Journal of KIIT. Vol. 19, No. 1, pp. 87-97, Jan. 31, 2021. pISSN 1598-8619, eISSN 2093-7571
http://dx.doi.org/10.14801/jkiit.2021.19.1.87

IoT 헬스케어 기기의 원격 업데이트 취약점 분석

전형석*, 이성기**

Analysis of Remote Update Vulnerabilities of IoT Healthcare Devices

Hyeongseok Jeon*, Sungkee Lee**

요 약

일상생활에서 건강관리에 대한 관심이 높아지고 인터넷의 발달로 인해 IoT 헬스케어 기기가 일반 가정에도 널리 보급되고 있다. 그러나 IoT 헬스케어 기기는 민감한 정보인 사용자의 건강 데이터를 다루고 있음에도 불구하고, 기기의 성능 제한으로 인해 보안 대책이 미미한 실정이다. 특히 네트워크상에서 원격으로 기기를 업데이트하는 OTA(Over-the-air) 프로그래밍 과정에서 보안 공격에 의한 OTA 업데이트 프로토콜 유출 및 이진 코드 유출에 의하여 민감 정보가 유출될 수 있으며, 유출된 민감 정보의 악용에 의하여 기기가 이차 공격을 당할 위험이 크다. 본 논문에서는 IoT 헬스케어 기기의 OTA 업데이트 과정에서의 취약점을 분석하고 업데이트 과정에서의 실제공격 모의실험을 통하여 취약점을 증명하였다. 모의실험에서는 OTA 업데이트 과정에서 스니핑 공격을 수행하여 OTA 업데이트 프로토콜과 소프트웨어 이진 코드를 탈취하였고, 이진 코드의 역공학을 통해 기기의 동작에 관련된 정보를 획득하였으며, 또한 기능이 제거된 더미 프로그램(Dummy program)을 설치하여 기기를 무력화 하는 과정을 보였다.

Abstract

Due to the growing interest in health care in everyday life and advances in the internet technology, IoT healthcare devices are widely used at home. However, despite the fact that IoT healthcare devices generate very sensitive health information, security measures are very low because of devices' capacity limitation. During the remote update of the device software over network which is called over-the-air (OTA) programming, OTA update protocols and binary codes can be leaked by security attacks that leads to sensitive information leaks, or the devices are suffered from secondary attacks by malicious use of the leaked sensitive information. In this paper, we analyzed the vulnerabilities of OTA updates of IoT healthcare devices and demonstrated them by simulating the real-world security attacks. In the simulation, OTA update protocols and binary codes are captured by executing the sniffing attacks, the device operation information is obtained through reverse engineering, and demonstrated the disabling of the device by installing a dummy program.

Keywords
internet of things, IoT, healthcare, network, security, over-the-air update, OTA

* 경북대학교 컴퓨터학부 박사과정
- ORCID: https://orcid.org/0000-0001-6529-3068
** 경북대학교 컴퓨터학부 교수(교신저자)
- ORCID[1]: https://orcid.org/0000-0002-3464-7246

· Received: Dec. 22, 2020, Revised: Jan. 18, 2021, Accepted: Jan. 21, 2021
· Corresponding Author: Sungkee Lee
School of Computer Science and Engineering, Kyungpook National University, 80 Daehakro, Bukgu, Daegu, 41566, Korea
Tel.: +82-53-950-6375, Email: sklee@knu.ac.kr

Ⅰ. 서 론

IoT 헬스케어 기기는 휴대 또는 착용하거나, 신체에 일부 삽입되는 형태로 환자를 치료 및 보조하고 지속적으로 모니터링하며 수집한 데이터를 의료서비스 제공자나 외부 시스템에 전송하는 기기들을 일컫는다. 최근에는 IoT 헬스케어 기기의 대중화로 인해 환자들의 치료 목적뿐만 아니라 가정이나 일상생활에서 일반 사용자들의 건강관리를 위해서도 널리 이용되고 있다. 2025년에는 IoT 헬스케어 기기의 시장 규모가 1800억 달러를 넘어설 것으로 추산되는 만큼 이러한 추세는 앞으로도 더욱 가속화될 것으로 전망된다[1][2].

기존의 헬스케어 기기와 IoT 헬스케어 기기의 가장 큰 차이점은 외부 기기와의 연결 방식이다. 기존의 헬스케어 기기는 외부 기기와 연결되지 않아 측정한 생체 신호 전송 및 기기 관리를 수동으로 하거나 LAN, Serial, USB 등 물리 인터페이스를 통해 외부기기와 연결하여 해당 작업을 수행했다. 그러나 IoT 헬스케어 기기들은 Bluetooth나 WiFi 등 무선 네트워크를 통해 외부 기기와 연결되어 시간이나 장소에 관계없이 사용자의 건강 관련 정보를 전송하거나 기기 설정 변경 및 새로운 소프트웨어 업데이트 등을 수행할 수 있게 되었다[3].

이렇듯 IoT 헬스케어 기기가 통신 및 기기 관리의 수단으로 무선 네트워크를 사용함으로써 편의성을 확보하고 대중화할 수 있었으나 기계와 기계간 통신이 증가하고 통신 자동화에 따른 부작용으로 해킹이나 정보 유출 등의 위험성도 커졌다[4]. 특히 IoT 헬스케어 기기들은 사용자와 가까운 거리를 유지하고 착용 시 이질감을 제거하기 위한 목적으로 소형화가 이루어지는데, 이 과정에서 비교적 작은 컴퓨팅 자원과 배터리 용량을 가지게 된다[5][6][7]. 또한 일정 수준의 성능을 보장하는 PC나 스마트폰과 달리 IoT 헬스케어 기기들은 목적과 형태, 이동수단, 연결되는 외부기기, 사용한 마이크로 컨트롤러 등에 따라 그 성능이 매우 상이하며 이는 일관된 보안 정책을 적용하는 것을 어렵게 만든다[8]. 이런 이유로 많은 IoT 헬스케어 기기들은 보안 문제의 사각지대에 놓여 있으며 무선 통신 프로토콜이나 보안 수준, 사용자 인증 방식 등은 통일되지 못하고 제조사의 재량에 따라 결정되고 있다[9]. 그러나 IoT 헬스케어 기기들은 다른 IoT 기기들과는 달리 단순히 사용자 편의뿐만 아니라 사용자의 건강 문제와 직결될 수 있기 때문에 안전을 위한 기기 보안이 반드시 필요하다. 비록 IoT 헬스케어 기기가 사용자의 개인 정보와 같은 민감한 정보를 가지고 있지 않고 사용자에게 직접적인 위해를 가할 수 있는 기능을 탑재하고 있지 않더라도 데이터 전송 과정에서 사용자 정보를 가지고 있거나 사용자에게 피해를 줄 수 있는 상위 시스템과 연결되기 때문에 반드시 안전하지는 않다.

