Poster: Fit and Vulnerable: Attacks and Defenses for a Health Monitoring Device

Mahmudur Rahman (PhD Student), Bogdan Carbunar, Madhusudan Banik (PhD Student)
Florida International University, Miami, FL
Email: {mrahm004, carbunar, mbani002}@cs.fiu.edu

I. Introduction

The fusion of social networks and wearable sensors is becoming increasingly popular, with systems like Fitbit [1] automating the process of reporting and sharing user fitness data. Fitbit consists of (i) *trackers*, wireless-enabled, wearable devices that record their users' daily step counts, distance traversed, calories burned and floors climbed as well as sleep patterns when worn during the night, (ii) an online social network (called *webserver* in the following) that automatically captures, displays and shares fitness data of its users and (iii) user USB base stations which acts as a bridge between trackers and the online social network. Trackers communicate to bases over the ANT [2] protocol. Figure 1 shows a system snapshot.

While popular and useful in its encouragement of healthy lifestyles, the combination of health sensors and social networks makes social sensor networks the source of significant privacy and security issues. We have reverse engineered the semantics of tracker memory banks, the command types and the tracker-to-social network communication protocol, and have shown that Fitbit is vulnerable to a wide range of attacks. We have then built FitBite, a suite of tools that exploit these vulnerabilities. We propose FitLock, a lightweight extension that uses efficient cryptographic tools to secure the Fitbit protocol. The project website containing the source code of FitBite and FitLock is made publicly available at [3].

II. Reverse Engineering Fitbit

We reverse engineered the Fitbit communication protocol, including the message communication format among the participating devices and the structure and data format of each memory bank. A tracker has two types of memory banks, (i) *read banks*, containing data to be read by the base and (ii) *write banks*, containing data that can be written by the base.

Fitbit uses *service logs*, files that store information concerning communications involving the base. On the Windows installation of the Fitbit software, daily logs are stored in cleartext in files. Data retrieved from the tracker to be uploaded to the social network is encoded in base64 format. We have exploited Fitbit's lack of encryption in the messages sent between the base and the tracker to implement a USB based filter driver that separately logs the data flowing to and from the base. The log data we captured reveals that during the upload session, the webserver reads data from 6 memory banks, writes on 2 write memory banks and clears data from 5 memory banks by sending requests to the tracker through the base. The read bank #1 stores the daily user fitness records while the write bank #0 stores 64 bytes concerning the device settings as specified on the user's Fitbit account. The communication between the webserver and a tracker through a base is embedded in XML blocks, that contain base64 encoded opcodes – commands for the tracker. All opcodes are 7 bytes long and vary according to specific type of instructions (e.g., read, write, erase).

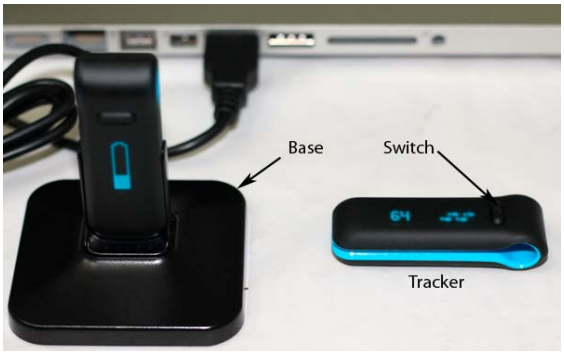


Fig. 1. Fitbit system components: trackers (one mounted on the base), the base (arrow indicated), user laptop. The arrow pointing to the tracker shows the switch button, allowing the user to display various fitness data.

III. FitBite: Attacking Fitbit

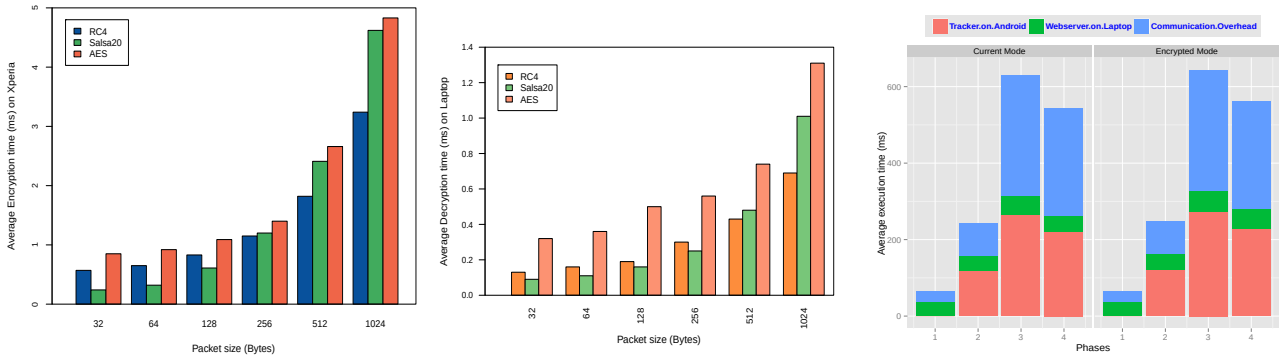
FitBite consists of two modules. The Tracker Module (TM) reads and writes the tracker data. The Base Module (BM) retrieves/injects data from/to the tracker and uploads it into the account of the tracker's owner on the webserver. FitBite implements the following attacks:

Tracker Private Data Capture (TPDC). The TM module is used to discover any tracker device within a radius of 15 ft and capture the fitness information it stores. This attack can be launched in public spaces(e.g., parks, sports venues, etc).

Tracker Injection (TI). The TM module is used to modify any of the fitness data stored by nearby trackers. FitBite reads the selected data from the specific memory bank and modifies the target bytes. The TM can simultaneously modify multiple fitness records (memory banks).

