IoT 보안 지침

엔드포인트 생태계

IoT 보안 지침 엔드포인트 생태계

버전 2.0

2017 년 10 월 31 일

본 문서는 GSMA 의 구속력이 없는 영구 참조 문서입니다

보안 분류: 비기밀

본 문서의 열람과 배포는 보안 등급에서 허가된 자에게 한정됩니다. 본 문서는 협회의 기밀이며 자작권 보호 대상입니다. 본 문서는 공급된 목적에 한하여 사용해야 하며, GSMA 의 사전 서면 승인 없이 보안 등급에 따라 허가받은 자 이외의 사람에게 이 문서 수록된 정보의 전부 또는 일부를 공개하거나 제공해서는 안 됩니다.

저작권 고지

Copyright © 2018 년 3 월 29 일 Thursday AM 10:54:04 GSM Association

면책 조항

GSM 협회("협회")는 본 문서에 수록된 정보의 정확성이나 완전성, 적시성에 대해 어떠한 진술이나 보증, 약속(명시적으로나 묵시적으로나)도 하지 않고 책임도 지지 아니하며 배상할 의무도 없습니다. 본 문서에 수록된 정보는 예고 없이 변경될 수 있습니다.

독점금지 고지

본 문서에 수록된 정보는 GSM 협회의 독점금지 준수 정책에 부합합니다.

목차

1 서론

1.1 GSMA IoT 보안 지침서 세트 개요

본 문서는 태동기의 "사물 인터넷"(IoT) 업계가 IoT 보안 문제를 함께 이해할 수도 있도록 마련한 GSMA 보안 지침서의 한 부분입니다. 본 지침서는 구속력이 없으며 안전한 IoT 서비스를 개발하는 방법을 보급해 서비스 분야 전체에서 보안 모범 사례가 정착되게 하는 데 목적이 있습니다. 지침서에서는 IoT 서비스에 만연한 보안 위협과 취약점에 대처하는 방안을 제시합니다.

GSMA 보안 지침서의 구성은 다음과 같습니다. 부속 문서 CLP.12 [2]와 CLP.13 [3](본 문서)를 읽기 전에 'CLP.11 IoT 보안 지침 개요서'을 일종의 예비서로 읽기를 권장합니다.

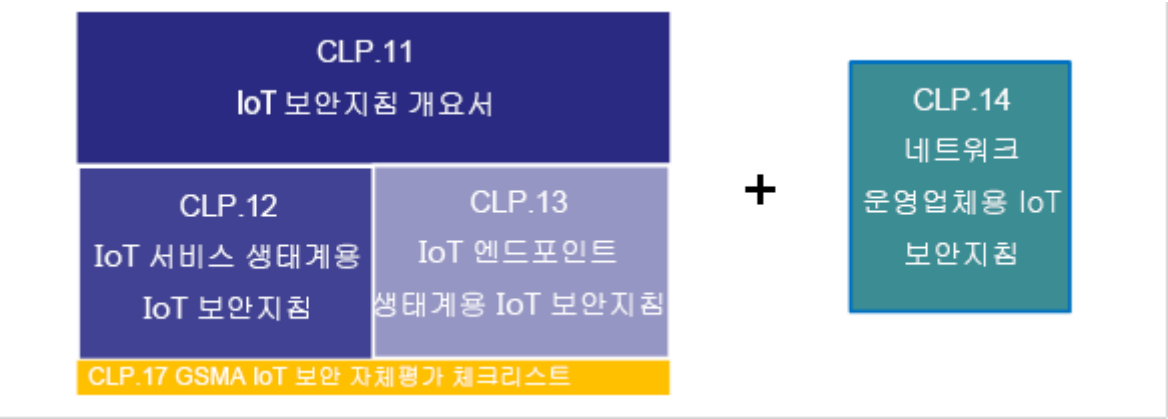

그림 1 - GSMA IoT 보안 지침서의 구조

네트워크 운영사, IoT 서비스 업체와 IoT 생태계의 기타 파트너들은 시스템 보안과 데이터 보호를 위해 IoT 서비스 업체에게 서비스를 제공하고자 하는 네트워크 운영사를 대상으로 최상위 보안 지침을 제시하는 GSMA 문서 CLP.14 "네트워크 운영자용 IoT 보안 지침서(IoT Security Guidelines for Network Operators)"[4]를 읽어 보시기 바랍니다.

