https://doi.org/10.5392/JKCA.2021.21.06.051

임베디드 디바이스 보안을 위한 SDN 적용 시 고려사항

Considerations for Applying SDN to Embedded Device Security

구금서*, 심갑식**
경상국립대학교 교양학부*, 경상국립대학교 휴먼헬스케어학과**

GeumSeo Koo(goodman4009@gmail.com)*, Gabsig Sim(gssim@gnu.ac.kr)**

요약

사물인터넷과 빅데이터 그리고 인공지능으로 상징되는 4차 산업혁명시대에 다양한 임베디드 디바이스가 기하급수적으로 증가하고 있다. 이러한 디바이스는 낮은 사양임에도 통신 기능을 보유하고 있어서 개인 정보 유출 가능성이 높아지고 있으며 보안의 위협 또한 증가하고 있다. 임베디드 디바이스는 하드웨어부터 네트워크를 통한 서비스까지 대부분의 단계에서 보안 이슈가 발생 가능하다. 또한 저사양과 저전력 등 자원 제약의 특징을 가지며 관련 기술의 표준화가 이루어지지 않은 상황이므로 일반적인 보안 기법을 적용하기에는 어려움이 따른다. 본 연구에서는 임베디드 디바이스에 SDN 적용 시 취약점과 발생 가능한 문제점과 고려사항을 제시하였다. 하드웨어 관점에서 와이파이 칩과 블루투스의 문제, 오픈플로우 구현상의 문제, SDN 컨트롤러 및 구조적 특성에 따른 사례를 고려하여 제시하였다. SDN은 데이터 플레인과 제어 플레인을 각각 분리하여 둘 사이에 표준화된 인터페이스를 제공하여 통신을 효율적으로 제어할 수 있으며 빠른 변화에 대응하기 어려운 기존 네트워크 기술에서의 보안의 한계에 대응할 수 있다.

■ 중심어 : | 임베디드 디바이스 | 디바이스 보안 | 소프트웨어 정의 네트워크 | 오픈플로우 | 라즈베리파이 |

Abstract

In the era of the 4th industrial revolution symbolized by the Internet of Things, big data and artificial intelligence, various embedded devices are increasing exponentially. These devices have communication functions despite their low specifications, so the possibility of personal information leakage is increasing, and security threats are also increasing. Embedded devices can have security issues at most levels, from hardware to services over the network. In addition, it is difficult to apply general security techniques because it has characteristics of resource constraints such as low specifications and low power, and the related technology has not been standardized. In this study, we present vulnerabilities and possible problems and considerations in applying SDN to embedded devices in consideration of structural characteristics and real-world discovered cases. This study presents vulnerabilities and possible problems and considerations when applying SDN to embedded devices. From a hardware perspective, we consider the problems of Wi-Fi chips and Bluetooth, the problems of open flow implementation, SDN controllers, and examples of structural properties. SDN separates the data plane and the control plane, and provides a standardized interface between the two, enabling efficient communication control. It can respond to the security limitations of existing network technologies that are difficult to respond to rapid changes.

■ keyword : | Embedded Devices | Device Security | SDN | OpenFlow | RaspberryPi |

* 이 논문은 2020~2021년도 경상국립대학교 대학회계 연구비 지원에 의하여 연구되었음

접수일자 : 2021년 01월 27일
심사완료일 : 2021년 04월 01일
수정일자 : 2021년 04월 01일
교신저자 : 심갑식, e-mail : gssim@gnu.ac.kr

I. 서 론

사물인터넷(IoT: Internet of things)과 인공지능(AI: Artificial intelligence) 그리고 빅 데이터(Big data)로 상징되는 4차 산업혁명시대에 다양한 임베디드 디바이스와 모바일 기기가 기하급수적으로 증가하고 있다. 최근의 임베디드 디바이스는 일반 데스크탑 컴퓨터 대비 상대적으로 낮은 사양임에도 인터넷 등의 통신 기능을 보유하고 있어 생활 전반의 개인 정보의 유출 가능성과 보안의 위협이 지속적으로 증가하고 있다[1][2]. 임베디드 디바이스는 물리적인 하드웨어 단계에서부터 네트워크를 통한 서비스까지 대부분의 단계에서 보안 이슈가 발생 가능하다. 임베디드 디바이스의 일반적인 특징은 범용 컴퓨터 대비 저사양과 저전력 등 자원 제약의 특징을 가지며 관련 기술의 표준화가 이루어지지 않은 상황이므로 일반적인 보안 기법을 적용하기에는 어려움이 따른다.

기존 임베디드 디바이스에 보안을 적용하기 위해서 SSL(Secure Socket Layer), WTSL(Wireless Transport Layer Security), IPSec(Internet Protocol Security) 등의 보안 기법을 적용하지만 자원 제약의 특성과 빠르게 증가하는 트래픽 등의 환경에 대응하기 위해서 소프트웨어적인 측면에서 대응이 필요하다.

본 연구에서는 임베디드 디바이스에 SDN(Software Defined Network) 적용 시 발생 가능한 문제점과 고려사항을 구조적 특성 및 실제 발견된 사례를 기반으로 제시한다. SDN은 데이터 플레인(data plane)과 제어 플레인(control plane)을 각각 분리하여 둘 사이에 표준화된 인터페이스를 제공하여 통신을 효율적으로 제어할 수 있다. 변화에 대응하기 어려운 현 네트워크 기술의 한계에 대응할 수 있는 대안이 될 수 있다. 적용 대상 모델로는 IoT 관련 분야에서 많이 활용하는 초소형 컴퓨터인 라즈베리파이3 모델B이다. II장에서는 기존 연구에서 제안된 보안 기법과 임베디드 디바이스의 공격 유형 및 SDN의 기본 구조를 살펴본다. III 장에서는 SDN의 구조 및 라즈베리파이에 OpenFlow 스위치를 설치하고 실행하는 방법을 살펴보고 임베디드 디바이스에 적용 시 문제점과 요구사항을 제시한다. IV장에서는 분석한 요구 사항에 대한 결론을 정리하고 향후 연구방향을 제시한다. 임베디드 디바이스가 폭발적으로 증가하고 있는 시점에서 자원 제약의 특징을 가지는 임베디드 디바이스에 SDN 적용 시 취약점과 고려 사항을 참고하여 보다 안전하게 시스템을 보호할 수 있기를 기대한다.

