I still See You! Inferring Fitness Data from Encrypted Traffic of Wearables

Andrei Kazlouski[1,2], Thomas Marchioro[1,2], Harry Manifavas[1,2] and Evangelos Markatos[1,2]

[1]*Computer Science Department, University of Crete, Greece*

[2]*Institute of Computer Science, Foundation for Research and Technology, Greece*

Keywords: Wearables, Security, Privacy.

Abstract: In this paper we describe a cyberattack against 2 well-known wearable devices. The attacker presented in this paper is an "honest but curious" Internet Service Provider (ISP) sitting somewhere in the path between the device and the cloud. The ISP launches the attack when the smartbands try to synchronize their data with the permanent cloud storage. By launching its attack, this "honest but curious" ISP is able to learn much personal information about the users of the smartbands, including the frequency of measuring the users' heart rate and weight; the number and duration of workouts; as well as whether (i) sleep or (ii) steps were recorded on a given day. We show that privacy leaks might occur even when the transferred data are fully encrypted, and the representative mobile application utilizes state-of-the-art security mechanisms: certificate pinning, and source code obfuscation.

1 INTRODUCTION

Over the past few years there have been an impressive increase in the uptake and everyday use of wearable devices, such as wristbands and smartwatches. Worn by the end users 24 hours per day - seven days per week, these devices measure, collect, and communicate a wealth of health- and fitness-related information. As a result, consumers can find out how intensely they exercise and what is their progress over time. Although this information is collected on the end device (wristband), it is usually sent to the user's smartphone and eventually transferred to the permanent storage space in the cloud associated with the manufacturer of the wearable device. However, the information collected by those wearable devices is personal and may sometimes be deeply confidential. Indeed, a user's heart rate, sleep patterns, stress and oxygen saturation levels can be used to draw important conclusions about physical and mental health.

In this paper we explore whether private health data can be retrieved by undesirable third parties. Although there are several places where this information can be leaked, including (i) the wristband which is used by the end user, (ii) the smartphone where the app runs, or (iii) even the cloud where the data are stored, in this paper we focus on the communication link *between* the smartphone and the cloud.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 813162. The content of this paper reflects the views only of their author (s). The European Commission/ Research Executive Agency are not responsible for any use that may be made of the information it contains.

Threat Model. The threat model that we consider in this paper is as follows: we assume that the user has a wearable device. The device is connected to a smartphone via Bluetooth and the smartphone is connected to the cloud via the open Public Internet. We assume that one of the ISPs who links the user's smartphone to the cloud would like to find information about her. Such information may include whether the user owns a wristband, how often the user trains, what is the duration of the training sessions, etc. This ISP could be the first one that connects the user to the Internet, or even another ISP down the line in the path from the user's smartphone to the cloud. The ISP is assumed to be "honest but curious", i.e it will *not* try to actively attack the user by installing a "man-in-the-middle" (MITM) proxy or exploiting other vulnerabilities. The ISP will *honestly* just do its job, that is to deliver the user's IP packets to their destination. At the same time, however, the ISP may try to find as much information as possible from the received IP packets. Figure 1 displays a diagram of the threat model we consider. To our knowledge, our work is

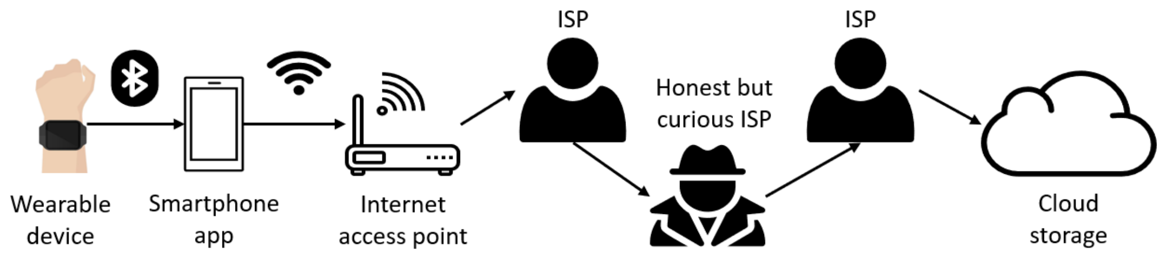

Figure 1: We assume that the smartband is connected to the smartphone which uploads the data on the cloud over the Internet - over one or more Internet Service Providers (ISPs). We assume that at least one of the ISPs is *honest but curious*; it accepts IP packets to be delivered to the cloud. It does *not* try to actively attack the user, but rather to *infer* information from traffic.

the first to do a successful *passive* attack in the domain of fitness trackers. We discuss prior attack models in section 2.