1.1.1 GSMA IoT 보안 평가 체크리스트

문서 CLP.17 [19]에 평가 체크리스트가 제시돼 있습니다. IoT 제품과 서비스, 구성품 공급업체는 본 문서를 통해 자사의 제품과 서비스, 구성품이 GSMA IoT 보안 지침에 부합하는지 스스로 평가할 수 있습니다.

GSMA IoT 보안 평가 체크리스트[19]를 작성하면 회사가 사이버 위험으로부터 제품과 서비스, 구성품을 보호하기 위해 강구한 보안 조치를 검증할 수 있습니다.

작성된 신고서를 GSMA 에 제출하면 평가 확인을 받을 수 있습니다. GSMA 웹사이트에서 아래 절차를 참고하십시오.

https://www.gsma.com/iot/future-iot-networks/iot-security-guidelines/

1.2 문서의 목적

본 문서는 IoT 엔드포인트 장치의 관점에서 IoT 서비스의 구성요소를 평가하는 데 이용해야 합니다. IoT 의 관점에서 엔드포인트란 인터넷과 연결된 제품이나 서비스의 일부로서 기능이나 임무를 수행하는 실물 컴퓨팅 장치를 말합니다. 예를 들면, 웨어러블 피트니스 장치, 산업용 제어 시스템, 자동차 텔레매틱스 장치 등이며 개인용 드론도 엔드포인트로 볼 수 있습니다. 이 같은 실물 장치의 구동에 사용되는 기술에 대해서는 모두 보안 위험을 평가해야 합니다. 그리하여 리더(reader)가 IoT 서비스에 위협이 될 만한 요소를 빠짐 없이 찾아내 시정할 수 있도록 실무 설계 지침을 만들어야 합니다.

본 문서의 범위는 IoT 엔드포인트 장치의 설계와 구현에 관한 제언으로 한정됩니다.

본 문서에서는 IoT 사양이나 표준의 개발을 제안하지 아니하며 현재 시중에 나와 있는 솔루션과 표준, 모범 사례만을 언급합니다.

본 문서는 또 기존 IoT 서비스의 퇴출을 촉구하려는 목적도 없습니다. 보안 확보를 고려할 때에는 네트워크 사업자의 기존 IoT 서비스와 하위 호환성을 유지해야 합니다.

지역에 따라 필요한 경우 국가의 법규가 본 문서에 명시된 지침에 우선할 수도 있습니다.

1.3 대상 독자

본 문서의 주된 독자는 다음과 같습니다.

- IoT 서비스 업체 - 새롭고 혁신적인 커넥티드 제품과 서비스를 개발하고자 하는 기업이나 조직. IoT 서비스 업체가 많이 활약하고 있는 업종으로는 스마트홈, 스마트 시티, 자동차, 운수, 건강, 전기, 가전 등이 있습니다.
- IoT 디바이스 제조업체 - IoT 서비스가 가능하도록 IoT 서비스 업체에게 IoT 디바이스를 공급하는 업체.
- IoT 개발업체 - IoT 서비스 업체를 대신해 IoT 서비스를 개발하는 업체.
- IoT 서비스 업체에 서비스를 제공하는 네트워크 운영사.