II. 관련연구

1. 보안기법

사물인터넷 시대의 도래로 관련 임베디드 디바이스의 수요가 증가하고 있으며 통신 기능의 추가로 개인 정보의 유출 가능성 또한 증가하고 있다. 기존 범용 데스크탑 컴퓨터 기준의 보안 기법은 자원제약의 특징이 있는 임베디드 환경과 차이가 있으므로 동일한 환경에서 적용에 한계가 존재한다. 임베디드 디바이스의 일반적인 특징은 저사양, 저전력, 소형, 저가격 등이다. 이러한 특징으로 상대적으로 많은 보안의 취약점을 가지고 있다.

보안의 기본적인 목표 사항은 데이터의 기밀성(Confidentiality: 노출(Disclosure)로 부터의 보호), 데이터 무결성(Data integrity: 변경(Alteration)으로 부터의 보호), 가용성(Availability: 파괴(Destruction)로 부터의 보호)의 3가지를 의미하며 줄여서 CIA라고 부른다.

임베디드 디바이스의 보안에 적용하는 기술 중 SSL(Secure Socket Layer)[3]은 브라우저(Browser)와 서버(Server)간 암호화를 지원하는 비대칭 알고리즘 기반의 기술로 비밀키를 서버에 보관하여 비밀키를 가지지 못하면 암호를 해제하지 못하는 기술도 사용된다. SSL은 넷스케이프(Netscape)사에서 개발되어 사용되었으며 국제 표준으로 승격되어 TLS(Transport Layer Security)로 이름이 변경되었으며 무선 인터넷 환경에서는 협대역 통신 채널을 위해 최적화된 WTLS(Wireless Transport Layer Security)가 사용된다.

IPSec(Internet Protocol Security)[4]은 IETF에서 표준화한 네트워크 계층의 보안 프로토콜로서 암호화 기술을 이용하여 IP 패킷(Packet) 단위로 데이터 변조

방지 및 은닉 기능을 제공한다. IPSec의 기능은 AH, ESP 2개의 프로토콜로 구성되며 AH와 ESP와 함계 암호화 키를 관리하는 IKE의 3가지로 구분 짓는다. AH(Authentication Header)는 데이터 송신자의 인증을 허용하고 데이터 무결성을 검사하여 스니핑(Sniffing) 등 해당 데이터를 다시 보내는 재생 공격을 막는다. ESP(Encapsulating Security Payload)는 송신자의 인증과 데이터 암호화를 제공하여 기밀성을 유지한다. IKE(Internet Key Exchange)는 키교환으로 통신 시간의 보안 설정을 유지하며 8시간의 보안을 유지한다.

OSCAR(Object Security Architecture for the Internet of Things)[5]에서는 자원 제약적인 CoAP 노드 환경에서 안전한 자원(resource)의 요청과 응답을 위한 DTLS 기반의 안전한 채널의 통신 방법을 제안하였다. OSCAR는 생산자(Producer)와 소비자(Consumer) 사이에 인증 서버(Authorization Server)를 두어서 디지털 서명 및 암·복호화 방식을 통해서 두 노드 사이의 인증, 리소스 요청 및 응답을 빠르고 안전하게 수행하는 인터페이스 역할을 수행한다. 무선센서네트워크(WSNS: Wireless Sensor Networks) 환경에서 두 노드 사이에 교환되는 데이터의 도청과 악용으로부터 저사양의 임베디드 디바이스를 보완하기 위한 수단으로 경량화 암호화 알고리즘을 이용해서 교환되는 데이터의 암·복호화 방법을 제안하였다.

암호화 기술 중 임베디드 디바이스의 취약점을 이용한 공격 방법으로 펌웨어 롤백 및 펌웨어 중간자 공격 등이 있으며 시스템 제어권을 탈취해 디바이스의 중요 데이터와 개인 정보가 유출된다.

표 1. Common attack method

Software Attack	Hardware Attack
virus, worm	de-packaging
software weakness	SPA, DPA
buffer overflow	timing
low cost, easy to access.	high cost, difficulty access.

2. 임베디드시스템의 공격 분류

일반적인 임베디드 환경에서의 보안 공격 방식을 소프트웨어 관점과 하드웨어 관점에서 살펴본다[6]. 임베디드 디바이스에서의 일반적인 특징과 공격 유형은 [표 1]에 정리하였다. 먼저 소프트웨어 공격을 살펴보면 상대적으로 낮고 쉬운 접근을 보인다. 바이러스(Virus)와 웜(Worm)을 통한 BIOS 공격을 예로 들 수 있다. 초기 BIOS는 ROM(Read Only Memory)에 저장되어 코드 등이 갱신 될 수 없었으나 오늘날의 임베디드 디바이스는 대부분 플래시 ROM의 사용으로 소프트웨어 재작성이 가능하다. 이로인해 바이러스 혹은 악성코드를 통한 공격이 가능해졌다.