Motivation for the Attack. A valid question to ask would be: *Is it plausible that a major ISP would tarnish their reputation, and engage in mass profiling?*

We argue, however, that it might not be major ISPs who would be willing to perform the attack. Corporate health and wellness is growing increasingly popular lately. Lesser companies utilize wearables to boost the health of their employees [1]. In case when an organisation does not force workers to reveal their fitness data, they can still launch constant health surveillance of the employees, acting as a curious ISP. Besides, it is nearly impossible to detect a passive attacker who does not attempt to modify the data.

Contributions. The contributions of this paper can be summarized as follows:

- We present the architecture of a system that studies the information sent from the smartbands to the cloud.
- We demonstrate that it is possible for "honest but curious" ISPs to find information about the users' patterns and activities even when the communication between the smartphone and the cloud is encrypted.
- We show that encryption alone might not enough to protect user's privacy. We propose approaches that when combined with encryption can be used to preserve privacy in a much more effective way.

2 RELATED WORK

Security of fitness trackers has been widely an object of study. Several works have shown that even the most popular devices do not always apply state-of-the-art security mechanisms, enabling the attacker to exploit the lack of protection. Prior works have performed attacks on smartbands with an active adversary, and evaluated security of wearables.

Traffic Analysis of HTTPS. The idea of encrypted traffic leaking information has been around since the introduction of HTTPS. Traffic Analyses of the HTTP Protocol over TLS were conducted in (Cheng and Avnur, 1998; Danezis, 2009; Bissias et al., 2005). In (Chen et al., 2010; Zhang et al., 2010; Sun et al., 2002) authors presented side-channel leaks in web services. Hitz was able to fingerprint websites based on encrypted traffic (Hintz, 2002). Liberatore and Levine in turn were able to profile users of websites based on encrypted traffic (Liberatore and Levine, 2006). All these papers prove that encryption alone is often not enough to prevent possible privacy leaks.

Attacks on Smarbands. A number of previous works demonstrated successful **active** attacks on various wristwatches exploiting different vulnerabilities (Zhang and Liang, 2017; Rahman et al., 2013). These papers imply full access for the adversary to the band, phone, and in some cases active (Fereidooni et al., 2017a; Fereidooni et al., 2017b) require a MITM setup for the attack. Thus, their most probable attack model is a *malicious user*, not an ISP. In these studies authors were able to bypass TLS encryption and inject fake data to the remote server. Arias et al. (Arias et al., 2015) managed to access the internal flash of Nike+ Fuelband, inject a modified firmware, and take full control on the band.

Security Analysis of Commercial Wristwatches. Previous works conducted security studies on the particular consumer samrtbands (Clausing and Schiefer, 2016; Cyr et al., 2014; Goyal et al., 2016; Andrew et al., 2016; Clausing et al., 2015b; Clausing et al., 2015a). These works examined security of consumer wearables, including Fitbit, Jawbone, Xiaomi, Apple, Microsoft, Mobile Action, etc. These reports reviewed whether devices use encryption; the source code of the applications is obfuscated; certificate pinning is utilized; and secure Bluetooth Low Energy (BLE) connection is in place.

[1] https://www.washingtonpost.com/business/economy/with-fitness-trackers-in-the-workplace-bosses-can-monitor-your-every-step–and-possibly-more/2019/02/15/75ee0848-2a45-11e9-b011-d8500644dc98_story.html

3 ATTACK AND SETTINGS

In this section we describe the attack in details. We also provide the setting of our experimental infrastructure and our employed tools.

3.1 Attack Description

Following our threat model (section 1) the attacker aims to passively profile users of smartbands based on the IP traffic he receives. On a high level our attack consists of three main steps:

1. discovering the *ground truth*
2. inferring the data leaks from encrypted traffic
3. mass profiling end users of smartbands.

