A Case Study on Vulnerability Analysis and Firmware Modification Attack for a Wearable Fitness Tracker

Jaewoo Shim[1], Kyeonghwan Lim[1], Jaemin Jeong[1], Seong-je Cho[1], Minkyu Park[2] and Sangchul Han[2]*
[1]Dept. of Computer Science and Engineering, Dankook University
Yongin-si, Gyeonggi-do 16890 South Korea
{tlawodn94, limkh120, snorlax, sjcho}@dankook.ac.kr
[2]Dept. of Computer Engineering, Konkuk University
Chungju-si, Chungcheongbuk-do 27478 South Korea
{minkyup, schan}@kku.ac.kr

Abstract

As wearable fitness trackers have been used to collect and analyze users' behaviors such as vital signs and physical workout in daily life, many studies have recently addressed security and privacy issues of wearable fitness trackers. In this paper we focus specifically on (1) analysis of vulnerabilities in firmware of a fitness tracker, a gateway (Android app) connected to the fitness tracker, and communication protocol between the fitness tracker and its genuine gateway using reverse engineering, and (2) creation of a PC based fake gateway by exploiting the vulnerabilities. We finally demonstrate a proof-of- concept firmware attack against a fitness tracker using the created fake gateway.

Keywords: Fitness Tracker, Vulnerability Analysis, Firmware Modification, Fake Gateway, Reverse Engineering

1 Introduction

Wearable fitness trackers including wrist-worn bands have been very popular these days. Wearable fitness trackers can track personal data and promote healthy behaviors [10, 12, 5, 7, 9, 6]. They include embedded sensors and analytic algorithms that can track, analyze and guide wearers' behaviors. For example, they can log footsteps, burned calories, heart rate, and sleep phases, and help to achieve predefined goals for health. A wearable fitness tracker, Fitbit, can sync with users' PCs, mobile phones, and smart watches to monitor steps taken, hours slept and other data they might choose to enter [10]. The wearer's data collected and logged by the fitness trackers is a kind of quite private and sensitive information. As the fitness trackers gather and process potential sensitive data, they must be secured from data manipulation and theft by malicious attackers. Therefore, many research groups have recently addressed security issues and their analysis of fitness trackers [12, 5, 7, 9, 6].

The fitness tracker systems usually consist of three components: a fitness tracker, a gateway such as smartphone, tablet, and PC (also referred to as personal communication device), and a cloud-based server [5, 7, 9]. One reason for this configuration may be to reduce cost by leaving out the network adaptor from the tracker, instead of leveraging the one on the smartphone. In this paper, we use the term gateway and smartphone interchangeably. A tracker is responsible for providing sensor readings and transmitting the data to the smartphone. Data can be synced to the smartphone using wireless communication such as Bluetooth Low Energy (BLE) or ANT+. The smartphone app is then periodically synced to a cloud-based server.

*Corresponding author: Department of Computer Engineering, Konkuk University, 268 Chungwondaero, Chungju-si, Chungcheongbuk-do, 27478, Korea, Tel: +82-43-840-3605

Until now, much attention has been paid to the issues on insecure communication between a tracker and gateway as well as between a gateway and server, sensitive data exposure, and a malicious app on the gateway [12, 5, 7, 9, 6]. However, to the best of our knowledge, there has not yet been substantial interest in the tracker's firmware modification attacks using a fake gateway to cheat a genuine one. Firmware modification attack is so severe because it can tamper with the firmware for a device and run arbitrary code with the highest privilege on the device. Recently, Rieck [9] demonstrated a new firmware modification attack against a fitness tracker, where an adversary manipulated plain HTTP traffic and TLS proxy between an original gateway and server not using a forged gateway.

In this paper, we first investigate firmware update process for a specific fitness tracker available in Korea, and analyze several vulnerabilities in the tracker's firmware, a smartphone app working with the tracker, and the communication protocol between the tracker and smartphone app using static/dynamic reverse engineering. It is also possible to obtain the tracker's firmware from the smartphone app and reverse engineer it. We then create a fake gateway based on a PC and perform firmware modification attack to the tracker using the fake gateway. The main contribution of this paper is to present a new attack vector or path on fitness trackers.