버퍼 오버플로(buffer overflow) 공격은 운영체제와 응용 소프트웨어에서 발생하며 버퍼가 경계 검사를 수행할 때 발생한다. 정해진 메모리의 범위를 넘치게(overflow)해서 원래의 리턴 주소를 변경해서 의도하지 않은 임의의 프로그램이나 함수를 실행시키는 기법이다. 버퍼 경계는 부정확한 루프 경계, 포맷 스트링 공격 등으로 인해 위반될 수 있으며 스택, 힙 그리고 함수 포인터에 오버라이트(Overwrite)가 발생한다. 대응 방안으로 스택 가드는 무결성 체크용 값인 카나리(canary)를 복귀주소와 변수 사이에 삽입하여 오버플로가 발생하면 카나리 값을 확인하여 복귀 주소로의 호출을 방지한다. 또한 함수의 시작시에 복귀 주소를 Global RET에 저장 후 함수 종료 시 저장된 값과 스택의 RET값을 비교하여 값이 다를 경우 오버플로우로 처리하여 실행을 중단한다. 이외에 메모리 공격을 방어하기 위해 주소 공간 배치를 난수화하여 실행시마다 메모리 주소를 변경하여 오버플로우 발생 시에 특정 주소 호출을 차단하는 ASLR(Address Space Layout Randomize)이 있다.

다음으로 하드웨어 관점에서의 공격으로 임베디드 하드웨어 공격의 대표적인 예로 de-packaging을 들 수 있다. 칩의 패키지를 제거하여 칩 내부의 레벨 구조를 노출하여 공격자이 원하는 정보를 획득할 수 있다.

전력 분석 공격은 암호키와 관련된 데이터를 처리할 때 칩이 사용하는 전력량을 비교 분석하여 동작을 추론 가능하다. 대표적으로 단순 전력 분석인 SPA(Simple Power Analysis)는 암호 관련 계산의 전력 프로파일을 활용한다. 차등 전력 분석인 DPA(Differential Power Analysis) 공격은 전력 소모 데이터로부터 암호키를 추론하기 위해 통계 분석을 적용한다.

타이밍 공격은 암호화 관련 계산 작업의 실행 시간이 데이터 의존적이라는 관측을 이용한다. 이외에 전압과 클록, 온도 등의 환경적인 조건에 의존하여 결점을 주입하여 공격하는 방식이 있다. 지금까지 일반적인 임베디드 디바이스의 공격 유형을 살펴보았다.

3. SDN(Software Defined Network)

소프트웨어 정의 네트워크(SDN: Software Defined Network)[7-11][17]는 소프트웨어를 기반으로 데이터 경로 설정 부분과 제어 파트를 분리하여 관리하므로 하드웨어 기반의 네트워크 제어에 비해 효율성이 높은 차세대 네트워크 기술이다. 네트워크 제어 기능이 물리적인 네트워크와 분리되어 있으므로 논리적으로 네트워크를 관리하고 제어할 수 있으며 다양한 네트워크 서비스를 수행할 수 있다. 개방된 API를 활용하여 네트워크를 기능별로 모듈화하면 소프트웨어로 구현이 가능하므로 임베디드 디바이스와 유사한 낮은 성능의 프로세서가 장착된 하드웨어에서 스위치로 활용 가능하다. SDN의 기본 구조는 [그림 1]과 같다.

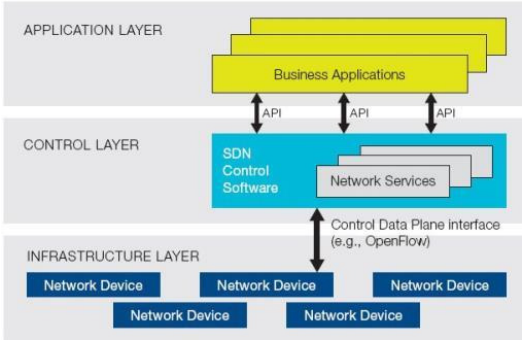

그림 1. SDN Architecture

SDN 환경을 만들려면 각 네트워킹 명령어를 원활하게 주고받을 수 있는 개방형 인터페이스가 만들어져야 하며 이를 위해서 오픈플로우(OpenFlow)가 사용된다. 오픈플로우(OpenFlow)[12][13]는 SDN을 구성하는 하부 요소로서 제어 기능을 가진 장비와 네트워킹 스위치 간 통신을 담당하는 표준 인터페이스에 해당하는 프로토콜을 말하며 계층기반으로 분리되어 별도의 프로토콜을 제공한다. 오픈플로우는 인터넷 스위치, 라우터 등의 제어 평면(Control Plane)에 탑재될 수 있으며, SDN 기술에 사용된다.

기존 네트워크와 SDN의 차이점을 [표 2]에 간략하게 정리 하였다.

표 2. Previous Network vs SDN

Section	Previous	SDN
Network Aspect	Hardware	Software
Composition Initiative	H/W vendor	user
Openness	close	open
Compatibility	Vendor Protocol	Standard Protocol
Manageability	Inefficiency/ High cose	Efficiency/ Reasonable
Technology Acceptance	Vendor	User
Market fairness	Monopoly	Fair competition

III. 제안내용

본 장에서는 SDN의 기능별 특징을 바탕으로 임베디드 디바이스에 적용 시 발생 가능한 문제점과 고려사항을 제시한다. 먼저 라즈베리파이 기반의 환경에서 미니넷(Mininet)을 사용하여 가상의 네트워크 환경을 구성하는 방법을 간략히 설명하고 실제 적용 시 발생 가능한 문제점을 살펴본다. 다음으로 SDN의 콤포넌트별 구조를 분석하여 라즈베리파이 환경에서 보안 적용 시 발생 가능한 취약점과 고려사항을 제시한다.