First two steps the curious ISP performs on a **local** setup as a preparation for the attack. The third step the attacker launches **globally**, analyzing all the IP traffic that it receives and forwards.

Discovering the Ground Truth. By "ground truth" we imply the actual data that are sent before encryption. Being able to synchronize a single activity at a time, the adversary is able to observe the exact encrypted traffic that corresponds to a specific activity. The attacker aims to find patterns in encrypted traffic that reveal what data are being transffered. To learn so, the ISP employs a MITM proxy between a smartphone which runs a representative app, and the cloud. Such proxy allows the adversary to observe all the data in plain text. In short, MITM (i) decrypts the traffic, (ii) examines the packet contents, (iii) re-encrypts the traffic, and finally (iv) sends the traffic to its destination. Step (ii) above enables the attacker to look into the packets and find out what exactly is communicated between the smartphone and the cloud. Steps (iii) and (iv) are necessary so as to make sure that communication between the app and the cloud happens in an un-interrupted way. To clarify, the adversary installs the MITM proxy only on a *local* testing setup to study the representative application, it does not attempt to tamper with users' phones.

Discovering Data Leaks. After the adversary sees what data are sent to the cloud after synchronization, it can compare them with the encrypted IP traffic. At this step the attacker utilizes specifics of the TLS encryption: unlike hashing, the size of the output is not fixed. That is, encryption approximately preserves size of the input text:

$$heart\_rate : 80 \xrightarrow{\text{encryption}} ABCDEFGHIJKLM$$
$$heart\_rate : 100 \xrightarrow{\text{encryption}} ABCDEFGHIJKLMN$$

In this case if the adversary learns that this particular encrypted text contains heartbeat data, it will be able to differentiate between 2 and 3-digit beats per minute (BPM). To learn the ground truth for all possible activities, the ISP repeats the above process for every action. That is, it synchronizes 1 activity at a time, and sends all possible variation of the given activity to the cloud. Based on the data obtained from MITM and the size of the correspondent ciphertext, it tries to retrieve information about this action. If the activity produces more than a single encrypted packet, the adversary has to consider not only the size of the packets, but also their order.

Mass Profiling of End Users. Once the curious ISP is able to identify activities that are represented by specific sequences of encrypted packets, it is ready to apply this attack on a bigger scale. At this step, however, the adversary cannot utilize MITM; it can only rely on the plain IP packets that are arriving to be transferred further. Here the attacker is able to exploit the fact that vendors of smartbands do not have many unique IP addresses assigned to their servers that store data of users. By patiently talking with manufacturer servers on its test setup, the adversary is able to gradually learn all the unique IPs of the storage cloud. While TLS allows parts of the accessed URL to be encrypted, and, thus, inaccessible for the ISP, there is no way to conceal IP addresses. To successfully establish mass profiling, the ISP needs to:

- filter the traffic according to the list of IP addresses of vendors servers
- apply the metadata rules learned from previous steps.

3.2 Settings

Studied Fitness Trackers. In this paper we do not use the real company and product names. We adopt pseudonyms instead. We studied 2 of the most popular smartbands readily available: *SmartBandA* and *SmartBandB*. We notified the affected companies about the vulnerabilities.

Learning Ground Truth. For our MITM setup, we utilized the MITM proxy Burp Suite[2]. To disable certificate pinning we employed an open-source Frida toolkit[3]. To observe the encrypted traffic, we used the network protocol analyzer Wireshark[4]. We run the latest android apps on a Google Nexus 6 phone.

Mass Surveillance. For traffic filtering we utilized the intrusion detection system Snort[5] and Wireshark.

[2]https://portswigger.net/burp
[3]https://frida.re/
[4]https://www.wireshark.org/
[5]https://www.snort.org/

Table 1: Information about the activities of users for 2 studied smartbands that the adversary is able to learn. *Occurrence* implies that the attacker can detect whether the activity was done by the user; hence, its frequency as well.

Activity	*SmartBandA*	*SmartBandB*
Heartbeat	occurrence	occurrence & whether heart rate is $\geq$ 100 BPM
Weight	occurrence	occurrence & whether weight is $\geq$ 100 kg
Workout	occurrence & duration of the workout	occurrence
Steps	-	occurrence
Sleep	occurrence	occurrence

3.3 Automatic Activity Aetection

Following the formal attack description, we designed a pipeline for the *automatic* activity detection.