The paper is organized as follows. Section 2 provides related work. Section 3 describes a firmware update process of a fitness tracker and its security threat. In Section 4, we systematically analyze the vulnerabilities of the BLE communication between the tracker and its smartphone app, the smartphone app itself, the tracker's firmware using reverse engineering, and then make a fake gateway and perform a firmware modification attack via the fake gateway. We conclude with future work in Section 5.

2 Related Work

Margaritelli [8] described the Bluetooth security concept and the authentication process. The study showed that a theoretically robust protocol could be easily ruined by poorly implementing it. Barcena [11] described fitness tracking possibilities and security issues, and analyzed apps working with smartphone, apps working with a tracker and the tracker itself. For example, they analyzed the data packages sent and received by the tracker, with the result that most of these could be used to track the user. According to additional analysis of the network traffic, 20 percent of the apps transmitted passwords in plain text. Unucheck [4] presented some obvious threats which were directly consequent to a missing security concept for the tracker. Exploiting heart rate data from a fitness tracker was theoretically possible to observe consumer reactions on certain products. Regarding data authenticity and integrity, there are also some scenarios in which unsecured trackers can lead to serious health related damages [5, 4]. It is also possible to manipulate the trackers function by setting or resetting alarm timers, trigger vibration, etc.

Kim [7] proposed a threat model by dividing the communication path of a fitness tracker into gateway to update server, gateway to database server, BLE connection, and device itself. They also explored vulnerabilities of the three commercial fitness trackers. They found firmware can be modified on some fitness trackers during update procedure. They also found an attacker can arbitrarily modulate the firmware because the devices do not have protection scheme such as obfuscation. However, they only showed the possibility of firmware tampering, but did not prove it. In our work, we tamper the firmware of fitness trackers using a fake gateway.

Rieck [9] defines a smartphone, a tablet, or a computer as a Personal Communications Device (PCD). Data is transmitted between a fitness tracker and an app on the PCD. They analyzed the firmware update process and found the firmware can modified when receiving the firmware from the server to the PCD. They also showed when the server uses the HTTP protocol to communicate with PCD, an attacker can modulate the firmware by performing the Man-in-the-middle (MITM) attack as shown in Fig 1.

In this paper, we analyze the data transmitted between a PCD and a fitness tracker, not between a

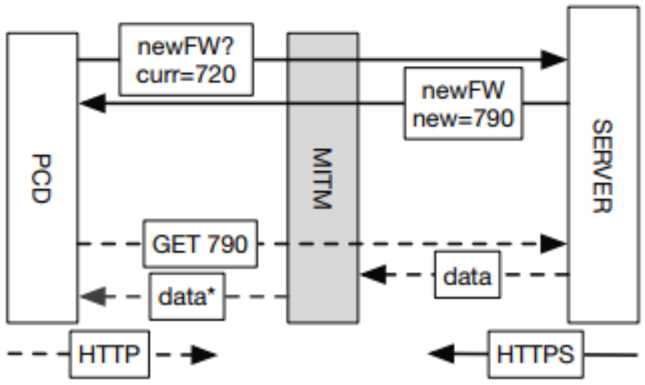


Figure 1: Attacking update procedure[9]

PCD and a server. We reverse engineer an app on a PCD statically and dynamically to gain control of firmware updating procedure. Then, we build a fake gateway (PCD) and successfully update the fitness tracker's firmware with our modified one through the fake gateway.

3 Firmware Update Procedure and Security Threat Analysis of Fitness Trackers

We describe the security threats to fitness trackers. Section 3.1 describes the general firmware update procedure of the fitness tracker. Section 3.2 describes the security threats that may occur during the update procedure, focusing on the communication between the gateway and the fitness tracker.

3.1 Fitness Tracker Firmware Update Procedure

Fig 2 shows a typical firmware update procedure of the fitness tracker. Generally, the fitness tracker firmware update is offered in two forms depending on the vendor.

- Update via mobile application
- Update via gateway

In the first form, when a new firmware is released, the mobile application is updated to get the new firmware. After that, the current firmware information of the fitness tracker is received and the version is checked whether the firmware is to be updated. If the firmware is old, the mobile application send the firmware update request to the device. In the second form, the device checks the available firmware update through the gateway and request to download the new firmware if any from a web server. The gateway then sends the new firmware to the fitness tracker.