IoT 헬스케어 기기를 비롯한 IoT 기기들은 소형화를 위해 유선 인터페이스가 제거되거나 기기의 전원 공급을 위해 제한적으로 사용되고 있기 때문에 기기의 소프트웨어를 업데이트할 때 네트워크를 통해 원격으로 업데이트하는 OTA(Over-the-air) 프로그래밍 방식을 사용하게 된다. 이러한 OTA 업데이트는 관리의 효율성과 편의성 때문에 기기 개발사 및 생산 업체에서 가장 선호하는 방법이다. 그러나 IoT 헬스케어 기기에 대한 보안 정책 적용의 어려움 때문에 OTA 사용 시 보안 취약점이 발생할 수 있다. 업데이트 프로토콜에 따라 전송되는 프로그램을 탈취하거나 탈취한 프로그램을 분석 또는 변조하여 악성 프로그램을 IoT 헬스케어 기기에 설치할 경우 기기 오작동을 유발하는 것은 물론이고 2차 해킹의 도구로 활용할 수 있기 때문에 심각한 문제를 야기할 수 있다.

본 논문에서는 IoT 헬스케어 기기의 OTA 업데이트 과정에서의 취약점을 분석하였으며 업데이트 과정에서의 실제 공격을 모의실험하여 취약함을 증명하였다. 이렇게 취약점 분석 및 증명을 통해 해커의 공격 방식을 예측하고 보안 적용에 우선시 되는 부분을 파악하여 보안의 강도를 설정하는데 기여할 수 있다.

본 논문의 구성은 다음과 같다. 2장은 관련 연구로 IoT 헬스케어 기기의 기본적인 특징과 소프트웨어 업데이트 방식에 따른 장단점을 대해 설명한다. 3장에서는 OTA 업데이트로 발생할 수 있는 취약점에 대해 분석하고 4장에서는 모의 공격을 수행함으로써 분석한 취약점을 증명한다. 마지막 5장에서는 결론 및 향후 연구 과제에 대해 기술한다.

II. 관련 연구

2.1 IoT 헬스케어 기기의 특징

IoT 헬스케어 기기는 일반적으로 사용자의 생체 신호를 측정하고 이를 게이트웨이에 전송하는 것을 목적으로 한다. 그림 1에서 보듯이 게이트웨이는 IoT 헬스케어 기기에서 측정한 정보를 가공하고 환자 정보 등 기타 정보를 추가하여 개인 건강 기록 형태로 서버나 클라우드에 저장한다. IoT 헬스케어 기기가 측정한 사용자의 생체 신호 전송은 일반적으로 블루투스나 WiFi 등 근거리 무선 통신을 통해 이루어진다. IoT 헬스케어 기기는 휴대형, 착용형, 삽입형 등 다양한 형태로 사용자와 가까운 거리를 유지하며 사용 편의성을 위해 소형화 및 경량화를 지향하게 되고, 이 과정에서 기기는 게이트웨이에 비해 비교적 작은 컴퓨팅 자원을 가지게 된다. 일반적으로 이 모델에서 IoT 헬스케어 기기는 필요성과 성능상의 한계를 고려하여 그 자체가 환자의 개인 정보나 인증서 등 민감한 정보를 가지지 않고 게이트웨이에서 민감 정보를 관리하게 된다. 따라서 IoT 헬스케어 기기에서 게이트웨이를 거쳐 서버 쪽으로 갈수록 정보의 중요성이 커지고 기기의 성능도 증가하며, 그 결과 더 강한 보안과 개인 정보보호가 요구 및 적용 되고 있다.

IoT 헬스케어 기기의 소형화 과정에서 USB 포트 등 외부 기기와 연결하는 유선 인터페이스는 제거될 가능성이 크고, 디스플레이나 버튼과 같은 사용자와의 상호 작용을 위한 인터페이스 역시 최소화된다. 이러한 이유로 IoT 헬스케어 기기의 소프트웨어를 업데이트하거나 설정 변경 등을 수행할 때는 외부 기기에서 무선 통신으로 연결되어 원격에서 유지 보수 작업이 이루어지는 경우가 많으며, 이 방식은 비교적 작지만 다수의 전문화된 센서나 기기들이 상위 기기에 연결되어 동작하는 IoT 헬스케어 서비스에 특히 유용하다.

2.2 IoT 헬스케어 기기 업데이트 방식 비교

표 1은 업데이트 방법에 따른 주요 차이점과 장단점을 보인다.