1.4 정의

용어	설명
액세스 포인트 이름	엔드포인트 디바이스가 연결되는 네트워크 연결 지점의 식별자. 여러 가지 서비스 타입과 연결돼 있으며 네트워크 운영업체별로 구성되는 경우가 많습니다.
공격자	IoT 서비스를 상대로 한 해커나 위협원, 위협 액터, 사기꾼, 그 외 악성 위협. 이 같은 위협은 단독 범인이나 조직 범죄, 테러, 적대적 정부 및 그 기관, 산업 스파이, 해킹 그룹, 정치 활동가, '하비스트' 해커, 연구자, 의도하지 않은 보안 또는 프라이버시 침해에서 올 수 있습니다.
셀룰러	3GPP 표준화 모바일 네트워크 기술(GSM, UMTS, LTE (inc LTE-M), NB-IoT 등)의 총칭.
클라우드	애플리케이션과 그 데이터를 호스팅하고 저장, 관리, 처리하는 인터넷상의 원격 서버 네트워크.
컴플렉스 엔드포인트	휴대전화나 위성 같은 장거리 통신, 이더넷과 같은 유선 연결을 통해 백엔드 서버로 상시 연결되는 엔드 포인트 모델. 3 절 참조.
임베디드 SIM	쉽게 분리하거나 교체할 수 없는 SIM 으로서 디바이스에서 분리와 교체는 불가하며 프로필을 안전하게 바꿀 수 있는 SIM 을 말합니다.
엔드포인트	엔드포인트란 인터넷과 연결된 제품이나 서비스의 일부로서 기능이나 임무를 수행하는 실물 컴퓨팅 장치를 말합니다. 3 절에 IoT 장치의 3 대 분류와 각 엔드포인트 분류의 예가 제시돼 있습니다.
사물 인터넷	복수의 기계와 장치, 어플라이언스가 조율된 형태로 복수의 네트워크를 통해 인터넷에 연결된 상태를 일컫는 말. 여기서 장치란 태블릿, 가전제품과 같은 일상적 기물 외에도 기계간(M2M) 통신 기능이 있어 데이터를 주고 받을 수 있는 자동차, 모니터, 센서 등을 통칭합니다.
IoT 서비스	IoT 디바이스에서 나온 데이터를 이용해 서비스를 하는 컴퓨터 프로그램을 통칭합니다.
IoT 서비스 생태계	기능을 제공하고 실무에 배치된 엔드포인트에서 데이터를 수집하는 데 필요한 일단의 서비스와 플랫폼, 프로토콜, 기타 기술. 자세한 내용은 CLP.11 [1]를 참고하십시오.
IoT 서비스 업체	새롭고 혁신적인 커넥티드 제품과 서비스를 개발하고자 하는 기업이나 조직.

용어	설명
네트워크 운영업체	IoT 엔드포인트 디바이스를 IoT 서비스 생태계와 연결하는 통신 네트워크의 운영자 또는 소유자.
조직의 신뢰 기반(RoT)	ID 와 애플리케이션, 통신의 암호화 보안 방법을 관장하는 암호화된 정책과 절차.
서비스 액세스 포인트	통신 네트워크를 통해 IoT 서비스의 백엔드 인프라에 들어가는 지점.
구독자 ID 모듈	모바일 네트워크에서 디바이스의 모바일 네트워크 연결과 네트워크 서비스 접근을 허용하기 위해 사용되는 스마트 카드.
트러스트 앵커	트러스트 앵커란 계층 구조를 지닌 크립토그래픽 시스템에서 트러스트(신뢰)가 있다고 가정해 추론하지 않는, 권위 있는 엔터티를 말합니다.
신뢰 컴퓨팅 기반	신뢰 컴퓨팅 기반(TCB)이란 제품이나 서비스에 들어 있는 알고리즘과 정책, 비밀의 집합체를 말합니다. TCB 는 하나의 모듈로서 제품이나 서비스는 이를 통해 자기의 신뢰도를 측정하고 네트워크 피어의 인증 수준을 측정하며 주고받은 메시지의 무결성을 검증할 수 있습니다. TCB 는 보안 제품과 서비스 구축의 근간이 되는 기본 보안 플랫폼 역할을 합니다. TCB 의 구성요소는 환경(엔드포인트의 하드웨어 TCB, 클라우드 서비스용 소프트웨어 TCB)에 따라 달라지지만 추상적 목표와 서비스, 절차, 정책은 매우 유사합니다.
신뢰 실행 환경(TEE)	리치 운영체제와 함께 실행돼 운영체제에게 보안 서비스를 제공하는 환경을 말합니다. TEE 실행에 이용할 수 있는 기술에는 여러 가지가 있으며, 구현되는 보안 수준도 기술에 따라 달라집니다.
UICC	ETSI TS 102 221 에 명시된 보안 요소 플랫폼으로서 암호화를 통하여 분리된 보안 도메인에서 복수의 표준화 네트워크 또는 서비스 인증 애플리케이션을 지원할 수 있는 것을 말합니다. ETSI TS 102 671 에 명시된 임베디드 폼 팩터 안에 구현될 수도 있습니다.