3.2 Security Threat Analysis of Fitness Trackers

This section describes security threats that can occur during the communication between a fitness tracker and a gateway. As explained in the previous section, the fitness tracker firmware update is done through a gateway such as a smartphone. We need to analyze the threats to apps on the gateway. If an attacker directly attacks a gateway responsible for firmware update, or creates a fake gateway to request a firmware update, malicious firmware can be delivered to a fitness tracker.

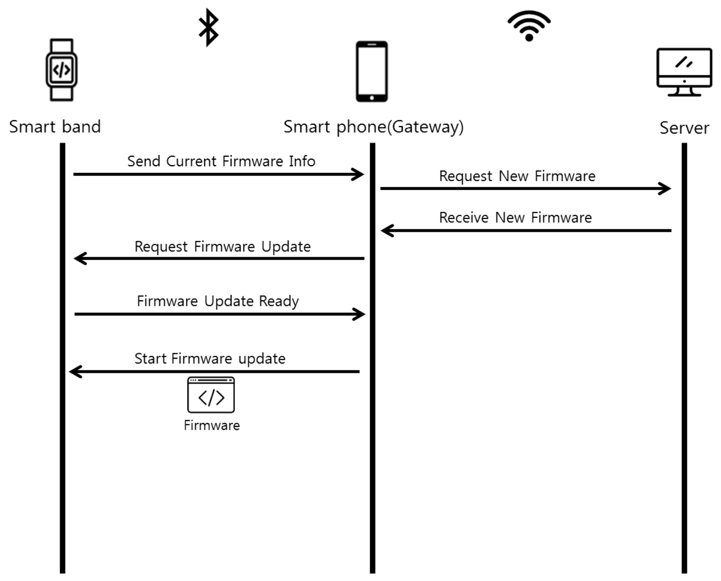


Figure 2: The typical firmware update procedure

First, an attacker attacks a fitness tracker by reverse engineering a gateway application. The attacker infects the victim's gateway and installs a malicious app. Then, a malicious firmware embedded in the malicious app is delivered to the fitness tracker.

If an attacker creates a fake gateway, he can attack a fitness tracker without having to infect the victim's gateway. He collects and analyzes existing applications in the markets such as Google Play Store, and captures the code that performs updates. Based on the analysis, he creates a fake gateway and performs a script that executes the same task as the application's code does. Then, he can attack the fitness tracker.

4 Experiment and Evaluation

Fig 3 illustrates the firmware modification attack model. First, an attacker obtains the gateway app of the target fitness tracker and reverse engineers it. Second, he analyzes BLE communication between the gateway app and the target fitness tracker. Third, he reverse engineers the target fitness tracker's firmware obtained from its gateway app and modifies the firmware. Finally, he builds a fake gateway on a PC and triggers a firmware update with the modified firmware.

In order to prevent malicious behavior of attackers, we keep anonymous the target fitness tracker used in our experiment. We refer to the target fitness tracker as the target device. This section evaluates by experiment the security threat of fake gateway triggered malicious firmware update on the target device.

4.1 Reverse Engineering Gateway App

We obtain the gateway app from `apkpure.com` which is a mirroring site for Android apps. The site provides older versions of apps as well. We reverse engineer the gateway app statically utilizing a Dex to Java decompiler `jadx` [3].

We find out that the gateway app does not employ any anti-reverse engineering technique such as obfuscation or packing. In order to analyze the process of firmware update, we mainly analyze communication-related APIs. We discover that the gateway app uses an Android standard library `android.bluethooth`

