저사양 IoT 디바이스의 안전한 펌웨어 업데이트를 위한 시스템 설계 및 구현

이기영O 이기웅 김병선 조진성
경희대학교 컴퓨터공학과
lkysecurity@khu.ac.kr, lkw1023@khu.ac.kr, ykbs0903@khu.ac.kr, chojs@khu.ac.kr

System Design and Implementation for Secure Firmware Update in a Lightweight IoT Device

Kiyeong LeeO Kiwoong Lee Byoungseon Kim Jinsung Cho
Department of Computer Science and Engineering, KyungHee University

요 약

저사양 IoT 디바이스의 문제점은 현존하는 다양한 보안 솔루션들을 적용하기에는 많은 제약이 있다는 점이다. 이는 대부분의 보안 솔루션들이 고성능 PC 환경을 대상으로 구현되기 때문이며, 이러한 한계는 IoT 디바이스의 기술적인 보안 취약점과 다양한 보안 공격들이 꾸준히 증가하는 원인이 되고 있다. 본 논문에서는 저사양 IoT 디바이스의 제약적인 환경에서 적용 가능한 안전한 펌웨어 업데이트 시스템 설계를 제안하고 각종 공격 방법들을 대상으로 본 논문에서 제안한 시스템의 성능을 검증한다.

1. 서 론

범용(COTS; Commercial Off-The-Shelf) IoT 디바이스란 시중에서 쉽게 구매 할 수 있으며 사용자가 다양한 기능을 추가하여 개발하거나 사용 할 수 있는 Open Source 기반의 하드웨어 플랫폼이다. 이 하드웨어 플랫폼의 특징은 다양한 서비스 분야에 활용 될 수 있도록 저전력, 가격 제약, 소형화 형태로 구성되어 있다[1]. 그러나, 저사양 IoT 디바이스의 문제점은 현존하는 다양한 보안 솔루션들을 적용하기에 많은 제약들을 가지고 있다는 점이다. 이러한 제약들로 인해 각종 취약점들이 발견되고 있다.

저사양 IoT 디바이스에 위협이 되는 취약점 중, 펌웨어 중간자 공격(Man-In-The-Middle), 펌웨어 롤백 공격(Rollback)과 같은 펌웨어 교체 공격(Firmware Replacement Attack)은 적법한 펌웨어가 아닌 변조된 펌웨어를 디바이스에 강제로 업데이트하는 공격이다[2]. 이러한 공격을 받은 디바이스는 공격자로부터 시스템 제어권을 탈취당해 디바이스 내에 저장된 중요 데이터와 개인 정보들이 유출되는 상황이 발생하고 지속적인 2차, 3차 공격으로부터 노출된다.

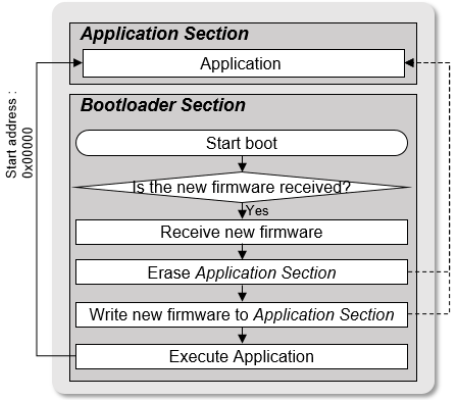
본 연구에서는 저사양 IoT 디바이스 플랫폼 내 펌웨어 이미지 교체/변조를 통해 발생하는 문제점들을 차단하기 위해 디바이스 보안 플랫폼을 개발하였고 개발된 플랫폼을 기반으로 안전한 펌웨어 업데이트(Secure Firmware Update) 시스템을 제안하고 취약점 공격에 대한 대응 시나리오를 통해 제안한 시스템의 성능을 검증한다.

본 논문의 목차는 다음과 같다. 2장에서는 일반적인 저사양 IoT 디바이스의 업데이트 과정과 임베디드 보안 칩인 Secure Element(SE)에 대한 관련 연구 내용을 설명한다. 3장에서는 제안하는 시스템에 대해 설명하고 4장에서는 제안한 시스템을 기반으로 디바이스 플랫폼 구현 설명 및 보안 위협으로부터의 성능을 검증한다. 5장에서는 결론 및 향후 계획을 제시한다.