1.5 약어

용어	설명
3GPP	3 세대 프로젝트 파트너십
AC	교류
API	응용 프로그램 인터페이스

용어	설명
APN	액세스 포인트 이름
BLE	블루투스 저에너지
BT	블루투스
CLP	GSMA 커넥티드 리빙 프로그램
CPE	고객 구내 장비
CPU	중앙처리장치
EEPROM	전기적으로 지울 수 있고 프로그램 가능한 읽기 전용 메모리
eUICC	임베디드 UICC
FIB	집속 이온빔
GBA	범용 부트스트래핑 아키텍처
GPS	위성 위치 확인 시스템
GSMA	GSM 협회
IoT	사물 인터넷
IP	인터넷 프로토콜
ISM	산업용, 과학용, 의료용
LAN	근거리 네트워크
LPWA	저전력 장거리 통신망
LTE-M	기계 롱텀 에볼루션
MCU	마이크로컨트롤러 장치
NB-IoT	협대역 사물 인터넷
NVRAM	비휘발성 랜덤 액세스 메모리
OMA	오픈 모바일 연대
PAN	개인 네트워크
PSK	사전공유키
RAM	랜덤 액세스 메모리
ROM	읽기 전용 메모리
SCADA	감시 제어 및 데이터 취득

용어	설명
SPI	직렬 주변장치 인터페이스
SSH	보안 쉘
SIM	구독자 ID 모듈
SRAM	정적 랜덤 액세스 메모리
TCB	신뢰 컴퓨팅 기반
TTL	트랜지스터 간 로직
UART	범용 비동기 수신기/송신기

1.6 참고 문서

참고	문서 번호	제목
[1]	CLP.11	IoT 보안 지침 개요서(IoT Security Guidelines Overview Document)
[2]	CLP.12	IoT 서비스 생태계를 위한 IoT 보안 지침(IoT Security Guidelines for IoT Service Ecosystem)
[3]	CLP.13	IoT 엔드포인트 생태계를 위한 IoT 보안 지침(IoT Security Guidelines for IoT Endpoint Ecosystem)
[4]	CLP.14	네트워크 운영자를 위한 IoT 보안 지침(IoT Security Guidelines for Network Operators)
[5]	OMA FUMO	OMA 펌웨어 업데이트 관리 객체(OMA Firmware Update Management Object) www.openmobilealliance.org
[6]	na	ST-LINK/V2 in-circuit debugger/programmer http://www.st.com/
[7]	na	모바일 IoT 이니셔티브(Mobile IoT Initiative) https://www.gsma.com/iot/mobile-iot-initiative/
[8]	na	Nmap 보안 스캐너(Nmap Security Scanner) https://nmap.org/
[9]	CLP.03	IoT 장치 여결 효율 가이드라인(IoT Device Connection Efficiency Guidelines) https://www.gsma.com/iot/gsma-iot-device-connection-efficiency-guidelines/
[10]	na	연방 정보 처리 기준(Federal Information Processing Standards) www.nist.gov/itl/fips.cfmwww.nist.gov/itl/fips.cfm

참고	문서 번호	제목
[11]	na	EMVCo www.emvco.com/
[12]	na	SIM 얼라이언스 - 오픈 모바일 API(SIM Alliance - Open Mobile API) simalliance.org/key-technical-releases/
[13]	GPD_SPE_013	글로벌플랫폼 보안 요소 액세스 제어(GlobalPlatform Secure Element Access Control) www.globalplatform.org/specificationsdevice.asp
[14]	GPD_SPE_024	글로벌플랫폼 신뢰 실행 환경 API 규격(GlobalPlatform Trusted Execution Environment API Specification) www.globalplatform.org/specificationsdevice.asp
[15]	GPC_SPE_034	글로벌플랫폼 카드 규격(GlobalPlatform Card Specification) www.globalplatform.org/specificationscard.asp
[16]	ISO/IEC 29192-1	정보기술 - 보안 기법 - 경량 크립토그래피(Information technology -- Security techniques -- Lightweight cryptography) www.iso.org/obp/ui/#iso:std:iso-iec:29192:-1:ed-1:v1:en
[17]	TS 33.220	일반 인증 아키텍처(Generic Authentication Architecture, GAA); 범용 부트스트래핑 아키텍처(Generic Bootstrapping Architecture, GBA) www.3gpp.org
[18]	TS 33.222	범용 인증 아키텍처(GAA), HTTPS 를 이용한 네트워크 애플리케이션 기능의 접속 www.3gpp.org
[19]	CLP.17	GSMA IoT 보안 평가 체크리스트 https://www.gsma.com/iot/iot-security-assessment/
[20]	TS-0003	oneM2M 보안 솔루션(oneM2M Security Solutions) www.onem2m.org
[21]	3GPP TS33.163	저산출 머신형 통신(MTC) 장치를 위한 배터리 효율 보안(BEST)(Battery efficient Security for very low Throughput Machine Type Communication (MTC) devices (BEST)) www.3GPP.org