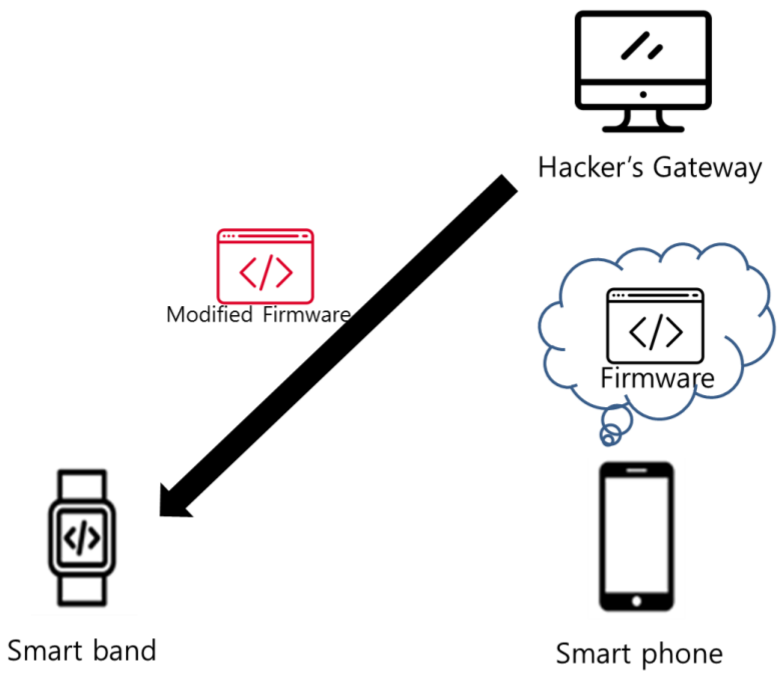


Figure 3: Firmware modification attack model

and transfers data to the target device by calling `writeCharacteristics()`. We also identify a method that decides whether to proceed firmware update or not when the target device sends a packet that includes the current firmware information. This method makes a decision depending on internal conditions. Firmware update proceeds if all the following conditions are satisfied;

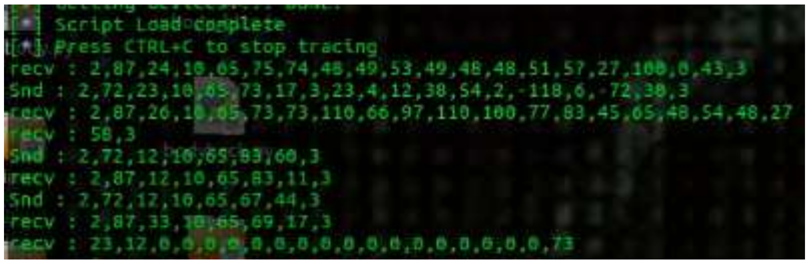
1. A boolean variable is true that indicates valid firmware update.
2. The version of new firmware is higher than a specific version.
3. A firmware exists in directory asset in its APK file.
4. The battery percentage of the target device is greater than 30%.

4.2 Analysis of BLE Communication

We analyze the packets between the gateway app and the target device utilizing a dynamic instrumentation framework `frida` [1]. We hook the BLE communication methods discovered in the static analysis and investigate the packet data. Fig 4 shows an example of BLE packet data. After repeating static and dynamic analysis of the gateway app and the packet data, we identify the structure of BLE data related to firmware update request.

4.3 Firmware Reverse Engineering & Modification

By reverse engineering the gateway app, we find out the firmware exists in directory `asset` in APK file. We extract the firmware and save it as a file. We use IDA pro[2] to reverse engineer and modify the firmware.



```
[*] Script Load complete
[*] Press CTRL+C to stop tracing
recv : 2,87,24,10,65,75,74,48,49,53,49,48,48,51,57,27,100,0,43,3
Snd : 2,72,23,10,65,73,17,3,23,4,12,38,54,2,-118,6,-72,30,3
recv : 2,87,26,10,65,73,73,110,66,97,110,100,77,83,45,65,48,54,48,27
recv : 58,3
Snd : 2,72,12,10,65,83,60,3
recv : 2,87,12,10,65,83,11,3
Snd : 2,72,12,10,65,67,44,3
recv : 2,87,33,10,65,69,17,3
recv : 23,12,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,73
```

Figure 4: API hooking using frida

The target's firmware is a binary file whose format is not known. To obtain information about the binary, we dismantle the target device. We discover that the device adopts nRF51 series SoC produced by Nordic Semiconductor. We also discover that a bootloader and user service programs are delivered together when the firmware is updated. Based on such information and the SoC's datasheets, we achieve information on the file format of the firmware. We disassemble the firmware using IDA pro and acquire its mnemonic codes.