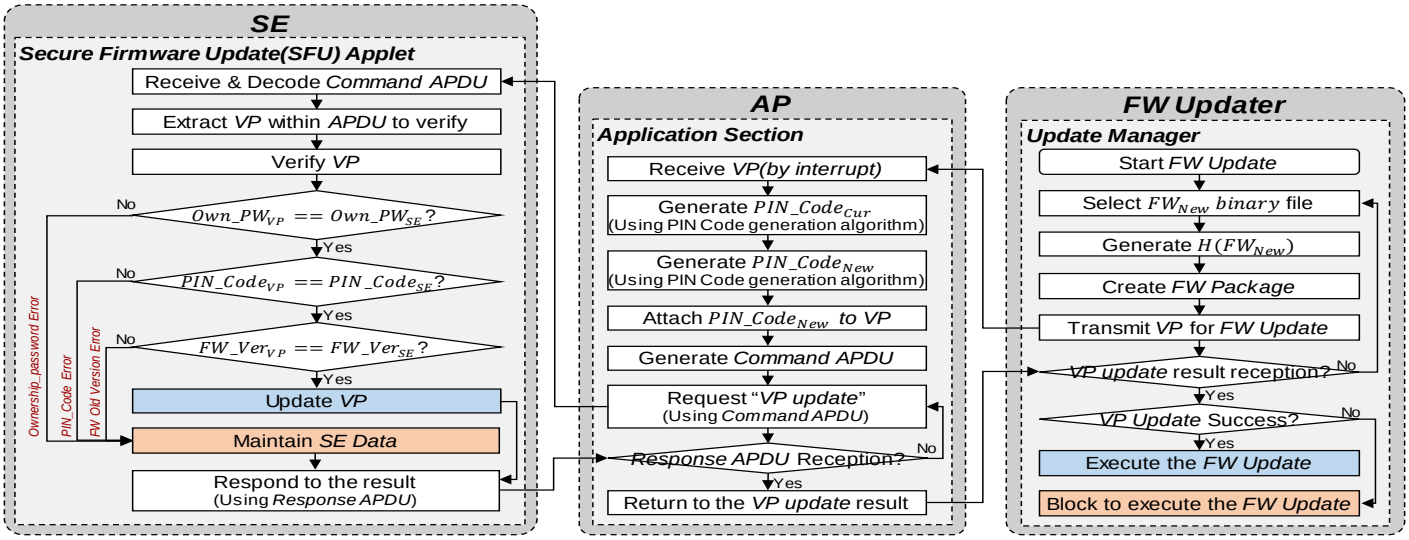
"본 연구는 삼성전자의 지원으로 수행되었음."

2. 관련 연구

그림 1과 같이 저사양 IoT 디바이스는 전원이 인가되면 부트로더(Bootloader)가 실행되고 외부에서 시리얼 포트를 통해 펌웨어 이미지를 수신하기 위한 대기 상태가 된다. 펌웨어 이미지를 수신하면, 부트로더는 업데이트 진행을 위해 기존 어플리케이션 영역의 어플리케이션 펌웨어를 삭제하고 새로운 펌웨어의 바이너리 코드를 삽입한다. 삽입이 완료되면, 어플리케이션의 시작 주소로 점프하여 펌웨어를 실행한다. 이러한 일반적인 업데이트 과정은 외부에서 업데이트를 시도하는 주체에 대한 인증 체계를 갖추고 있지 않고



[그림 1] 일반적인 업데이트 과정



[그림 2] 안전한 펌웨어 업데이트 시스템 구성

업데이트를 진행하는 취약점을 가지고 있다[3].

한편, 여러 보안 단체에서 개발한 하드웨어 보안 칩은 기본적으로 디바이스 인증을 제공하고, 저장된 개인정보 및 데이터를 보호하기 위한 암호화 키를 생성하며, 생성된 키는 보안 칩 내부에 안전하게 저장되어 소프트웨어 기반 악성 프로그램으로부터 보호하는 기능을 제공한다. 또한, 칩 내부에는 별도의 운영체제(i.e., JavaCard OS)가 포함되어 있어 기존 시스템과 통합하기 쉽고, 시스템의 요구사항에 맞춰 칩 내부에 개별적인 보안 솔루션을 프로그래밍 할 수 있다는 장점이 있다. 현재 SE를 활용하여 SIM-Card, 임베디드 어플리케이션(Embedded Application)과 같은 경량 환경에서의 보안 기능을 강화하고 있다[4].

3. 제안하는 시스템

본 장에서는 SE를 통해 안전한 펌웨어 업데이트(Secure Firmware Update)의 시스템 설계와 각 구성 요소들의 동작 과정을 설명한다.

