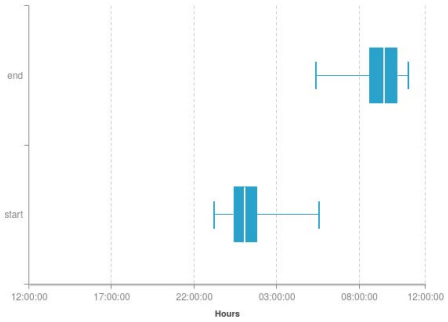

Figure 5. The figure shows the users average sleeping hours

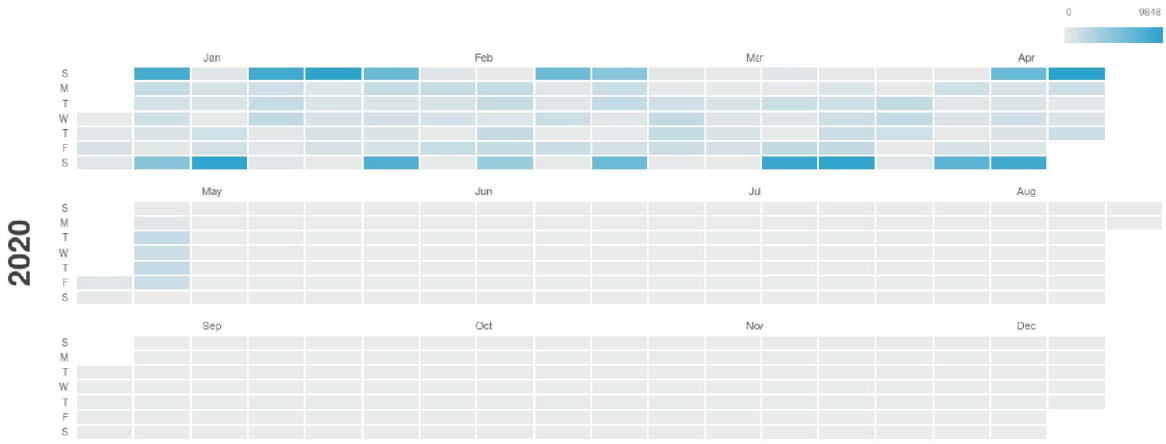

Figure 6. The calender shows the activity of the user over the course of one year

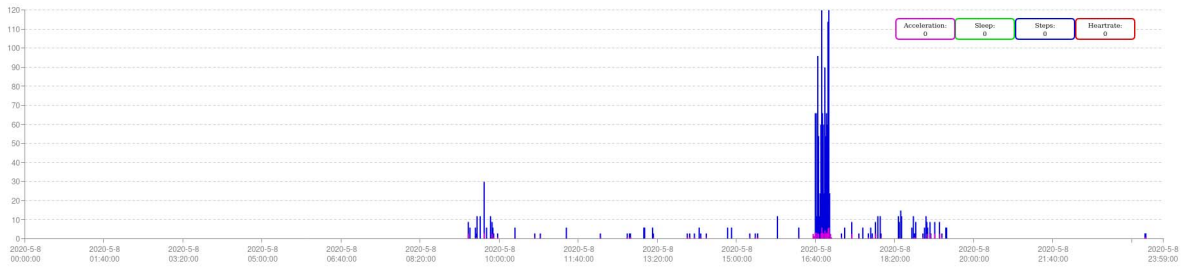

Figure 7. The graph shows the collected data over time