By analyzing the firmware's mnemonic codes, we identify how text are displayed in the target device; the bitmaps of characters and some images are included in the firmware. Since this study aims to present the possibility of firmware modification attack, we simply create a new firmware by modifying the bitmaps.

4.4 Updating Modified Firmware via Fake Gateway

While reverse engineering the gateway app, we find out that there is no re-authentication between the target device and the gateway app when updating firmware if the target device is already paired. This implies that any gateway can establish a connection to the target device. Hence, we builds a fake gateway on a PC running Ubuntu linux 16.04. This PC is equipped with a Bluetooth dongle and the fake gateway can perform BLE communication with the target device. We write a python script that uses a python module `bluepy` which provides BLE communication. Based on the analysis described in Section 4.2, the script triggers firmware update of the target device. The fake gateway transfers the modified firmware described in Section 4.3. The target device displays a message ”HACKED” instead of ”CONNECT” as shown in Fig 5.

In addition, we download several versions of the gateway app and extract several versions of firmware. Then we trigger firmware update with them, upgrading or downgrading firmware arbitrarily. We can see that the target device does not either examine firmware modification or manage firmware version.

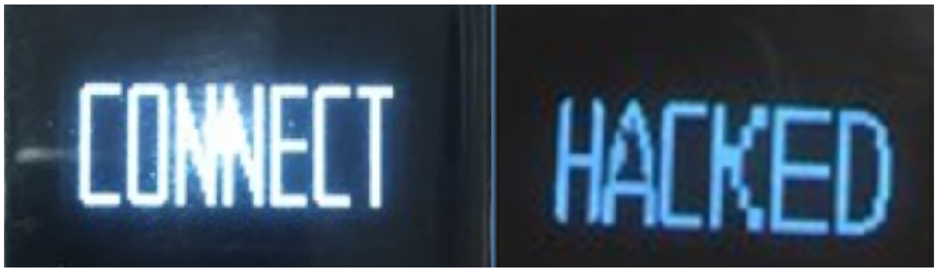


Figure 5: Result of updating modified firmware

4.5 Discussion

This section presents policies that can counter the vulnerability disclosed in the above sections. 1) Mobile application (gateway app) and fitness tracker need to be obfuscated or packed to protect them from reverse engineering attacks. 2) Two-way authentication between gateway app and fitness tracker is necessary for every firmware update request so that only authenticated gateway app can trigger firmware update. 3) Communication data between gateway app and fitness tracker need to be encrypted to provide the confidentiality of firmware. 4) The integrity verification of firmware is necessary by applying cryptographic hash function in order to defend firmware modification attacks.

5 Conclusion and Future Work

To maintain the software in fitness trackers, vendors periodically replace the internal software through firmware update service. In this case, if an attacker modifies the firmware during the firmware update process and uploads the modified firmware to the device, he can execute arbitrary codes in the victim's fitness tracker [12]. In addition, the security of the smartphone app is also very important because the smartphone stores some or all of the recorded data. However, little research has been carried out on the security of the smartphone app working with a fitness tracker.

In this paper, firmware analysis and firmware tampering attacks are conducted for a commercial fitness tracker and defense techniques are discussed. In order to do this, we analyze the process of delivering the firmware developed by the manufacturer to the device and the firmware update process. Firmware is loaded into the fitness tracker through the mobile application of the smartphone. In this paper, fitness tracker firmware is collected by analyzing mobile application that performs fitness tracker firmware update, and reverse engineering is conducted on collected firmware. Then, the firmware was modulated and the modulated firmware was randomly loaded on the smart band. As a result, it is shown that a firmware modulation attack is possible in a commercial fitness tracker, and we suggests a way to defend it.

Acknowledgments

This research was supported by (1) the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2017-2015-0-00363) supervised by the IITP (Institute for Information & communications Technology Promotion) and (2) Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT and Future Planning (No. 2015R1A2A1A15053738).

References

[1] Frida. `https://www.frida.re/` [Online; Accessed on October 3, 2017].

[2] IDA. `https://www.hex-rays.com/products/ida/` [Online; Accessed on October 3, 2017].

[3] Jadx. `https://github.com/skylot/jadx` [Online; Accessed on October 3, 2017].