3.1 시스템 설계

그림 2와 같이 펌웨어 업데이트는 크게 FW(Firmware) Updater, AP(Application Processor), SE(Secure Element)로 구성되어 있으며, 각각의 기능은 다음과 같다.

• **FW Updater:** FW Updater는 펌웨어 업데이트를 수행하는 주체로써, 총 2단계의 업데이트 과정을 수행한다. 첫 번째 단계는 새로운 펌웨어의 정보들을(i.e., Hash, Password, Version) SE의 Data Section에 업데이트를 수행한다. 두 번째 단계는 새로운 펌웨어를 AP에 업데이트하는 단계를 수행한다.
• **AP:** AP는 실제 펌웨어가 실행되는 디바이스(i.e.,아두이노)를 의미하며, 실제 공격의 주 타겟이 되는 대상이다.
• **SE:** FW Updater가 시도하는 첫 번째 업데이트 과정에서 패스워드 인증, 버전 검증, 펌웨어 무결성 검증을 수행함으로써 안전한 업데이트 환경을 제공한다.

각 구성을 통한 업데이트 절차는 크게 Secure Firmware Update 과정과 Secure Boot 과정으로 구분된다. Secure Boot 과정은 Secure Firmware Update가 진행된 이후에 수행된다. 각 과정에 대한 설명은 다음과 같다.

3.2 Secure Firmware Update 동작

FW Updater는 앞서 언급한 두 단계의 업데이트 과정을 위해 FW Package를 생성한다. FW Package는 VP(Verification Property)와 새로운 펌웨어 바이너리 이미지를 가지고 있다. VP는 새로운 펌웨어 바이너리의 해시[2], 펌웨어의 버전, 업데이트 단계에서 SE와의 인증 패스워드인 Ownership Password 정보들을 가지고 있다. FW Updater는 첫 번째 업데이트 과정을 수행하기 위해 VP를 AP에게 UART(Universal Asynchronous Receiver & Transmitter)로 전송한다. AP는 UART로 전송 받은 데이터들을 인터럽트를 발생시켜 우선적으로 처리한다. AP는 수신한 데이터들 중에 새로운 펌웨어의 해시 값을 Key Generation algorithm을 기반으로 Secure Boot 단계에서의 인증 패스워드인 핀 코드(PIN Code)를 생성하여 VP에 추가한다. 이때의 핀 코드는 제조사만이 알고 있는 알고리즘으로써 펌웨어의 해시 값을 이용해 생성한다. 핀 코드 알고리즘은 외부로 노출되지 않도록 비공개 형태로 관리한다. AP는 추가한 VP의 검증을 위해 데이터 전송 프로토콜인 APDU(Application Protocol Data Unit) 구조를 만들고 추가한 VP를 페이로드(Payload)에 저장한 후, SE에게 전송한다. SE는 AP로부터 전송 받은 APDU에서 VP를 추출하고 VP 내의 정보들을 Data Section에 있는 기존 펌웨어의 정보들과 비교 검증을 수행한다. SE는 VP와의 비교 검증이 성공하면 성공 결과를 AP에게 APDU 형식으로 전송한다. AP는 SE로부터 수신한 결과를 FW Updater에게 재전송 한다. FW Updater는 AP로부터 수신한 결과가 성공하면 업데이트의 두 번째 단계로 FW Package에

있는 새로운 펌웨어 바이너리 이미지를 이용해 업데이트를 수행하고 실패하면 펌웨어 업데이트를 중단한다[3].

3.2 Secure Boot 동작

Secure Boot는 AP에서 실행되고 있는 서비스 어플리케이션 펌웨어의 무결성 검증을 수행한다. AP는 무결성 검증 대상인 현재 펌웨어의 해시를 만들고 핀 코드를 생성한다. AP는 생성한 해시와 핀 코드를 APDU에 저장한 후, SE에게 펌웨어의 무결성 검증 요청한다. SE는 수신한 APDU의 핀 코드와 해시 값을 추출해 Data Section에 있는 데이터들과 비교 검증을 수행한다. SE는 비교 검증의 결과를 AP에게 APDU 형식으로 전송하고 AP는 수신한 결과가 성공이면 정상적인 부팅을 수행하고 실패가 되면 부팅을 중단한다.

4. 구 현

4.1 SArduino

본 연구에서는 안전한 펌웨어 업데이트 시스템을 기반으로 그림 3과 같이 아두이노 종류인 Arduino Mega2560 위에 와이파이 쉴드(Wifi-Shield), SE 칩인 Infineon사의 Optiga TrustP를 장착한 SArduino를 구현하였다.