1. 라즈베리파이 환경에 미니넷 설치

임베디드 디바이스 중 하나인 라즈베리파이[14]는 영국 잉글랜드에 위치한 라즈베리 파이 재단에서 컴퓨터를 사용한 과학교육 목적으로 개발한 실습용 키트이면서 SBC(Single Board Computer)이다. 컴퓨터의 최소 구성 요소를 모두 갖추고 있어서 초소형 컴퓨터로 불리며 낮은 가격으로 사용에 부담이 없다. 라즈베리파이의 기본 운영체제는 리눅스 기반의 Raspberry Pi OS(이전에 Raspbian이라 부름, Debian Buster with Raspberry Pi Desktop, Release: February 12th 2020, Kernel Version: 4.19, Size: 2.983MB)를 제

공하며 라즈베리파이 공식 홈페이지(www.raspberrypi.org)[11]에서 무료로 다운로드 및 설치가 가능하다. 또한 'Raspberry Pi Imager'를 통한 설치도 제공한다. 라즈베리파이3 모델 B의 기본 사양은 [표 3]과 같다.

미니넷(Mininet)[15]은 일반적인 컴퓨터 환경 등에서 OpenFlow 네트워크를 생성하는 에뮬레이터 프로그램으로 쉽게 가상의 네트워크 환성 구성이 가능하며 VMware Workstation, VirtualBox 내에서 실행되는 Virtual Machine을 다운로드 가능하다. 라즈베리파이에 SDN 적용을 위해서 미니넷(mininet)을 설치하였다.는 방법을 간략히 설명하고 라즈베리파이 자체 환경에서 발생 가능한 문제점을 살펴본다.

설치 절차를 명령어 기반으로 간략히 설명하였다.

표 3. Raspberry Pi Specification

Name	Raspberry Pi 3 Model B
CPU	Broadcom BCM2387 SoC 1.2GHz ARM Cortex-A53 64bit(Quad-Core)
GPU	Broadcom Dual Core VideoCore IV
Memory	1GB LPDDR2 SDRAM
Memory Slot	Push/pull SDIO
Ethernet/WIFI	10/100 BaseT ethernet socket
USB 2.0	4 * USB 2.0 connector
I/O	26 GPIO, 1 Uart, 2PWM CSI & DSI, 1 SPI, 2 I2C, PCM/i@s,
GPIO	Connector 40-pin 2.54mm(100mil) expansion header
OS	Windows 10, Linux, Android
Power	Micro USB Socket 5V1, 2.5A

1단계: RaspberryPi의 Github에서 Mininet을 복제.

```
pi@raspberrypi1~$ sudo git clone
git://github.com/mininet/mininet.git
```

2단계: Util 디렉토리로 이동

```
pi@raspberrypi1~$ cd mininet/util/
```

3단계: 아래 옵션 중 하나를 사용하여 install.sh 스크립트를 실행

```
pi@raspberrypi1 ~/mininet/util $ sudo ./install.sh -fnv
```

Option 1: Basic Install:
Option 2: Install everything:

```
pi@raspberrypi1 ~/mininet/util $ sudo ./install.sh -a
```

위 스크립트로 Mininet을 설치를 완료하고 설정 사항을 테스트하기 위해서 아래의 명령을 수행한다.

```
pi@raspberrypi1 ~ $ sudo mn
```

수행 이후 단일 스위치와 두 개의 호스트로 구성된 토폴로지가 생성된다. 이후 호스트 간의 연결을 테스트하려면 "pingall" 명령을 사용하고 Mininet 종료시 "exit"를 사용한다. OpenFlow 기반의 SDN 적용으로 SDN Control Software는 스위치와 라우터 등 OpenFlow 지원 장비를 중앙 집중적으로 세밀하게 제어할 수 있으며 자동화 도구를 기반으로 비용의 축소와 안전성 및 신뢰성이 증가하게 된다.

라즈베리파이 기반으로 SDN 운영 시 하드웨어 단계부터 SDN 구조적 특징을 단계적으로 살펴보면서 고려사항을 제시한다.

2. SDN 아키텍처 분석

라즈베리파이에 오픈플로우(OpenFlow) 기반의 SDN을 적용 시 발생 가능한 문제점과 보안 고려사항을 [그림 2]의 SDN의 구조[17]와 역할별 특징으로 구분하여 제시한다.

오픈플로우의 경우는 SDN을 구현하는 대표적인 인터페이스로서 ONF(Open Networking Foundation)에서 개발한 표준이다. SDN 환경을 만들려면 각 네트워킹 명령어를 원활하게 주고받을 수 있는 개방형 인터페이스가 만들어져야 하며 이를 위해서 오픈플로우(OpenFlow)가 사용된다. 데이터 플레인(Data plane)과 제어 플레인(Control plane) 사이의 중계 역할을 수행하며 실제 packet routing과 별개로 처리되어 routing 방법을 프로그래밍 할 수 있다. 그로인해 라우팅 가격을 낮출 수 있으며 네트워크를 단순화 할 수 있다.

오픈플로우(OpenFlow)[12][13][18][19]는 SDN을 구성하는 하부 요소로서 제어 기능을 가진 장비와 네트워킹 스위치 간 통신을 담당하는 표준 인터페이스에 해당하는 프로토콜이다. 오픈플로우는 SDN을 지원하는 프로그래밍 가능한 API로서 소프트웨어와 스위치, 라우터 등 네트워크 장비에 직접 접속해서 포워딩(Forwarding)과 제어(Control) 기능을 계층구조로 분리하여 별도의 프로토콜을 제공한다. 또한 제조사 중심

의 라우팅(Routing) 알고리즘을 벗어난 경로 설정 등이 가능하므로 장비에 독립적으로 트래픽 제어가 가능하다.