표 1. 인터페이스에 따른 업데이트 특징 비교
Table 1. Comparison of update characteristics by interface

Interface / Characteristic	Wired	Wireless	Not Supported
Types	USB, Serial, LAN, etc.	Bluetooth, ZigBee WiFi, etc.	-
Security	Moderate	Variant	High
Updater	Expert	Central management server	-
Cost	High	Low	-
Update rate	Low	High	-
Flexibility	Moderate	High	Low

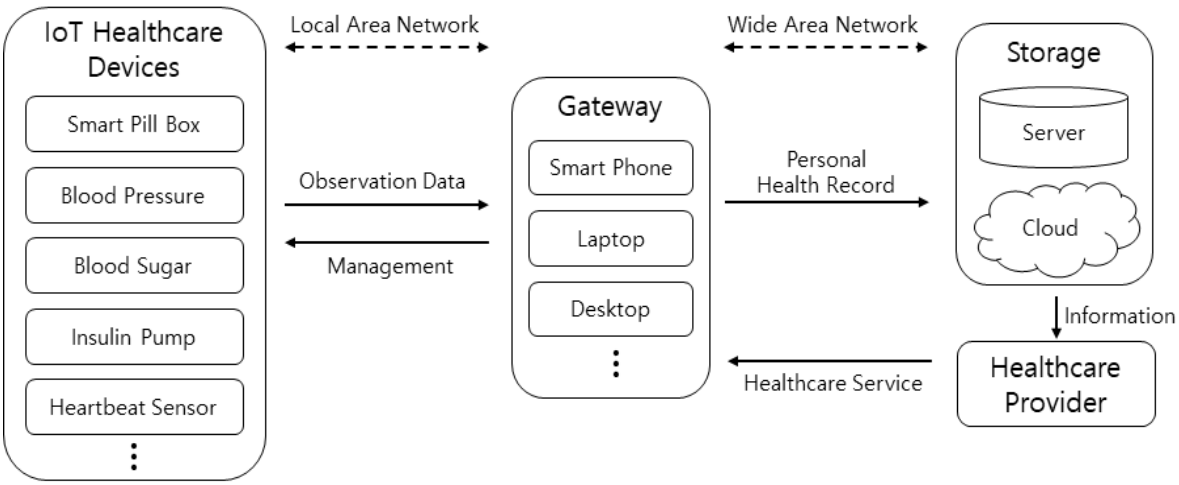

그림 1. IoT 헬스케어 서비스 모델
Fig. 1. IoT healthcare service model

버그 수정, 기능 개선을 위해 IoT 헬스케어 기기의 소프트웨어를 업데이트하는 방법은 크게 유선 인터페이스 또는 무선 인터페이스를 이용하는 방법으로 구분할 수 있으며, 경우에 따라서는 유지 보수를 위한 인터페이스가 존재하지 않을 수도 있다.

USB나 Serial 등 유선 인터페이스를 사용하는 경우에는 주로 전문가에 의해 업데이트가 이루어지며 많은 기기를 일일이 업데이트해야 하기 때문에 많은 시간과 비용이 소모되고 사용자가 업데이트를 위해 서비스 센터에 기기를 위탁하는 절차가 필요하므로 업데이트 적용률이 낮은 편이다. 그러나 내부자가 소프트웨어를 유출 시키지 않는다면 비교적 보안성이 높고 해킹 공격 역시 개별 기기에 한정되기 때문에 안정성도 높다는 장점이 있다.

반면에 Bluetooth, WiFi 등 무선 인터페이스로 OTA 프로그래밍 업데이트 방식을 사용하는 경우에는 중앙 관리 시스템 또는 중앙 관리 시스템의 요청을 받은 게이트웨이에서 개별 기기에 새로운 소프트웨어를 자동으로 업데이트할 수 있다.

그림 2는 OTA 업데이트 개요를 설명하고 있으며 IoT 헬스케어 기기에 네트워크로 연결된 중앙 관리 시스템이 직접 업데이트를 수행하거나 게이트웨이에 업데이트할 소프트웨어를 전달하고 명령하여 대신 수행하게 할 수도 있다. OTA 업데이트 방식을 사용할 경우 업데이트에 적은 비용과 시간이 소모되며 편의성도 높고 업데이트 적용률도 높다. 특히 IoT 헬스케어 기기는 사용자로부터 수집한 건강 관련 기록들을 게이트웨이를 통해 서버에 전송하기 때문에 구조상 OTA 업데이트 모델을 적용하기 쉽다.

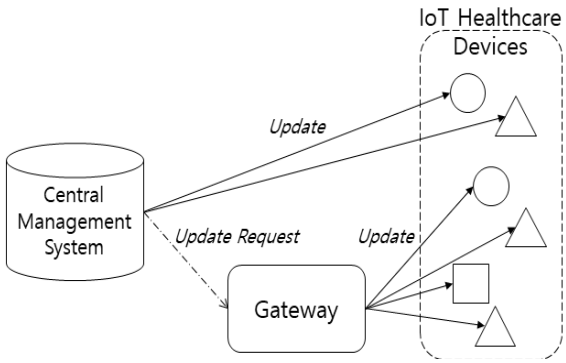

그림 2. OTA 업데이트 모델
Fig. 2. OTA update model

그러나 IoT 헬스케어 기기와 중앙 관리 시스템 또는 게이트웨이 간 보안 정책에 따라 상이한 보안성을 가지며 소프트웨어나 업데이트 프로토콜이 유출될 경우 이를 악용하여 다수의 기기들이 공격받을 수 있다.

상황에 따라 IoT 헬스케어 기기가 업데이트를 위한 인터페이스 및 기능을 지원하지 않을 수도 있는데, 이 경우 소프트웨어가 유출될 가능성이 매우 적고 기기에 악성 소프트웨어를 설치하기가 사실상 불가능해 보안성은 매우 뛰어나지만 기기의 오류 수정이나 새로운 기능을 제공할 수 없어서 유연성이 떨어진다.

III. OTA 업데이트 취약점 분석

2장에서 설명한 바와 같이 무선 인터페이스를 활용한 OTA 업데이트는 IoT 헬스케어 기기를 관리하는데 적합한 구조를 가지고 있으나 보안상의 위험성이 따른다.