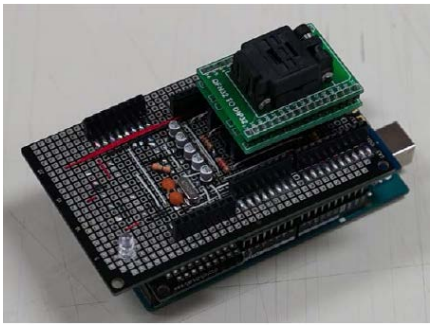
4.2 보안 위협 대응

본 장에서는 구현한 SArduino를 대상으로 보안 위협 공격들을 수행하였으며, 공격들에 대한 대응 결과는 다음과 같다.

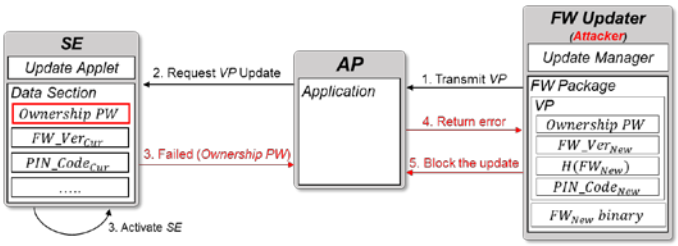
- **중간자(Man-In-The-Middle) 공격:** 그림 4와 같이 변조된 펌웨어를 업데이트 시도하려는 공격자는 SE의 VP 검증 과정에서 SE의 Ownership Password를 알지 못하기 때문에 검증이 실패하고 업데이트 진행이 불가능하게 된다.
- **펌웨어 롤백(Rollback) 공격:** 그림 5와 같이 적법한 이전 버전의 펌웨어를 이용한 업데이트는 SE의 Version 검증에서 하위 버전 임을 판별하기 때문에 검증이 실패한다. 따라서, 공격자로부터의 펌웨어 업데이트가 차단된다.
- **펌웨어 교체 공격(Firmware Replacement Attack):** 변조된 펌웨어로 교체한 이후 부팅을 시도하면, Secure Boot 과정에서 의 펌웨어 무결성 검증에서 핀 코드가 불일치 하기 때문에 실행이 중단된다.
- **부트로더 교체 공격(Bootloader Replacement Attack):** AP자체를 교체하여 부팅을 시도하려는 공격자는 SE에 접근하는 방식을 알지 못하기 때문에 접근이 실패하고 펌웨어 실행을 중단한다.

5. 결론 및 향후 계획

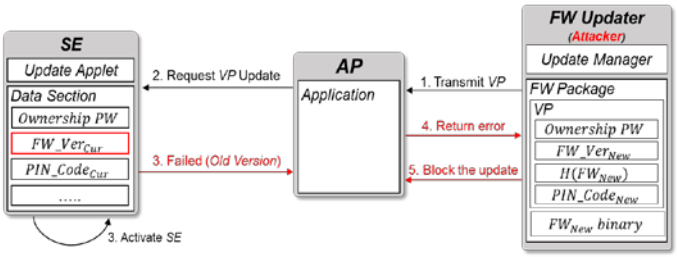
본 논문에서는 저사양 IoT 디바이스의 안전한 펌웨어 업데이



[그림 3] SArduino 하드웨어 프로토타입



[그림 4] 중간자 공격 시도 과정



[그림 5] 펌웨어 롤백 공격 시도 과정

트를 위한 시스템 설계를 제안하였다. 또한, 각종 보안 위협에 대한 대응을 통해 본 연구의 성능을 검증하였다.

향후 연구 계획은 AP와 SE간의 데이터들이 부채널 공격(Side Channel Attack), 재전송 공격(Replay Attack)과 같이 도청으로부터 보호하기 위한 연구를 진행할 것이다.

참 고 문 헌

[1] Sachin Babar, Antonietta, Stango, Neeli Prasad, Jaydip Sen, Ramjee Prasad, “Proposed Embedded Security Framework for Internet of Things(IoT)”, IEEE, Accession Num. 12094585, 2011

[2] Atmel, “Atmel AT02333: Safe and Secure Bootloader Implementation for SAM3/4”, Atmel Application Note specification, 2013

[3] Atmel, “AVR109: Self Programming”, Application Note specification, 2004

[4] Pascal Urien, “Three Innovative Directions Based On Secure Element for Trusted and Secured IoT System”, IEEE, Accession Num. 16557955, 2016