오픈플로우는 컨트롤러와 스위치로 구성되며 제어 플레인(Control plane)은 소프트웨어로 컨트롤러로 구현하고 데이터 플레인(Data plane)은 오픈플로우 프토토콜을 펌웨어로 구성하는 L2 스위치 형태로 구현한다.

SDN은 오픈플로우(Openflow)와 같은 개방형 API를 통해 네트워크의 트래픽 전달 동작을 소프트웨어 기반인 SDN 컨트롤러에서 관리 및 제어한다.

SDN 계층별 기술 요소는 크게 SDN Layer와 SDN Interface로 나뉜다. SDN Layer의 기술요소로 SDN Application, SDN Controller, SDN Datapath로 나뉜다. SDN Application은 네트워크 요구 사항과 네트워크 동작을 NBI(Northbound Interface)를 통해서 SDN Controller에 직접적으로 전달하는 프로그램이다. 또한 하나의 SDN 애플리케이션 로직과 하나 이상의 NBI 드라이버로 구성되며 각각의 NBI 에이전트를 통해 하나 이상의 상위 수준 NBI를 제공한다.

SDN Controller는 SDN Application 계층의 요구사항을 SDN Datapath로 변환하고 SDN Application에 네트워크 추상화를 제공하는 역할을 수행하며 논리적으로 중앙 집중화된 엔티티이다. 또한 하나 이상의 NBI 에이전트와 SDN Control Logic 및 CDPI(Control to Data-Plane Interface) 드라이버로 구성된다.

SDN Datapath는 광고된 전달(Advertised Forwarding) 및 데이터 처리 기능의 가시화와 경합 없는 제어를 제공하는 논리적 네트워크 제어장치이며 CDP에이전트와 하나 이상의 트래픽 포워딩 엔진 세트와 0개 이상의 트래픽 처리 기능으로 구성된다.

SDN Interface의 기술요소로는 CDPI(Control to

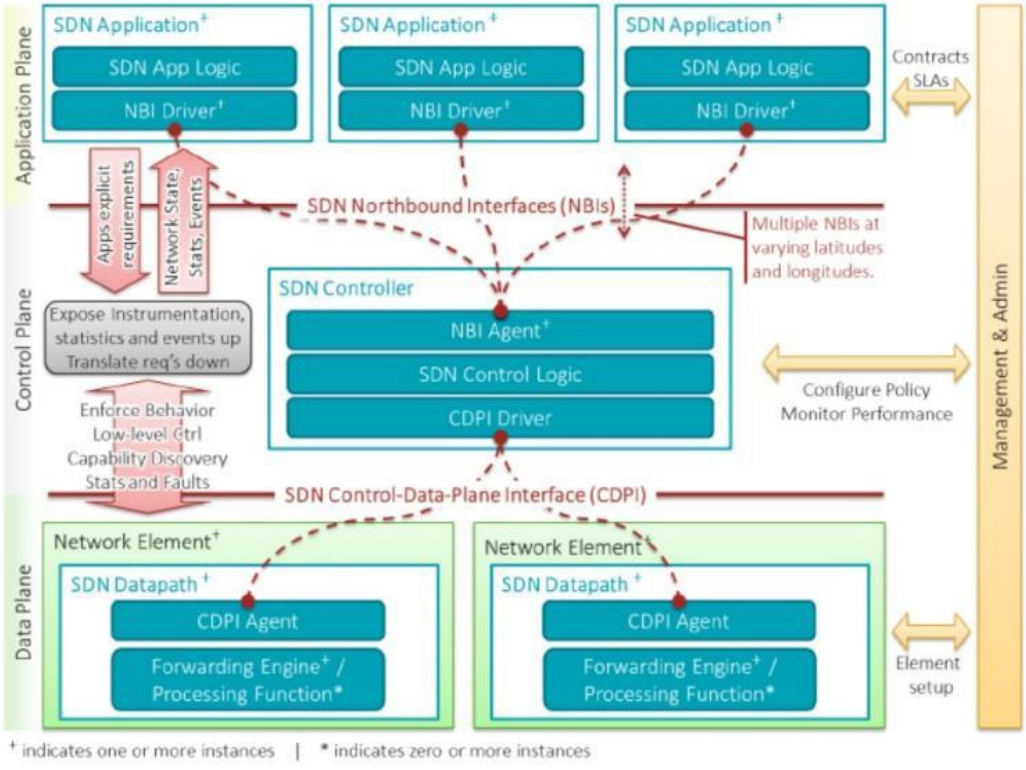

그림 2. Overview of SDN Architecture

Data-Plane Interface), NBI(Northbound Interface), Interface Drivers & Agents, Management & Admin으로 나뉜다.

CDPI는 SDN Controller와 SDN Datapath간에 정의 된 인터페이스이며 모든 포워딩 작업에 대해 프로그래밍 방식으로 제어하여 기능광고, 통계보고, 이벤트 알림을 제공한다.

NBI는 SDN Application과 SDN Controller 사이의 인터페이스이며 일반적으로 추상 네트워크를 가시화를 제공하고 네트워크 동작 및 요구사항을 직접 표현할 수 있다. 이러한 인터페이스는 개방적이고 중립적이므로 상호 운영 가능한 방식으로 구현된다.

Interface Drivers & Agents에서 각 Interface는 driver-agent 쌍으로 구현된다. Agent는 남쪽(southern), 아래쪽인 인프라를 나타내고 Driver는 북쪽(northern), 위쪽인 Application을 나타낸다.