표 2는 OTA 업데이트 시 유출될 수 있는 주요 정보들과 각각의 악용 사례를 정리한 것으로, 업데이트 프로토콜 유출로 인해 공격자가 IoT 헬스케어 기기에 악성 소프트웨어를 설치할 수도 있고 업데이트 과정에서 소프트웨어의 이진 코드가 유출되어 역공학(Reverse engineering)을 통해 다른 공격에 사용할 수 있는 정보가 유출될 수도 있다. 이러한 위험성에도 불구하고 IoT 헬스케어 기기는 기기별로 상이한 성능과 특성으로 인해 표준화된 프로토콜이나 보안 절차의 제정이 미비한 실정이다[10]. 또한 IoT 기기의 특성상 보안 대책이 마련되어 있다고 하더라도 사용자에 의해 무력화되기도 한다.

표 2. OTA 업데이트 시 주요 유출 정보 및 악용 사례
Table 2. Leakable key information and abuse cases when using OTA updates

Leackable Key Information		Abuse case
Update protocol		Install malware
Binary code	Authentication information	Rogue device attack, Man-in-the-middle attack
	Data exchange format	Unauthorized data accessing, Data forging attack
	Control command	Unauthorized device control, Cause malfunction

예를 들면, 공용 무선 네트워크를 통해 IoT 헬스케어 기기와 게이트웨이를 연결하는 경우에는 기본적인 암호화를 통한 보안 혜택을 받을 수 없다.

본 장에서는 IoT 헬스케어 기기에 보안 절차 적용의 필요성을 확인하기 위해 OTA 업데이트 상황에서의 취약점을 정리한다.

3.1 OTA 업데이트 프로토콜의 유출

무선 인터페이스를 통해 보안이 적용되지 않은 네트워크에 연결하여 OTA 업데이트를 수행하는 과정에서 네트워크 스니핑(Sniffing) 공격으로 업데이트 프로토콜이 유출될 수 있다. 특히 비교적 시스템 자원이 부족하고 컴퓨팅 성능이 떨어지는 IoT 헬스케어 기기의 경우 간단한 프로토콜을 사용하여 OTA 업데이트가 이루어질 가능성이 크기 때문에 더욱더 쉽게 분석 당할 수 있다. 그 결과 업데이트를 수행하는 중앙 관리 시스템 및 게이트웨이의 네트워크 특성 정보와 업데이트 과정에서 기기 간에 송수신하는 메시지 등이 유출될 수 있다.

공격자는 유출된 업데이트 프로토콜을 모방함으로써 악성 소프트웨어를 IoT 헬스케어 기기에 설치할 수 있으며 그 결과 기기의 기능을 무력화하여 사용자에게 피해를 입히거나 기기의 통신 기능을 악용하여 IoT 헬스케어 기기와 연결된 다른 시스템에도 2차적인 피해를 입힐 수 있다. 실제로 IoT 기기를 분산 서비스 거부 공격(Distributed Denial of Service attack, DDoS)에 활용하거나[11], 특정 동작 수행을 반복시켜 해당 기기 또는 해당 기기와 연결된 주변 기기의 배터리를 방전 시키는 공격(Battery exhaustion attack) 등이 가능하다[12]. 이러한 공격은 같은 OTA 업데이트 프로토콜을 사용하는 다수의 IoT 헬스케어 기기에 동시다발적으로 수행할 수 있기 때문에 피해가 쉽게 확산된다.

3.2 이진 코드 유출

네트워크 스니핑으로 OTA 업데이트 과정에서 소프트웨어의 이진 코드가 유출되는 경우에는 역공학을 통해 다음과 같은 추가 정보가 유출될 수 있다.

첫째로는 기기나 사용자 인증 관련 정보 또는 IoT 헬스케어 기기와 게이트웨이 간 연결 및 통신에 사용되는 정보들이 유출될 수 있다. 이러한 정보들이 유출된다면 정상 IoT 헬스케어 기기를 흉내내 게이트웨이와 연결하거나 정상 게이트웨이를 흉내 내어 IoT 헬스케어 기기의 연결을 유도하는 비인증 기기 공격(Rogue device attack)[13], IoT 헬스케어 기기와 게이트웨이 간 연결에 끼어들어 전송 데이터를 가로채거나 조작하는 중간자 공격(Man-in-the-middle attack, MITM)에 활용될 수 있다[14].

둘째, IoT 헬스케어 기기가 사용자로부터 수집한 건강 관련 정보를 게이트웨이에 전송할 때 사용하는 데이터 전송 양식이 유출될 수 있다. 데이터 전송 양식이 유출될 경우 인가받지 않은 제3자가 정상 사용자의 건강 관련 데이터를 열람할 수 있으며 비인증 기기나 중간자 공격을 통해 사용자의 건강 상태를 임의로 조작하여 전송할 수 있게 된다. 이 경우 IoT 헬스케어 기기의 특성상 기기가 전송하는 데이터에 사용자의 신상정보와 같은 민감한 정보가 포함되어 있지 않은 비교적 보안 요구 수준이 낮은 기기임에도 불구하고 건강에 직접적인 위해를 가하게 된다. 공격자가 악의적인 목적을 가지고 게이트웨이에 지속적으로 사용자의 정상적인 생체 신호를 전송하는 경우 헬스케어 서비스 제공자가 사용자의 응급 상황을 적시에 파악할 수 없으며 반대로 비정상적인 생체 신호를 전송하여 응급 상황으로 오인하도록 만들 수 있다. 그 결과 서비스의 품질과 데이터 저장소의 신뢰성을 파괴할 뿐만 아니라 사용자의 건강에 심각한 악영향을 초래할 수도 있다.

셋째, 제어 명령어 및 개발자가 디버그를 위해 남겨둔 명령어가 유출될 수 있다. 이러한 기기 제어 명령어들은 해커의 주요 공격 수단 중 하나이며 IoT 헬스케어 기기를 원격에서 임의로 동작시킬 수 있기 때문에 사용자에게 직접적인 피해를 유발할 수 있다.

IV. OTA 업데이트 공격 모의 실험

본 논문에서는 IoT 헬스케어 기기의 보안 취약점

을 증명하기 위하여 OTA 업데이트 과정에서 스니핑 공격을 수행하여 OTA 업데이트 프로토콜과 소프트웨어의 이진 코드를 탈취하였고 이진 코드의 역공학을 통해 기기의 동작에 관련된 정보를 획득하였다. 또한 기능이 제거된 더미 프로그램(Dummy program)을 설치하여 기기를 무력화하는 과정도 수행하였다.