Gathering All Relevant IPs. The attacker collects a list of IP addresses the smartband talks to. This is achieved by continuously initiating connection with the band and collecting correspondent IP addresses.

Traffic Filtering. The adversary filters traffic by the collected IP list, using a Network Intrusion Detection & Prevention System Snort.

Applying Metadata Rules. The ISP applies the previously learned rules for detecting activities using TShark: a Python implementation of the network protocol analyzer Wireshark. TShark enables more functionality for the TLS processing.

Table 1 illustrates what information the ISP is able to find out about 2 studied smartwatches. We explain our findings, and data leaks in the next 2 sections.

4 *CompanyA* SMARTBAND

Following the section 3.1, the adversary aims to retrieve information from the encrypted traffic. The official mobile application for *CompanyA SmartBandA* is called *AppA*. It is one of the most used apps for wearable devices. Overall, the application employs many state-of-the-art security mechanisms including source code obfuscation, SSL pinning, TLS protocol to ensure integrity and confidentiality of the personal data.

4.1 Detected Activities

We have noticed that the different activities produce different patterns in the traffic generated in the network (as expected). For many activities *AppA* encodes data in the URL-format before sending. We learned that the same actions are represented by a determined sequence of packets of almost the same size. Occasionally the same activities can produce packets of a slightly different size. That occurs due to the metadata that is sent together with the health information. For instance, the URL representation of the $18:40$ and $6:40$ clock times would differ by 1 character. Similarly, the encrypted payload sizes would differ by 1 byte. For applicable activities, we will introduce the range of $\pm X$ *bytes* that accounts for such possibilities. In the next sections we describe individual traffic sequences in more details. Table 2 depicts the size of the encrypted traffic for transferred data.

To summarize, the attacker is able to detect heartbeat measuring frequency; exercising frequency, and workouts length; whether the user slept, and recorded weight.

Table 2: Size of encrypted activity packets for *SmartBandA*. *File(s)* illustrates what plain text json files are submitted to the cloud. Size is measured in bytes.

Activity	File(s)	Size of File(s)
Heartbeat	$H1$	$981 + 142 * K \pm 1$
	$H2$	16450 ± 50
Workout	W1	≥ 1293
	W2	$S = 1725 \pm 25$
Sleep	S1	8140 ± 40
Weight	We1	1182 ± 3

4.1.1 Heartbeat Detection

Heartbeat can be easily measured by pressing a correspondent button on the smartband. When the application synchronizes with the cloud, it sends two json files: (i) *H1* and (ii) *H2*. The *H1* file contains (i) the metadata and (ii) user's profile settings. The *H2* file contains information exclusively about the heartbeat rate measured by the wristwatch and sent in the URL-encoding format. It contains all the heartbeat measurements that were taken by the user before the previous synchronization with the application (could be more than one measurement). A single heartbeat measurement always produces a *H2* json file that is 1123 bytes. We have observed that if the user takes two heartbeat measurements (i.e. pushes the button twice), the wristband will send a $1123+142$ bytes file. Similarly, if the user takes three heartbeat measurements, the wristband will send a file that is $1123 + 2 \times 142$ bytes. It appears that for each extra measurement, the appended value is always 142 bytes. For instance, an additional heartbeat of 69 BPM would add the following string to the json file: `,{"time":1591966729,"rate":"RQ==","type":2,"device_id":"16-digit-ID","source":25}`. Since the BPM is represented by 4 characters (`RQ==`), the size of an extra measurement always stays constant. We verified that the URL-encoding of this sequence produces a string of bytes that is 142 bytes long exactly.

We took as many as 50 measurements before synchronizing, and found that the every single experiment fit the following formula: $S = 981 + 142 \times K \pm 1$.