Management & admin은 Application, Control 및 Data Plane 외부에서 처리되는 정적 작업을 다룬다. 예를 들어서 제공자와 클라이언트 간의 비즈니스 관계 관리, 클라이언트에 리소스 할당, 물리적 장비 설정, 논리적 및 물리적 엔티티 간의 연결 가능성 및 자격 증명 조정과 부트 스트래핑 구성 등이 이에 해당된다.

SDN은 데이터 플레인(data plane)과 제어 플레인(control plane)을 분리하여 표준화된 인터페이스를 제공하여 통신 제어 기능을 다양하게 제어할 수 있다.

3. 임베디드 디바이스에 SDN 적용 시 고려사항

SDN은 개방형 표준 소프트웨어를 기반으로 네트워크를 제어할 수 있으며 높은 유연성을 제공한다. SDN을 임베디드 디바이스에 적용 시에는 자원 제약의 특징을 가지고 있으므로 SDN의 구조적 특징과 임베디드 디바이스의 특징을 고려하여 적용 해야한다. 임베디스 디바이스 연구에서 많이 활용되고 있는 '라즈베리파이3 모델 B' 기반으로 SDN 환경 구축 시 고려사항은 아래와 같다.

3.1 와이파이 칩의 취약점

- 취약점:
  와이파이(Wi-Fi) 칩의 구조적인 취약점)
  Kr00k(CVE-2019-15126)

- 공격유형:
  Kr00k는 브로드컴(Broadcom)과 사이프레스(Cypress)가 제조하는 칩에서 발견 되었으며 해당 장치의 네트워크 통신을 올-제로(all-zero) 암호화 키로 암호화하며 공격이 성공하면 무선 네트워크 패킷을 해독.

- 고려사항:
  해당 모델(라즈베리파이3)에 적용할 경우 와이파이 문제점에 대한 패치 필요

3.2 오픈플로우 구현상의 취약점

- 취약점:
1. 오픈플로우는 인터페이스의 및 전반적으로 중요한 역할을 담당하고 있지만 악성 코드 등에 대한 대비가 충분히 갖춰져 있지 않음.
2. 오픈플로우 프로토콜이 임베디드 디바이스에서는 펌웨어로 구현 되며 하드웨어 제어를 담당하므로 취약점 존재

- 공격유형:
  DDoS(Distributed Denial of Service) 공격은 "분산 서비스 거부"로 동시에 여러 곳에서 발생하는 DoS 공격이다. 일반적으로 악성 소프트웨어에 감염되어 서버의 대역폭 소비 및 서버 자체를 공격하여 서버의 처리 능력 이상을 요구하여 웹 리소스를 사용할 수 없게 만든다. 자원 제약이 있는 임베디드 디바이스 환경에서는 이로 인한 피해가 더 커질 수 있다.

- 고려사항:
  자원 제약이 있는 임베디드 디바이스 기반으로 오픈플로우 스위치 및 오픈플로우 네트워크를 제어 시 각별한 주의가 필요하다. 만약 전체 네트워크를 총괄하는 컨트롤러 플레인에 서비스 거부 공격(DoS)이 발생한다면 전체 네트워크가 영향을 받아서 리소

스 사용 불가 혹은 속도 저하의 가능성이 존재한다. 또한 악성 코드 등에 감염으로 정보 유출 가능성도 존재한다.

펌웨어 데이터 추출의 경우 UART port, JTAG, Flash Memory Dump, Update packet sniffing에 대응.

3.3 SDN 컨트롤러의 취약점

- 취약점:

SDN Controller의 중앙 집중화된 구조 및 상대적으로 큰 역할로 공격 시 취약점 발생

- 공격유형:

고려사항-2와 유사

- 고려사항:

SDN 컨트롤러의 악성코드 감염에 대한 방지가 필요하며 기본적으로 IDS(Intrusion Detection System)와 IPS(Intrusion Prevention System)으로 대응하고 원본 프로그램의 일관성에 문제가 발생하면 복구할 수 있도록 백업 등으로 대응한다.

3.4 SDN의 구조적 취약점

- 취약점;

1. [그림 2]의 SDN 구조와 같이 Control plane과 Data plane으로 나누어 관리되므로 취약점 발생
2. TLS 사용에 따른 과부하 발생

- 공격유형:

TCP(Transmission Control Protocol) 사용으로 중간자 공격(Man-in-the-middle attack) 발생 가능. 중간자 공격은 통신하고 있는 Control plane와 Data plane 사이에 끼어들어 서로 교환하는 정보를 바꿈.

스푸핑(Spoofing) 공격으로 통신 정보를 속여 정상적인 서비스 방해

- 고려사항:

TLS(Transport Layer Security) 보안 프로토콜 적용으로 발생하는 속도 지연으로 TCP(Transmission Control Protocol) 사용으로 인한 중간자 공격에 대응. 라즈베리파이 기반의 SDN에서 DDoS 공격을 받게되면 자원 제약이 있는 환경에서 정상작동에 더 큰 영향이 발생되므로 대비가 필요하다.

3.5 블루투스 프로토콜의 취약점

- 취약점:

블루투스 무선 프로토콜의 취약점 발생.

BIAS(Bluetooth Impresonation AttackS)

- 공격유형:

BIAS 보안 결함[16]은 기기가 롱텀(long-term, 링크키) 키는 두 개의 블루투스 기기가 최초에 패어링 할 때 생성되며 블루투스 기기가 통신의 필요가 있을 때마다 패어링 과정 없이 연결할 수 있는 세션키를 인증한다. 이러한 인증 과정의 버그를 통하여 이전에 성립된 롱텀 패어링 키를 알지 못해도 다른 기기와 인증 및 연결이 가능하다[그림 3].