2 IoT 엔드포인트 보안의 해결과제

IoT 서비스의 보안 문제는 많은 경우 이 서비스에 이용되는 IoT 엔드포인트의 독특한 특정과 직접 관련이 있습니다. 예컨대, IoT 엔드포인트 중에는 다음과 같은 특성을 지닌 것이 많은데, 이것이 여러 가지 보안 문제를 야기합니다.

2.1 낮은 전력 소모

- 원격으로 접속하지 못하는 엔드포인트의 경우 항구적인 전원 공급장치 없이 긴 수명(수년)을 확보하기 위해 전력 소모가 낮은 경우가 있습니다. 항구적인 전원이 있어도 태양 에너지처럼 제약이 있는 것이라면 마찬가지입니다.
- 전력 소모가 낮은 엔드포인트는 대개 연산적으로 단순한 크립토그래픽 운전(예컨대 ISO/IEC 29192 [16] 안에서 정의된 경량 크립토그래픽 운전)만 할 수 있습니다. 더 고급의 크립토그래픽 운전은 전력 소모의 증가를 동반하기 때문입니다. 이 경우 한정된 대역의 통신만 가능할 수도 있습니다. 그러면 크립토그래픽 능력도 제한되기 마련입니다.

2.2 낮은 원가

- 많은 IoT 서비스의 사업 모델이 IoT 엔드포인트의 저비용을 전제로 하고 있습니다. 이는 장치의 낮은 처리 능력과 적은 메모리, 제한된 운영체제를 의미할 때가 많습니다. 그 결과 장치가 '인터넷급' 크립토그래피를 수행하지 못할 수도 있습니다.

2.3 긴 수명(10 년 이상)

- 엔드포인트 중에는 수명이 길어야 하는 것이 많습니다. 특히 공공 용도나 산업용도(예: 스마트 가스 계량기)에 그런 게 많습니다. 여기에는 애로가 따릅니다. 장치를 설계할 때 정해진 크립토그래픽 설계가 장치의 수명이 다할 때까지 견뎌야하기 때문입니다. 가령, 그 10 년 사이에 공격자의 달러당 처리 능력이 16 배 증가할 때 장치의 능력은 그대로일 가능성이 높습니다.
- 장수명 장치의 관리도 문제입니다. 특히 보안 취약점이 발견되는데 해당 IoT 엔드포인트에서 해결할 수 없다면 더욱 그렇습니다.

2.4 물리적 접근

- IoT 엔드포인트 중에는 공격자가 실제로 접근할 수 있는 것이 많습니다. 그러면 그런 엔드포인트의 하드웨어 부품과 인터페이스는 공격의 목표가 될 가능성이 있고 개발자는 보안 대책을 마련해야 합니다.

IoT 서비스 중에는 IoT 엔드포인트가 장거리 통신망에 직접 연결되지 않는 경우가 많고 인터넷 프로토콜(IP) 기능이 없는 엔드포인트도 많습니다. 예를 들면, IoT 엔드포인트가 산업용, 과학용, 의료용(ISM) 무선 송수신기를 이용해 로컬 IoT 서비스 게이트웨이로 데이터를 전송하면 그 게이트웨이가 IP 를 이용해 통신 네트워크로 데이터를 전달하는 것입니다. 이 경우 종단간 통신의 보안을 확보하는 과정은 더욱 복잡해집니다.

뒤에서 설명하겠지만, IoT 엔드포인트의 기능과 관련 보안 위험에 따라, 복잡도가 다른 여러 가지 보안 방법을 적용해야 할 수도 있습니다.