4.1.2 Workout Duration Detection

The *CompanyA* wristwatch offers a great variety of exercising types: running, walking, cycling, swimming, etc. Users can start a new workout by simply pressing the correspondent button on their smartwatches. We established that a workout activity produces two json files: (i) *W1* and (ii) *W2*. *W2* contains statistical information about the workout such as a maximal reached speed, average stride length, minimal and maximal heart rate, number of burned calories. The structure of this file appears to stay relatively predictable regardless of the workout type, duration, and intensity. Indeed, the size of this file is always $S = 1725 \pm 25$ bytes. The *W1* file contains the "trace" of the user's heartbeat during the workout. This trace shows every change of the heartbeat. For example $79;3,-1,2$. This "toy" trace accounts for user's heart rate starting at 79, following by $79+3 = 82$, dropping to $82-1 = 81$, and finally settling at $81+2 = 83$. Since file *W1* contains the entire trace of a user's heartbeat, it appears that the length of *W1* is correlated with the *duration* of the workout. To verify this assumption we manually collected a dataset consisting of seventeen workouts, and tried to fit the linear curve through the datapoints. Figure 2 suggests that the empirical data fit the $payload = 1134 + 2.9 * time$ formula. To roughly estimate the duration of a workout, the following formula can be applied $T = (S - 1134)/2.9$, where S is the size of the packet carrying workout data.

Unlike the heartbeat activity, workouts do not pile in a single json file, but are sent as independent pairs of *W1* and *W2*. To detect the workout activity, adjacent packets that satisfy above size requirements need to be detected. Similarly to heartbeat, the workout activity is represented by multiple consecutive packets, which drastically reduces probability of false positives. As a result a curious ISP is able to detect the number of workouts performed by the user since the last synchronisation. Moreover, it is also possible to approximately detect the length of the workouts.

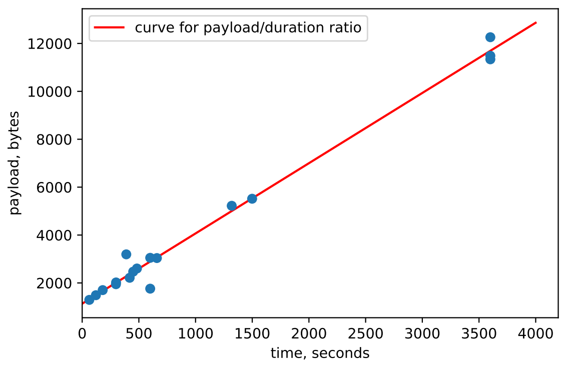

Figure 2: Size of "workout traffic" depending on duration. *SmartBandA* band does not record workouts under 1 minute. Also once the workout exceeds around 100 minutes, it is split into multiple packets, which makes it significantly harder to detect. Such split occurs because the maximal payload size of a TLS packets is 16 KB, and, generally, after 100 minutes a heartbeat trace exceeds this amount.

4.1.3 Sleep Tracking

Tracking sleep is another feature supported by the band, which appeals to many users. The band makes use of the heart rate sensor to detect sleep automatically. For this wristband, the sleeping activity is communicated via the *S1* file. This file also includes (i) device info: firmware version, hardware version, battery level, and (ii) events description: exercise count, number of alarms received. File *S1* appears to be transferred only after a user has slept. The packets sent with this file seem to be (after reconstruction) 8180 ± 2 bytes long. Sleep synchronization differs from other activities, since it does not produce any separate files dedicated to sleep exclusively.

4.1.4 Weight Tracking

Users can input and later adjust their weight in the *AppA* application. People who change their weight frequently are likely caring about their weight, and consider that information sensitive. Recording weight always produces the $We1.json$ (denoted as $We1$) packet that is always $1182 \pm 2\,bytes$ long. Similarly to sleep detection, changing weight produces only a single packet per activity, which makes probability of false positives higher.

5 *CompanyB* SMARTBAND

This section describes the results obtained for *CompanyB SmartBandB*. The official mobile application for *CompanyB SmartBandB* is called *AppB*. It is a very popular application for wearables that trends in

Play Market. The app encrypts data, using TLS. Unlike *SmartBandA*, *SmartBandB* is more of a "smartwatch" than a "fitness tracker" in a traditional sense. It has the same sensors (heart rate monitor, etc.), but allows users to input more of their fitness and health data. The band also uses regular Bluetooth instead of BLE. Similarly to section 4, we focus on detecting heartbeat, sleep, number of steps, workouts and weight changes.

5.1 Synchronization of Packets

Unlike *AppA*, *AppB* synchronizes all the data at once. In total **113** unique files that represent activities in the JSON format are sent to the server during synchronization. However, some of these files can be mapped to a single bigger activity. For instance, a total of 9 separate files (*total_protein*, etc.) are related to the food (e.g. a burger) that users input into the application. On the other hand, there is only a single *heart_rate* file for the heartbeat activity.