일반적인 네트워크 스니핑 공격은 전문적인 네트워크 모니터를 사용하며 프로그램 역공학은 분석 도구 소프트웨어를 활용한다. 그러나 본 연구에서는 보안이 적용되지 않은 OTA 업데이트의 위험성을 보이기 위해 Raspberry Pi로 네트워크 모니터를 구성하여 스니핑 공격을 수행하고 문자열 위주의 이진코드 분석을 통해 간단하게 주요 정보를 획득하였다.

4.1 OTA 업데이트 스니핑

스니핑 공격을 위해 Kali Linux 32-Bit를 설치한 Raspberry Pi 4 Model B와 시중에서 판매되고 있는 I사의 무선 네트워크 어댑터를 사용해 네트워크 모니터를 구성하고 Arduino MKR1000 WiFi 보드와 심박 센서를 결합하여 모의 IoT 헬스케어 기기를 제작하였다. 모의 IoT 헬스케어 기기는 무선 인터페이스로 PC와 연결하여 측정한 사용자의 심박 수치를 PC로 전송하거나 PC로부터 OTA 업데이트를 받는다. IoT 헬스케어 기기와 PC는 보안이 적용되지 않은 WiFi로 연결되도록 설정했으며 네트워크 모니터는 이를 탐지 가능한 물리적 거리에 위치하였다. 네트워크 모니터는 아직 IoT 헬스케어 기기와 게이트웨이의 아이피나 송수신 포트 등 네트워크 특성을 알 수 없기 때문에 수집한 패킷들 중에서 OTA 업데이트에 사용된 패킷들을 선별하는 과정이 필요하다.

패킷 스니핑 및 OTA 업데이트 패킷 선별 알고리즘은 다음과 같다.

1. 네트워크 모니터는 주변의 모든 네트워크 패킷 수집.
2. 수집된 패킷들 중 같은 무선 네트워크에 연결된 기기들 간에 송수신한 패킷 선별.
3. 데이터 전송 빈도수가 적고 전송 데이터 사이즈가 작은 기기 선별.
4. 짧은 시간 동안 비교적 큰 패킷을 수신한 기기 선별.
5. 패킷에서 어셈블리 명령어 검색.

두 번째 과정에서 패킷의 출발지와 도착지를 확인하고 같은 서브넷에 속해 있다면 두 기기는 동일 네트워크에 연결되어 있음을 알 수 있다. IoT 헬스케어 기기와 게이트웨이는 같은 네트워크에 연결되어 있으므로 이 과정을 통해 목표 기기와 패킷들을 선별할 수 있다.

세 번째 과정은 IoT 헬스케어 기기가 일반적으로 배터리 제한이나 기기 성능의 제한으로 지속적으로 패킷을 보내지 않고 측정 데이터 전송과 같이 이벤트가 발생했을 때만 메시지를 전송한다는 특징을 이용하는 것이다. 반대로 송수신 빈도수가 높은 기기는 PC나 스마트폰 등의 기기일 가능성이 높기 때문에 목표 기기 및 목표 패킷에서 제외한다.

네 번째 과정은 프로그램 업데이트에 수 킬로바이트에서 수십 킬로바이트 이상의 데이터 전송이 필요하고 크기가 큰 데이터는 여러 윈도로 나누어진 연속적인 패킷의 형태로 네트워크에서 전달된다는 것을 이용한 것이다. 따라서 비교적 패킷 송수신량이 많지 않은 기기에서 순간적으로 대량의 패킷을 수신한다면 IoT 헬스케어 기기가 OTA 방식으로 업데이트 패킷을 전달 받았다고 가정할 수 있다.

다섯 번째 과정은 위 과정에서 선별한 패킷에서 어셈블리 명령어를 찾아 OTA 업데이트 패킷임을 확인하는 과정이다. 이때 레지스터를 사용하지 않아서 항상 고정된 명령 코드(Operation code, Opcode)를 가지는 어셈블리 명령어들이 유효하다. 이러한 명령 코드로는 ret(Return from subroutine), clt(Clear T flag), ijmp(Indirect jump) 등이 있고, 특히 ret 명령 코드는 프로그램을 구성하는 서브루틴 별로 하나씩 존재하기 때문에 사용 빈도가 매우 높다. 연속된 패킷들에서 명령 코드와 일치하는 값들이 높은 빈도로 존재한다면 해당 패킷들은 프로그램의 일부일 확률이 높기 때문에 업데이트 패킷으로 간주한다.

상기한 과정을 통해 IoT 헬스케어 기기와 PC 게이트웨이를 특정하고 두 기기 간 OTA 업데이트 스니핑 시도를 총 10회 수행하였으며 그 결과는 다음과 같다. 업데이트 한 프로그램은 35개의 패킷으로 나뉘어 전송되었으며 1회 공격을 통해 평균 29.2개, 83.4%의 패킷을 스니핑 할 수 있었다. 최소 손실 회차에서는 1개의 패킷만 손실되었고 최대 손실 회차에서는 12개의 패킷이 손실되었다. 3회차 시도 때 소프트웨어 전송 전후로 전송되는 OTA 업데이트 포로토콜 패킷과 OTA 업데이트로 전송되는 소프트웨어 전체의 이진 코드를 확보할 수 있었다. 실험을 위해 구성한 네트워크 모니터를 일반인들도 시중에서 쉽게 구입할 수 있는 제품들로만 구성한 것을 감안하면 전문 네트워크 모니터를 사용하거나 동일한 네트워크 모니터를 다수 사용함으로써 스니핑 신뢰도와 정확성을 더욱 높일 수 있을 것으로 예상된다.