[4] M. B. Barcena, C. Wueest, and H. Lau. How safe is your quantified self? *Symantech*, August 2014.

[5] E. Clausing, M. Schiefer, U. Lösche, and M. Morgenstern. Security evaluation of nine fitness trackers. *AV-TEST*, June 2015.

[6] A. K. Das, P. H. Pathak, C.-N. Chuah, and P. Mohapatra. Uncovering privacy leakage in ble network traffic of wearable fitness trackers. In *Proc. of the 17th International Workshop on Mobile Computing Systems and Applica-tions (HotMobile'16), St. Augustine, Florida, USA*, pages 99–104. ACM, February 2016.

[7] D. Kim, S. Park, K. Choi, and Y. Kim. Burnfit: Analyzing and exploiting wearable devices. In *Proc. of the 16th International Workshop on Information Security Applications (WISA'15), Jeju Island, Korea, LNCS*, volume 9503, pages 227–239. Springer, August 2015.

[8] S. Margaritelli. Nike+ fuelband se ble protocol reversed, January 2015. `https://www.evilsocket.net/2015/01/29/nike-fuelband-se-ble-protocol-reversed` [Online; Accessed on October 3, 2017].

[9] J. Rieck. Attacks on fitness trackers revisited: A case-study of unfit firmware security. In *Proc. of the Sicherheit 2016, Bonn, Germany, LNI*, volume P-256, pages 33–44. Gesellschaft für Informatik, April 2016.

[10] N. D. Schüll. Data for life: Wearable technology and the design of self-care. *BioSocieties*, 11(3), September 2016.

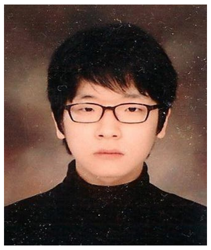
[11] R. Unuchek. How i hacked my smart bracelet, 2015. `https://securelist.com/how-i-hacked-my-smart-bracelet/69369` [Online; Accessed on October 3, 2017].

[12] W. Zhou and S. Piramuthu. Security/privacy of wearable fitness tracking iot devices. In *Proc. of the 9th Iberian Conference on Information Systems and Technologies (CISTI'14), Barcelona, Spain*, pages 1–5. IEEE, June 2014.

Author Biography

Jaewoo Shim received the B.E in the Dept. of Software at Dankook University, Korea in 2017. He is currently a master student at Dept. of Computer Science and Engineering in Dankook University, Korea. His research interests include computer system security, reverse engineering, and software vulnerability analysis.

Kyeonghwan Lim received the B.E. and the M.E. degree in Dept. of Software Science from Dankook University, Korea in 2015 and 2017 respectively. He is currently a Ph.D. student at Dept. of Computer Science and Engineering in Dankook University, Korea. His research interests include computer system security, mobile security.



Jaemin Jeong is an undergraduate student in the Dept. of Software at Dankook University, Korea. His research interests include mobile system security, secure boot and measured boot.

Seong-je Cho received the B.E., the M.E. and the Ph.D. in Computer Engineering from Seoul National University in 1989, 1991 and 1996 respectively. He joined the faculty of Dankook University, Korea in 1997. He was a visiting scholar at Department of EECS, University of California, Irvine, USA in 2001, and at Department of Electrical and Computer Engineering, University of Cincinnati, USA in 2009 respectively. He is a Professor in Department of Computer Science & Engineering (Graduate school) and Department of Software Science (Undergraduate school), Dankook University, Korea. His current research interests include computer security, smartphone security, operating systems, and software protection.

Minkyu Park received the B.E., M.E., and Ph.D. degree in Computer Engineering from Seoul National University in 1991, 1993, and 2005, respectively. He is now a professor in Konkuk University, Rep. of Korea. His research interests include operating systems, real-time scheduling, embedded software, computer system security, and HCI. He has authored and co-authored several journals and conference papers.

Sangchul Han received his B.S. degree in Computer Science from Yonsei University in 1998 and his M.E. and Ph.D. degrees in Computer Engineering from Seoul National University in 2000 and 2007, respectively. He is now a professor in the Dept. of Computer Engineering at Konkuk University. His research interests include real-time scheduling and computer security.