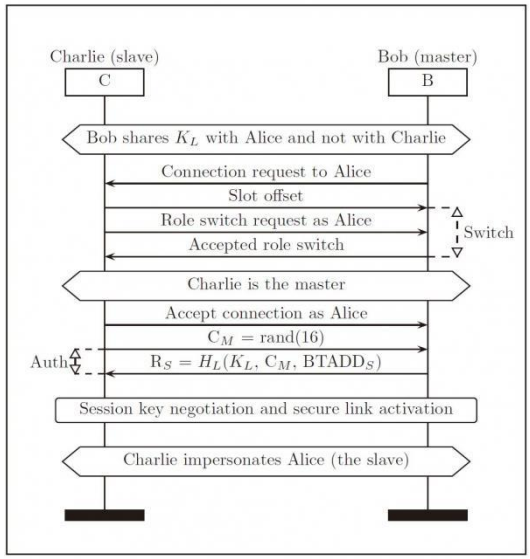

그림 3. BIAS 슬레이브 위장 공격

- 고려사항:
BIAS(CVE-2020-10135),
KNOB(CVE-2019-9506) 공격에 대한 패치를 실행한다.

이상으로 임베디드 디바이스에 SDN 적용에 대한 문제점과 고려사항을 제시하였다. 제시한 내용 외에도 발생 가능한 공격과 문제점은 다양하게 존재한다. 임베디드 디바이스 환경에서 SDN 구현 시 고려해야할 중요 요소는 오픈플로우의 컨트롤러의 중요도가 높으므로 이에 대한 대비가 필요하며 공격 이후 복구에 대한 대비도 필요하다는 것이다. '고려사항-4번'의 Control plane과 Data plane으로 분리 관리되는 구조에 대한 대비 또한 중요하다. 추가적으로 임베디드 디바이스(라즈베리파이)를 활용한 시스템으로 구축 시 가용시간 및 하드웨어의 물리적 안전성도 고려 대상이다. 또한 유무선 접속 방식에 있어서 무선(Wireless)기반은 누구나 접근 가능하도록 열려 있음에 따른 공격도 대비가 필요하다. 하드웨어 단계에서 소프트웨어 단계까지 다양한 공격 및 결함에 대비하여 다중 제어기를 사용하여 결함의 허용(Fault Tolerant)을 넓혀서 보다 높은 신뢰성을 제공해야 한다. 이를 위해서 라즈베리파이 기반의 클러스터 기반의 모델도 고려해볼 내용이다.

SDN 기반의 보안은 일반적으로 높은 수준의 보안이 요구되므로 다양한 인증 체계를 표준화하고 네트워크 사이의 액세스 제어 및 통합관리에 따른 하드웨어적 보안 대책도 요구된다.

임베디드 디바이스 기반의 모바일 시장의 증가와 다양한 통신 기능의 활용이 높아지고 있으며 클라우드 기반의 환경으로의 전환은 급변하는 네트워크 사용 환경의 변화에 대응은 쉽지 않지만 기존 네트워크 구조에 대한 변화는 반드시 필요하다. 트래픽 증가로 네트워크의 규모의 예측이 어려워지는 현실에서 빠르게 변화하는 사용자의 요구에 맞추어 새로운 서비스를 제공을 위한 개발에도 대응을 해야겠다. SDN은 기존 네트워크 구성의 문제점을 해결하며 복잡해진 경로 구성과 트래픽 증가에 효과적으로 대처가 가능하며 빅데이터(Big Data) 사물인터넷(IoT) 환경에 대응하여 분석에 필요한 대용량 네트워크를 작은 비용으로 구축 가능하며 보안 관점에서 다양한 보안 위협이 있는 환경에서 중앙 집중 관리 기반으로 빠르게 제어가 가능한 환경을 제공한다. 그러나 SDN의 소프트웨어적 기능만으로 AI, 빅데이터, 5G 등 기술이 융합되고 보안 위협이 지능화되고 고도화되는 시점에서 보안 위협에 대응하기에는 일부 제한과 한계를 가지고 있으므로 다양한 보완이 요구된다.

IV. 결론

제 4차 산업혁명 시대에서 ICT 분야는 빠르게 변화하고 있으며, 다양한 임베디드 디바이스의 필요성이 증가하고 있다. 이러한 환경에서 필수적으로 발생할 수 있는 보안 이슈와 개인 정보의 유출에 대한 관심은 높다. 기존의 범용 컴퓨터 기반의 보안 기법은 빠르게 변화하는 네트워크 환경에 대응하기에 부족하며 시스템 자원의 제약이 따르는 임베디드 디바이스에 적용에도 한계가 따른다. SDN 기반의 스위치 가격을 고려 했을 때 임베디드 디바이스 환경에서 네트워크의 하부구조에 영향을 받지 않고 소프트웨어적으로 네트워크를 제어하는 SDN은 보안 대응에 효율적이다. 본 연구에서는 활용도가 높아지고 있는 임베디드 디바이스에 SDN을 적용 시 발생할 수 있는 문제점을 SDN의 구조적 특징과 임베디드 디바이스의 특징을 기반으로 분석하여 새로운 환경에 적용 시 미리 대비할 수 있도록 고려사항을 제시하였다. 주요 고려사항을 아래와 같다.

1. 와이파이(Wi-Fi) 칩의 구조적 취약점에 대응
2. 오픈플로우의 공격 및 임베디드 디바이스의 펌웨어 기반 동작에 따른 대응
3. SDN Controller의 구조적 취약점으로 IDS와 IPS로 대응
4. SDN이 구조적으로 데이터 플레인과 제어 플레인으로 나뉘어 있으며 이로인한 중간자 공격 및 스푸핑 공격에 대응하고 TCP 사용에 대한 대응
5. 시스템온칩(SoC) 보드를 사용하는 임베디디디바이스의 블루투스 무선 프로토콜의 취약점에 대응

SDN 적용으로 먼저 중앙 집중화와 효율적 제어가 가능하며 일관된 정책으로 보안성이 증대된다. 임베디드 디바이스 기반의 보안은 디바이스 단계부터 서비스 단계까지 다양한 공격이 존재한다. 소프트웨어적인 부분에서 부팅단계의 보안과 임베디드 운영체제의 보완, 인증과 검증 및 서비스 거부(Dos) 공격 등에 대한 대비도 필요하다.