User Account Injection (UAI). The BM module is used to inject data on the Fitbit social network accounts of the owners of nearby trackers. It sends to the webserver a fabricated data reply embedding the desired fitness data, base64 format. The webserver does not authenticate the request message and does not check data consistency, thus accepts the data.

Free Badges and Financial Rewards. By successful injection of large values in their social networking accounts, FitBite enables attackers to achieve special milestones and acquire Fitbit provided "merit" badges, without doing the required work. Fitbit users can link their social networking accounts to systems that reward users for exercising, e.g., Earndit [4]. We



(a) (b) (c)

Fig. 2. FitLock overhead. (a) Encryption time overhead on Xperia. (b) Decryption time overhead on webserver (Dell laptop). (c) Comparison of end-to-end delay between the current Fitbit solution and our proposed encrypted solution

have shown through experiments that attackers can accumulate undeserved financial rewards.

Battery Depletion Attack. FitBite allows the attacker to continuously query trackers in her vicinity, thus drain their batteries at a faster rate.

Mule Attacks. Attacks may also launch physical, *mule* attacks, by attaching trackers to various moving objects (e.g., car wheel, fan). This enables the attacker to unfairly increase her fitness parameters.

IV. FitLock: Protecting Fitbit

The webserver WS is required to maintain a database (Map) associating a tracker id to tracker related data including symmetric key, user id and session id. FitLock consists of a secure tracker binding procedure (BindUserTracker) and an upload procedure (UploadData).

The BindUserTracker procedure allows a user A to bind his tracker T to his social network account hosted on WS. A logs in into his WS account and presses T's switch button for s seconds. T reports its identifier id_T to WS, through the user's base. WS uses the Map structure to retrieve the symmetric key associated with id_T, i.e., sk_T. It then generates a 6 digit long random value, N. WS sends to T the request value $id_T, E_{sk_T}(\text{“}WS'', Time, N)$ that uses sk_T, to decrypt it. It verifies the freshness (the $Time$ value) and authenticates WS through its ability to have encrypted this message. If the verifications succeed, the tracker displays the 6 digit random nonce N. User A reads and enters this nonce into a confirmation box in her Fitbit account. If WS finds any pending (not expired) request matching the value entered by the user, WS associates Id_A to id_T and sk_T in the Map structure.

The UploadData procedure consists of a succession of request/response steps between WS and T. A request from WS to T has the format $id_T, E_{sk_T}(REQ, S_{wst}, C_{ws})$. T's reply is of format $id_T, E_{sk_T}(RESP, S_{wst}, C_T)$. Here, REQ $\in$ {TRQ-REQ, READ-TRQ, WRITE, ERASE, CLOSE} and RESP $\in$ {TRQ-INFO, TRQ-DATA, CLEAR} are packet headers for communication steps of the original Fitbit protocol. S_{wst} is the current session id, C_{ws} and C_T are retransmission counters. Encryption authenticates the participants and the session ids and counters prevent replay attacks.

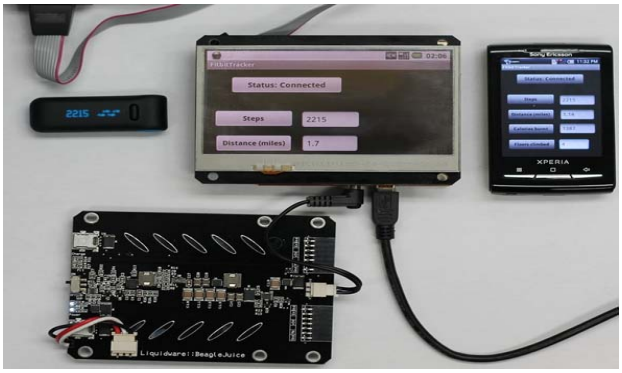


Fig. 3. Snapshot of testbed for FitLock, consisting of BeagleBoard and Xperia devices used as Fitbit trackers.

V. Evaluation

We implemented FitLock in Android. We have tested FitLock's tracker side on a Revision C4 of an OMAP 3530 DCCB72 720 MHz BeagleBoard system [5] and a Sony Ericsson Xperia X10 mini smartphone (ARM 11 CPU@600 MHz, 128MB RAM). We used a Dell laptop featuring a 2.4GHz Intel Core i3 processor and 4GB of RAM, for the web server (built on the Apache webserver 2.4). Figure 3 shows the setup. For a packet size of 1024 bytes, the average encryption time for Salsa20 [6] stream cipher is only 4.62ms (see Figure 2(a)), wheres the average decryption overhead is 1.01ms on the webserver (see Figure 2(b)). We have implemented both the Fitbit and FitLock (each protocol was divided into four phases) on our testbed. The end-to-end time of the FitLock protocol is 1518ms where the total time of Fitbit is 1481ms (see Figure 2(c)). Thus, FitLock adds an overhead of 37ms, accounting for 2.4% of Fitbit's time.

References

[1] Fitbit. http://fitbit.com/.

[2] Ant message protocol and usage. http://www.sparkfun.com/datasheets/Wireless/Nordic/ANT-UserGuide.pdf.

[3] FitBite and FitLock: Attacks and defenses on Fitbit Tracker. http://users.cis.fiu.edu/~mrahm004/fitlock.

[4] Earndit: We reward you for exercising. http://earndit.com/.

[5] G. Coley. *Beagleboard system reference manual.* BeagleBoard.org, December 2009.

[6] Daniel J. Bernstein. The salsa20 family of stream ciphers. In *New Stream Cipher Designs*, pages 84–97. Springer-Verlag Berlin, Heidelberg, 2008.