We discovered that for each of the files the app checks whether the value has changed (user did a correspondent action); if it is the case, it will send two files: (i) */set*, and (ii) */changes*. If the value has not changed, the app will only send a (i) */changes* file. Hence, it is the */set* files that contain the actual data which the adversary would like to obtain. If a user does every single activity possible before synchronizing, the app would send $2 * 113 = 226$ files at ones. If a user does nothing before synchronization, the app would send 113 files, all with the */changes* affix. Therefore, if an ISP observes a 119-files synchronization, it can conclude that user's actions led to the change of 6 files (113 */changes* and 6 */set*). Each file is carried by a single TLS data packet. All 113 */changes* packets can be easily detected. Their POST requests contain default fields (URL, User-Agent, Content-Type, etc.) without any content. This translates into their encrypted payload being between 580 and 620 bytes long (depending on the URL length). Hence, packets that exceed this size are */set* packets that actually carry the personal data of users. Following section 4, the ISP aims to find patterns in encrypted */set* packets to detect various activities.

Order of Synchronized Files. By studying the source code and observing MITM data, we were able to split 113 */changes* files in two "buckets". The first bucket consists of 10 files that are always sent before others. The remaining 103 json files represent the second bucket; they are sorted alphabetically before being sent. Files in the first bucket, however, are not sorted. Unfortunately for the attacker, the app does not always retain the same order when sending the files. It occasionally slightly (by 1-5 positions) changes the arrangement of records. Since the attacker does not have access to the plain text data, he cannot deterministically infer that the file #42 represents heartbeat. Instead the adversary has to consider the interval between #41-#46; any file in this range could potentially correspond to heart rate.

5.2 Detected Activities

For the attacker to successfully detect an activity two conditions need to be satisfied:

- the packet's TLS payload needs to be of a length correspondent to a particular activity (e.g. for heartbeat it is 1077 bytes).
- the packet needs to be within a sending order interval for this activity (e.g. for heartbeat it is between #41 and #46 out of 113 json files).

Table 3 depicts studied activities and correspondent encrypted patterns. To summarize, the attacker is able to detect heart rate measuring frequency, and heartbeats that are above 100 BPM; workout frequency; whether the user slept, took steps or recorded weight.

Table 3: Size of encrypted activity packets for *SmartBandB*. The field *Files* shows what plain text json files are submitted to the cloud. *Interval* describes possible order of the packets during synchronization. *Size* is measured in bytes. For the BPM (heartbeat) activity the precise formula for the possible size of the H file is $size = 751 + (326 + \frac{1}{2}) \cdot K \pm \frac{K}{2}$.

Activity	Files	Interval	Size of File(s)
BPM	H	42-47	$751 + (326 + \frac{1}{2}) \cdot K$
Workout	W	1-10	≥ 1720
	C	72-73	> 700
Sleep	S	111-113	998 or 1002
Steps	St	1-10	≥ 1216
	St2	1-10	> 700
	T	1-10	> 700
	C	72-73	> 700
	A	61-65	> 700
Weight	We	60-61	901 or 902

5.2.1 Heartbeat Detection

Heart rate can be measured by navigating to a representative section and pressing a button on the band. Heartbeat is sent to the server via the *heart_rate/set* (denoted as *H*) file. Alphabetically the *H* file appears in positions 42-47. We noticed that a single heartbeat measurement always results in a *H* file that is 1077-1078 bytes. The 1 byte difference occurs because user's heart rate can be either a 2- or a 3-digit number. Similarly to *SmartBandA*, multiple measurements of heartbeat are sent in a single file. If a user takes 2 heart rate measurements before synchronizing the app; the size of H increases by 326-327 up to 1404 ± 1 bytes. A similar length increase is observed for three, and more heartbeat measurements. We verified that for an arbitrary K heartbeat measurements, the size of the H file lies within the $[751+326 \times K, 751+326 \times K+K]$ bytes interval. Again, the interval is present due to up to K measurements possibly being 3-digit heart rates. Hence, the adversary is able not only detect the number of measurements, but also to tell the exact number of heartbeat measurements that are > 100 BPM.