모의실험으로 확보한 OTA 업데이트 패킷을 분석한 결과 업데이트할 프로그램의 바이너리 코드는 HTTP로 전송된다는 것을 알 수 있었다. 특히 HTTP 메시지의 헤더 속성 중 권한 인증에 사용되는 Authorization 속성에 ID와 password의 쌍으로 추정할 수 있는 HBSens:0800200c9a66 문자열이 Base 64로 인코딩되어 존재하는 것을 확인하였다. 이 밖에도 IoT 헬스케어 기기와 PC 게이트웨이의 IP, 포트번호 등 네트워크 특성 및 업데이트 요청 URI, IoT 헬스케어 기기가 업데이트 프로그램 수신 후 게이트웨이에 전송하는 응답 메시지 등을 확인할 수 있었다.

4.2 탈취 소프트웨어의 이진 코드 분석

스니핑 공격으로 획득한 소프트웨어 이진 코드는 소프트웨어 동작 시 사용되는 문자열들을 찾는 방식으로 분석을 수행하였다. 이러한 방식은 소프트웨어 전체의 동작 방식을 모두 파악할 필요가 없고 문자열은 이진 코드 상에서 연속된 형태로 나타나므로 분석 난도가 낮다. 바이너리 코드에서 추출한 문자열은 기기 간 통신 인증 과정에서 사용되는 아이디나 비밀번호 후보군으로써 무차별 대입 공격의 성공률을 높이거나 통신 프로토콜 및 데이터 전송 양식을 이해하는데 사용한다.

프로그램 동작 과정에서 문자열을 사용하는 방식은 프로그래밍 방법에 따라 결정되기 때문에 역공학을 통해 문자열을 식별하는 방법도 상이하다.

프로그램 영역에서 문자열을 읽고 쓰는 경우에 대한 이진코드 분석 알고리즘은 다음과 같다.

1. 4개의 연속된 ldi(Load immediate) 명령어 찾기.
2. 연속된 cp(Compare), cpc(Compare with carry) 명령어 찾기.
3. breq(Branch if equal) 명령어 찾기.
4. movw(Copy register word) 명령어 찾기.

첫 번째 과정에서 찾는 ldi 명령어는 8비트 상수를 레지스터에 저장하는 명령어이며 프로그램이 문자열의 시작과 끝 메모리 주솟값을 레지스터에 저장할 때도 사용되기 때문에 문자열 식별의 시작으로 활용한다. 각 주솟값은 16비트이므로 하나의 주솟값을 레지스터에 저장하기 위해서는 하위 주솟값과 상위 주솟값으로 나누어 레지스터에 저장해야 하기 때문에 두 개의 ldi 명령어가 필요하다. 따라서 문자열의 시작 메모리 주솟값과 끝 메모리 주솟값을 다루기 위해서는 4개의 연속된 ldi 명령어가 바이너리 코드 상에서 나타나게 된다.

두 번째 과정은 프로그램이 문자열을 활용하기 위해서는 문자열의 시작 주솟값부터 데이터를 읽고 주솟값을 하나씩 증가시키며 문자열 끝 주솟값까지 데이터 읽는 과정이 필요하다는 것을 이용한 것이다. 따라서 현재 데이터를 읽고 있는 주솟값이 끝 주솟값과 동일한지 확인하고 데이터 읽기를 종료할지 판단하기 위해서는 두 주솟값을 비교하는 연산이 필요하고, 이때 사용되는 것이 cp 명령어와 cpc 명령어이다. cp 명령어는 두 개의 레지스터를 비교하는 명령어이고 cpc는 올림 값(carry)을 포함하여 두 개의 레지스터를 비교하는 명령어이다. 두 주솟값을 비교하기 위해서는 하위 주솟값을 비교하기 위한 cp 명령어와, cp 명령어의 결과로 생성되는 올림 값을 포함하여 상위 주솟값을 비교하기 위한 cpc 명령어가 반드시 뒤따른다.

세 번째 과정에서 찾는 breq 명령어는 비교 연산인 cp 또는 cpc 명령어와 함께 사용되며 비교 연산의 결과에 따라 프로그램의 실행 부분을 분기하는 역할을 한다. 문자열 읽기 과정에서 아직 문자열 읽기를 마치지 못했다면 프로그램을 순차적으로 수행하여 문자열 읽기를 계속하고, 문자열 읽기를 마쳤다면 다음 서브루틴을 실행하기 위해 프로그램 실행 위치를 변경하는 목적으로 breq가 사용된다.

네 번째 과정의 movw 명령어는 두 개의 레지스터에 저장된 값을 다른 두 레지스터에 복사하는 명령어다. 문자열 읽기 과정에서는 레지스터에 저장된 문자열 주솟값을 포인터 레지스터로 옮기기 위해 사용된다. 프로그램 영역에 저장된 문자열을 읽기 위해서는 메모리 주솟값을 저장한 포인터 레지스터를 통해 문자열이 위치한 메모리에 접근해야 하기 때문에 반드시 필요한 과정이다.

다섯 번째 과정에서 찾는 lpm 명령어는 포인터 레지스터에 저장된 주솟값이 가리키는 메모리 영역에서 1 바이트를 읽는다. 문자열 읽기 과정에서 최종적으로 하나의 문자를 읽는데 사용된다.

정리하면 바이너리 코드 상에서 일련의 ldi, ldi, ldi, ldi, cp, cpc, breq, movw, lpm 명령어가 나타나면 프로그램 영역에서 문자열을 순차적으로 읽어서 사용하는 부분이라고 추정할 수 있다. 따라서 ldi 명령어들이 불러오는 메모리 주소에 저장된 값들을 확인함으로써 프로그램 안의 문자열을 추출할 수 있다.

표 3은 이진 코드 분석 과정에서 확인할 수 있었던 문자열의 예이다. 문자열은 메모리 주소 0x68~0x74에 있으며 주로 통신 제어에 사용되는 문자인 STX(\0x02)와 NULL(\0x00), 특수 문자인 '!'를 포함하여 \x02\0x00SN205315! 이다.