현재 연구 결과는 예방 단계에 집중되어 있으므로 향후 연구 계획으로 임베디드 디바이스에 SDN 적용 이후에 발생 가능한 다양한 문제점과 성능 평가에 대한 내용을 체계적으로 분석하고 대안을 제시하는것이 필요하다. 또한 OSCAR(Object Security Architecture for the Internet of Things) 등을 적용하여 해당 디바이스에 최적화된 보안 기법을 제안하고자 한다.

ICT 관련 임베디드 디바이스의 네트워크 접속이 기하급수적으로 높아지고 있는 현실에서 개인 정보 유출 방지 및 보안의 필요성도 높아지고 있다. 보안에 대한 높은 관심과 관련 연구를 통해서 취약점이 줄어들고 기술이 개선되어 더욱 발전해 나가길 기대한다.

참 고 문 헌

[1] 장정숙, 전용희, "임베디드 시스템 보안," 한국통신학회논문지, Vol.22, No.9, pp.81-97, Aug. 2005.

[2] P. Koopman, "Embedded System Security," EMBEDDED COMPUTING, pp.95-97, Jul. 2004.

[3] H. Cho and J. Lee, "An Analysis of the Vulnerability of SSL/TLS for Secure Web Services," Journal of the Korea computer industry education society, Vol.2, No.10, pp.1269-1284, 2001.

[4] IP Security Protocol, https://datatracker.ietf.org/wg/ipsec/about/, 2021.1.26.

[5] M. Vucini, B. Tourancheau. F. Rousseau, A. Duda, L. Damon, and R. Guizzetti, "OSCAR: Object security architecture for the Internet of Things," Ad Hoc Networks, Vol.32, pp.3-16, Sept 2015. B. V., Amsterdam, AD HOC NETWORKS, Vol.32, 2015.

[6] P. Kocher, R. Lee, G. McGraw, A. Raghunathan, and S. Ravi, "Security as a New Dimension in Embedded System Design," ACM, pp.753-760, Jun. 2004.

[7] Open Networking Foundation, https://opennetworking.org/, 2021.1.26.

[8] Open Networking Foundation, "SDN Architecture," Jun. 2014.

[9] K. Kirkpatrick, "Software defined networking," Communications of the ACM, Vol.56, No.9, Sept. 2013.

[10] W. Xia and Y. Wen, "A Survey on Software-Defined Networking," IEEE COMMUNICATION SURVEYS & TUTORIALS, Vol.17, No.1, 2015.

[11] Y. Aggarwal and U. Kumari, "Software Defined Networking: Basic Architecture & Its Uses In Enterprises," International Conference on "Computing: Communication, Network and Security"(IC3NS-2018), pp.74-80, May, 2018.

[12] J. H. Jung, "IETF I2NSF Standardization Trend for SDN/NFV-based Security Service," Telecommunications Technoogy Associations, Special Report, Vol.185, pp.12-18, Nov. 2019.

[13] J. H. Yoo, U. S. Kim, and C. H. Youn, "A Technical Trend and Prospect of Software Defined Network and OpenFlow," KNOM Review, Vol.15, No.2, pp.1-24, Dec. 2012.

[14] RaspberryPi, https://www.raspberrypi.org/, 2021.1.26.

[15] Mininet, http://mininet.org/, 2021.1.26

[16] Daniele Antonioli; Nils Ole Tippenhauer; Kasper Rasmussen, "BIAS: Bluetooth Impersonation AttackS," 2020, IEEESymposium on Secrity and Privacy(SP), May. 2020.

[17] S. Midha and K. Tripathi, "Extended Security in Heterogeneous Distributed SDN Architecture," Advances in Communication and Computational Technology, pp.991-1002, Aug. 2020.

[18] E. Torres, R. Reale, L. Sampaio, and J. Martins, "A SDN/OpenFlow Framework for Dynamic Resource Allocation based on

Bandwidth Allocation Model," EEE America Latina. Revista, Vol.18, No.5, pp.853-860, 2020.

[19] Alsaeedi, Mohammed, "Toward Adaptive and Scalable OpenFlow-SDN Flow Control: A Survey," IEEE ACCESS, Vol.7, pp.107345-107379, 2019.

저 자 소 개

구 금 서(GeumSeo Koo) 정회원

- 2003년 2월 : 경상대학교 컴퓨터과학과(이학사)
- 2005년 8월 : 경상대학교 컴퓨터과학과(공학석사)
- 2007년 8월 : 경상대학교 컴퓨터과학과 박사수료
- 2007년 3월 ~ 현재 : 경상국립대학교 교양학부 강사

〈관심분야〉 : 임베디드시스템, 시스템보안, 인공지능 윤리

심 갑 식(Gabsig Sim) 종신회원

- 1993년 8월 : 전남대학교 전산통계학과(이학박사)
- 1993년 10월 ~ 2021년 2월 : 국립경남과학기술대학교 교양학부 교수
- 2004년 3월 ~ 2005년 2월 : San Jose State University, CA USA 방문교수
- 2021년 3월 ~ 현재 : 경상국립대학교 휴먼헬스케어학과 교수

〈관심분야〉 : 인공지능, 스마트 헬스케어, 정보통신기술, 인공지능 윤리