5.2.2 Workout Detection

SmartBandB allows users to choose between 17 different workout types: running, walking. cycling, hiking, etc. Workouts are sent to the server as part of an *exercise/set* (W) json file. W is a part of the first non-alphabetic bucket, and can take positions 1-10. Working out also affects the *calories_burned* json file (C): positions 72-73. W contains workout description together with "workout live data": the trace of heartbeat during the exercise. The detection technique from section 4.1.2 could not be applied because multiple workouts are sent together in W. The increase of the file can be attributed to either a long workout or multiple exercises. Given only an encrypted file, it is impossible to distinguish between them. Nevertheless, the attacker can still estimate the minimal size of this file. The json will be the shortest in case of a single workout that lasted 1 second. The shortest value that we managed to obtain is 1720 bytes. There is no upper limit on the size of W (except 16 KB TLS limit).

5.2.3 Sleep Detection

Sleep is automatically recorded by *SmartBandB*. Users also have an option of manually editing/inputting it, and providing a quality mark. Sleep is transmitted via *sleep/set* (S) file that occupies positions 111-113. A sleep file is equal to either 998 or 1002 bytes, depending on how the user rated her sleep on the scale from 1 to 5. Detecting sleep appears to be relatively easy for the adversary since the encrypted payload can only be of 2 possible sizes. Hence, an honest but curious ISP is able to tell if a user slept since the preceding synchronization.

5.2.4 Retrieving Number of Steps

The *SmartBandB* band automatically tracks number of steps taken by users. This information is recorded in the *step_count/set* (denoted as *St*) file that takes positions 1-10. Taking steps also causes four other files to be transmitted to the server: *step_daily_trend* (#1-10), *tracker.pedometer_day_summary* (#1-10), *calories_burned* (#72-73), *activity.day_summary* (#61-65). *St* describes all the steps that were taken since the previous synchronization. It displays them as intervals depending on speed. For example if a user took 100 steps at a normal pace, and then run another 100 steps, the data will be recorded as two separate intervals. This means that similarly to section 5.2.2, there is no upper limit on the size of the *St* file (except 16 KB). We tried to establish the minimal size of the file by taking a single step before synchronizing. In this case *St* is 1216 bytes long. Considering that taking steps invokes changes of 4 different files, the attacker is able to detect the activity with a very high probability.

5.2.5 Weight Detection

Users can record their weight in the *AppB* app. Weight is send to the cloud via *weight/set* (denoted as We): positions 60-61. This file can be either 901 or 902 bytes long depending on whether the input weight is a 2- or a 3-digit number (in kg). Interestingly this allows the attacker to profile users whose weight exceeds 100 kilograms.

6 DISCUSSION

TLS and Wearables. Side-channel leaks of the TLS encryption have been discovered in many web-related systems. Most of such attacks, however, were only able to reduce entropy about the encrypted data. In our setup the attacker is able to extract relevant conclusions about the activities performed by users. Unlike the TLS leaks for web services, the ISP can send a single activity at a time, and directly pinpoint encrypted packets that correspond to a particular activity. This enables the adversary to infer data leaks with greater precision.

Possible Countermeasures:

- *Modifying Plain $\rightarrow$ Cipher Text Size Ratio.* Modification of encrypted payload size can be achieved by modifying the plain text json files. Since pruning those file may result in data corruption, the natural solution would be to selectively increase some of the transmitted files. This can be achieved with "dummy" text to extend files to an arbitrarily sized threshold.
- *Concealing Frequency of Packets Transmission.* To complicate the process of matching pairs of

plain text with encrypted traffic, the app might delay sending user data to the server.

- *Introducing Randomness for Order of Packets.* The process of mapping an activity to its encrypted counterpart could be convoluted by sending activities to a server in a random order. Randomizing the transmitting sequence makes it harder for the adversary to establish patterns in encrypted traffic.

We leave development of a secure traffic transmission system for future work.