이러한 과정들을 거쳐 추출한 문자열들과 각 문자열 별 실제 프로그램에서의 역할을 정리하면 표 4와 같다. 추출한 문자열들 중 HBSens와 \x02\0x00SN205315!는 OTA 업데이트 과정에서 권한 인증에 사용되는 주요 정보이며, 이 밖에도 측정 데이터 요청 메시지 식별자, 측정 데이터 전송 메시지를 구성하는 문자열, 접속할 WiFi 식별자 기본값, 디버그에 사용되는 문자열 등을 획득할 수 있었다.

표 3. 바이너리 코드 분석 예
Table 3. Example of binary code analysis

Addr	Hex	Meaning	
...	...		
68	02 00	STX, NULL	
6a	53 4e	S N	
6c	32 30	2 0	
6e	35 33	5 3	
70	31 35	1 5	
72	21 03	! ETX	
74	00 00	NULL, NULL	
...	...		
51a	c8 e6	ldi	r28, 0x68
51c	d0 e0	ldi	r29, 0x00
51e	04 e7	ldi	r16, 0x74
520	10 e0	ldi	r17, 0x00
522	0c 17	cp	r16, r28
524	1d 07	cpc	r17, r29
526	41 f0	breq	.+16
528	de 01	movw	r30, r28
52a	64 91	lpm	r22, Z
52c	86 e1	ldi	r24, 0x16
52e	91 e0	ldi	r25, 0x01
530	03 94 06 01	call	0x20c
534	21 96	adjw	r28, 0x01
536	f5 cf	rjmp	.-22
...	...		

표 4. 추출 문자열과 그 역할
Table. 4. Extracted strings and their roles

Extracted String (ASCII)	Role
TestWifi	Default SSID
HBSens	ID for OTA update
\x02\0x00SN205315!	New password for the next OTA update
TS:	Part of the data report message format (time stamp)
BPM:	Part of the data report message format (unit)
SSID:	Debug string
IPAddr:	Debug string
Attempting to connect:	Debug string
GetDataReq	Control string for data request

4.3 더미 프로그램 설치

본 논문에서는 스니핑 과정에서 확보한 OTA 프로토콜과 이진 코드 분석으로 추출한 문자열들을

사용하여 모든 기능이 제거된 더미 프로그램(Dummy program)을 IoT 헬스케어 기기에 설치하였다.

더미 프로그램을 설치하는 절차는 다음과 같이 수행하였다.

1. OTA 업데이트 프로토콜인 HTTP 메시지 생성.
2. 모의 IoT 헬스케어 기기에 생성한 HTTP 메시지 전송.
3. IoT 헬스케어 기기로부터 HTTP 응답 메시지 수신.
4. 수신한 응답 메시지의 상태 코드가 200 OK인 경우 더미 프로그램 설치 완료.
5. 수신한 응답 메시지의 상태 코드가 401 Unauthorized인 경우 이진코드에서 추출한 문자열을 바탕으로 권한 인증 정보 변경 및 HTTP 메시지 재생성.
6. 200 OK 응답 메시지가 수신 될 때 까지 2~5번 과정 반복.

첫 번째 과정에서 생성하는 HTTP 메시지는 스니핑 과정에서 획득한 HTTP 헤더 속성 중 권한 인증에 사용되는 Authorization 속성을 포함한 모든 헤더 속성과 동일하게 설정한다. 더미 프로그램은 HTTP의 바디에 싣는다.

두 번째 과정에서 HTTP 메시지 전송은 스니핑 과정에서 획득한 전송 URI와 포트 번호 등의 네트워크 정보를 사용한다.

세 번째 과정에서 401 Unauthorized 응답 메시지가 수신되었다면 이전 업데이트 과정에서 보안 목적으로 권한 인증 정보가 변경된 것을 의미한다. 이 경우 다섯 번째 과정을 따라 이진코드 분석 과정에서 획득한 문자열들을 권한 인증에 사용되는 ID와 password의 후보군으로 사용하여 HTTP 헤더의 Authorization 속성 값을 변경한 메시지를 다시 생성한다.

다섯 번째 과정과 여섯 번째 과정은 인증 권한을 얻고 더미 프로그램을 설치하기 위한 무차별 대입 공격 과정이다.

모의실험에서는 업데이트 과정에서 업데이트 권한 인증 password가 변경되었기 때문에 5번 및 6번 과정도 수행하였고, 그 결과 변경된 password인 \x02\0x00SN205315!을 찾아 더미 프로그램 설치를 완료할 수 있었다. 더미 프로그램이 설치된 모의 IoT 헬스케어 디바이스는 심박 측정 및 측정 데이터 전송이 중지되고 OTA 업데이트를 담당하는 기능도 제거되어서 더 이상 무선 인터페이스로 프로그램을 업데이트할 수 없었다. 따라서 기기에 유선 인터페이스 등 유지 보수를 위한 다른 수단이 없다면 기기의 사용이 영구적으로 불가능해졌다.

Ⅴ. 결론 및 향후 과제

본 논문에서는 IoT 헬스케어 기기가 보안이 적용되지 않은 OTA 업데이트를 사용할 때 발생할 수 있는 취약점을 분석하고 직접 모의 공격하였다. 네트워크 스니핑을 통해 원격에서 기기를 업데이트할 때 사용되는 프로토콜과 업데이트를 위해 전송되는 소프트웨어의 이진 코드를 탈취하였고, 탈취한 이진 코드를 분석하여 주요 정보를 포함한 문자열을 추출하였다. 그 결과 OTA 업데이트 프로토콜을 모방하고 업데이트 인증 권한 정보를 활용하여 더미 소프트웨어를 모의 IoT 헬스케어 기기에 설치함으로써 기기의 모든 기능을 정지시킬 수 있었다.

IoT 헬스케어 기기에서 안전한 OTA 업데이트를 사용하기 위해서는 다음과 같은 특징을 가진 보안 방법이 마련되어야 한다.