7 CONCLUSION

With a vast number of different commercial smartbands readily available, we predict many more wearables leak data while sending them to vendor's servers. An "honest but curious" ISP can easily purchase well-known brands of smartbands, and with the help of the proposed architecture for traffic analysis identify whether a particular band leaks any user's sensitive data. Unlike other attacks on wearables, in our threat model the adversary only observes the traffic without any intent to modify it. This makes it extremely laborious to discover the attacker. Even if detected, it is almost impossible to prove that an ISP committed an unlawful deed. We show that even when the connection channel is encrypted, and the mobile application is reinforced with state-of-the-art security mechanism, it might still be possible to obtain sensitive information about users. Such data include frequency of measuring heartbeat and weigh, the number and duration of workouts, occurrence of sleep, steps, and food records. Naturally, we do not discourage usage of modern defense techniques, but rather challenge vendors to consider bolstering their security even further. Finally, we propose several methods for the attack's mitigation.

REFERENCES

Andrew, H., Christopher, P., and Jeffrey, K. (2016). Every step you fake: A comparative analysis of fitness tracker privacy and security. *Open Effect Report*.

Arias, O., Wurm, J., Hoang, K., and Jin, Y. (2015). Privacy and security in internet of things and wearable devices. *IEEE Transactions on Multi-Scale Computing Systems*, 1(2):99–109.

Bissias, G. D., Liberatore, M., Jensen, D., and Levine, B. N. (2005). Privacy vulnerabilities in encrypted http streams. In *International Workshop on Privacy Enhancing Technologies*, pages 1–11. Springer.

Chen, S., Wang, R., Wang, X., and Zhang, K. (2010). Side-channel leaks in web applications: A reality today, a challenge tomorrow. In *2010 IEEE Symposium on Security and Privacy*, pages 191–206. IEEE.

Cheng, H. and Avnur, R. (1998). Traffic analysis of ssl encrypted web browsing. *URL citeseer. ist. psu. edu/656522. html*.

Clausing, D.-I. E., Schiefer, M., Lösche, U., and Morgenstern, D.-I. M. (2015a). Security evaluation of nine fitness trackers. *The Independent IT-Security Institute*.

Clausing, E. and Schiefer, M. (2016). Internet of things: Security evaluation of 7 fitness trackers on android and the apple watch. *AV TEST, Germany*.

Clausing, E., Schiefer, M., and Morgenstern, M. (2015b). Internet of things: security evaluation of nine fitness trackers. *AV TEST, The Independent IT-Security institue, Magdeburg, Germany*.

Cyr, B., Horn, W., Miao, D., and Specter, M. (2014). Security analysis of wearable fitness devices (fitbit). *Massachusetts Institute of Technology*, 1.

Danezis, G. (2009). Traffic analysis of the http protocol over tls.

Fereidooni, H., Classen, J., Spink, T., Patras, P., Miettinen, M., Sadeghi, A.-R., Hollick, M., and Conti, M. (2017a). Breaking fitness records without moving: Reverse engineering and spoofing fitbit. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 48–69. Springer.

Fereidooni, H., Frassetto, T., Miettinen, M., Sadeghi, A.-R., and Conti, M. (2017b). Fitness trackers: fit for health but unfit for security and privacy. In *2017 IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*, pages 19–24. IEEE.

Goyal, R., Dragoni, N., and Spognardi, A. (2016). Mind the tracker you wear: a security analysis of wearable health trackers. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pages 131–136.

Hintz, A. (2002). Fingerprinting websites using traffic analysis. In *International workshop on privacy enhancing technologies*, pages 171–178. Springer.

Liberatore, M. and Levine, B. N. (2006). Inferring the source of encrypted http connections. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 255–263.

Rahman, M., Carbunar, B., and Banik, M. (2013). Fit and vulnerable: Attacks and defenses for a health monitoring device. *arXiv preprint arXiv:1304.5672*.

Sun, Q., Simon, D. R., Wang, Y.-M., Russell, W., Padmanabhan, V. N., and Qiu, L. (2002). Statistical identification of encrypted web browsing traffic. In *Proceedings 2002 IEEE Symposium on Security and Privacy*, pages 19–30. IEEE.

Zhang, K., Li, Z., Wang, R., Wang, X., and Chen, S. (2010). Sidebuster: automated detection and quantification of side-channel leaks in web application development. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 595–606.

Zhang, Q. and Liang, Z. (2017). Security analysis of bluetooth low energy based smart wristbands. In *2017 2nd International Conference on Frontiers of Sensors Technologies (ICFST)*, pages 421–425. IEEE.