첫째로, OTA 업데이트 보안 절차는 경량이어야 한다. PC나 스마트폰에서 활용되고 있는 SSL과 같은 기존의 인증서 기반 보안을 모든 IoT 헬스케어 기기에 적용할 수는 없기 때문에, 비교적 적은 성능으로도 일정 수준의 보안 강도를 제공하는 경량화된 OTA 업데이트 보안 절차가 필요하다.

둘째, 소프트웨어의 역공학을 막거나 어렵게 할 수 있어야 한다. OTA 업데이트 과정에서 소프트웨어의 이진 코드가 유출되면 기기의 동작 방식은 물론 2차 공격에 필요한 정보들을 추출할 수 있기 때문에 이를 방지하기 위한 기능이 포함되어야 한다.

셋째, 소프트웨어 변조를 검출할 수 있어야 한다. 이를 통해 비인증 게이트웨이에서 IoT 헬스케어 기기에 악성 소프트웨어를 업데이트하는 것을 막을 수 있으며 IoT 헬스케어 기기로 사용자에게 위해를 가하거나 사용자 관련 정보 유출 및 주변 기기의 공격에 사용하는 등 2차 공격을 방지할 수 있다.

본 논문에서는 OTA 업데이트의 취약점을 모의실험을 통하여 구체적으로 분석하고 확인하였다. 향후 과제로 후속 논문에서는 본 논문에서 확인한 취약점을 해결하기 위한 보안 알고리즘을 제시하고 이를 구현한 후에 실제 시스템에 적용하여 본 논문에서 제시한 취약점을 해결할 예정이다.

References

[1] IoT in Healthcare Market by Component (Medical Device, Systems & Software, Services, and Connectivity Technology), Application (Telemedicine, Connected Imaging, and Inpatient Monitoring), End User, and Region - Global Forecast to 2025, https://www.alliedmarketresearch.com/iot-healthcare-market. [accessed: Dec. 19, 2020]

[2] Dong-Gyu Jeon, "A Study on IoT-Related Industry Trend", Korea Institute of Information Technology Magazine, No. 1, pp. 31-37, Jun. 2017.

[3] M. M. Alam, H. Malik, M. I. Khan, T. Pardy, A. Kuusik and Y. Le Moullec, "A Survey on the Roles of Communication Technologies in IoT-Based Personalized Healthcare Applications", IEEE Access, Vol. 6, pp. 36611-36631, Jul. 2018.

[4] P. Varga, S. Plosz, G. Soos and C. Hegedus, "Security threats and issues in automation IoT", 2017 IEEE 13th International Workshop on Factory Communication Systems (WFCS), Trondheim, 2017, pp. 1-6, Jul. 2017.

[5] H. Jayakumar, A. Raha, Y. Kim, S. Sutar, W. S. Lee, and V. Raghunathan, "Energy-efficient system design for IoT devices", 2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC), Macau, pp. 298-301, Mar. 2016.

[6] Woosik Lee, Hoejung Jung and Namgi Kim, "A Memory-based Transmission Power Control Algorithm in Wireless Body Sensor Networks", The Journal of Korean Institute of Information Technology, Vol. 14, No. 6, pp. 95-102, Jun. 2016

[7] Sukhoon Lee, Kwangsu Kim, and Dongwon Jeong. "A Data Transmission Mode Change Method for Improving Energy Efficiency in IoT Environments", Journal of Advanced Information Technology and Convergence, Vol. 10, No 1, pp. 57-69, Jul. 2020.

[8] Hae-Won Choi, Sangjin Kim and Myungchun Ryoo, "Cryptanalysis and Solution on Secure Communication Scheme for Healthcare System using Wearable Devices", Journal of Digital Convergence, Vol. 17, No 2, pp. 187-194, 2019.

[9] Se-hwan Park, Yee-nam Kwon, In-suk Song, Tae-wha-Lee, Sun-miKim, You-sik Hong, Young-hyun Lee and Jong-yun Kim, "Industry Environment Analysis of Smart Health Care", Proceedings of KIIT Conference, pp. 445-446, Dec. 2017.

[10] Yong-Hee Jeon, "A Study on the Security Modeling of Internet of Things(IoT)", The Journal of Korean Institute of Information Technology, Vol. 15, No. 12, pp. 17-27, Dec. 2017.

[11] G. Kambourakis, C. Kolias and A. Stavrou, "The Mirai botnet and the IoT Zombie Armies", MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM), Baltimore, MD, 2017, pp. 267-272, Oct. 2017.

[12] V. Shakhov, I. Koo and A. Rodionov, "Energy exhaustion attacks in wireless networks", 2017 International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON), Novosibirsk, 2017, pp. 1-3, Sept. 2017.

[13] Justice Owusu Agyemang, Jerry John Kponyo, Griffith Selorm Klogo, and Joshua Ofori Boateng, "Lightweight rogue access point detection algorithm for WiFi-enabled Internet of Things(IoT) devices", Internet of Things, Volume 11, Sep. 2020.

[14] Z Cekerevac, Z Dvorak, L Prigoda and P Cekerevac, "Internet of things and the man-in-the-middle attacks – security and economic risks", MEST Journal, Vol. 5, No 2, pp. 15-25, Jul. 2017.

저자소개

전 형 석 (Hyeongseok Jeon)

2014년 2월 : 경북대학교
심화컴퓨터공학(공학사)
2016년 3월 : 경북대학교
대학원 컴퓨터학부(공학석사)
2016년 3월 ~ 현재 : 경북대학교
대학원 컴퓨터학부 박사과정
관심분야 : 의료정보, IoT,
의료정보보안

이 성 기 (Sungkee Lee)

1979년 2월 : 서울대학교
전기공학(공학사)
1981년 2월 : 서울대학교
전기공학(공학석사)
1990년 8월 : University of Utah
컴퓨터과학(공학박사)
1990년 ~ 현재 : 경북대학교
컴퓨터학부 교수
관심분야 : 의료정보, 국제의료표준, 의료정보보안