3 IoT 엔드포인트 모델

IoT 엔드포인트 모델은 한 때 별개의 기술, 즉 실제 세상을 상대하며 인터넷 어딘가에 있는 서버와 연결해 지침을 얻거나 메트릭스를 제출하는 수단 정도로 치부되었으나 지금은 완전히 다른 존재가 되었습니다. 현대 엔지니어링에서 IoT 기술은 몇 가지 변종만이 존재하는 예측 가능한 모델로 변모하였습니다. IoT 엔드포인트도 예측 가능성이 높아지고 있으며 앞으로 다음 속성 중 하나만 갖게 될 것으로 예상됩니다.

- 경량 엔드포인트
- 복잡한 엔드포인트
- 게이트웨이(또는 "허브")

아래 다이어그램에 다음과 같은 보편적 IoT 엔드포인트가 몇 가지 제시돼 있습니다.

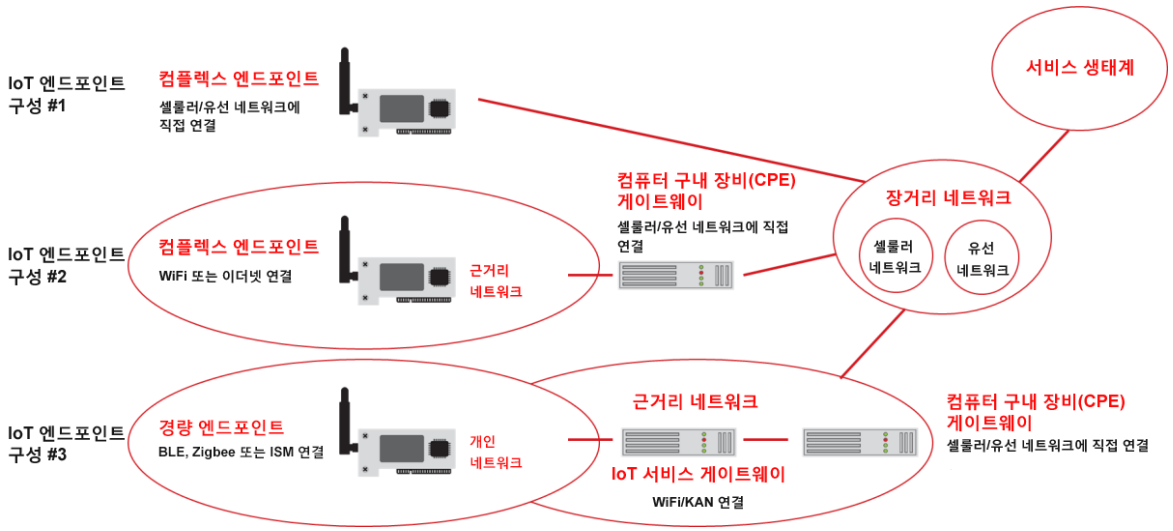

그림 2 - IoT 엔드포인트 구성의 예

3.1 경량 엔드포인트

이 엔드포인트는 조명 스위치나 문 자물쇠처럼 기능이 크게 제한된 센서나 간단한 실물 장치를 말합니다. 이런 엔드포인트의 목표는 한 가지 물리적 목적을 수행하고 서비스 생태계나 소비자에게 메트릭스를 제공하는 것입니다. 주로 8 비트 마이크로컨트롤러 등 매우 저렴한 처리장치와 BLE(Bluetooth Low Energy), Thread, Zigbee 등 단거리 개인통신망(PAN)을 이용합니다. 보통 저전력이며 동전 배터리나 태양력, 작은 리튬 폴리머 전지로 구동합니다. 대개 그림 2 '엔드포인트 구성의 예 3'와 같이 IoT 서비스 게이트웨이를 통해 서비스 생태계와 고객 구내 장비 게이트웨이에 연결됩니다.

경량 엔드포인트의 예는 다음과 같습니다.

- 웨어러블
- 가정용 보안 센서 엔드포인트
- 근접 신호등
- 비셀룰러 단말 장치

경량 엔드포인트는 원가가 낮아 적용할 만한 보안 기술이 별로 없습니다. 기판에 큰 전류 인입이나 비용, 공간을 요하는 보안 기술은 이런 시스템에 거의 쓰지 못합니다. 그렇지만 비용 대비 효과가 좋은 작은 트러스트 앵커로 견고한 보안 체계를 구현할 수 있습니다.

3.2 복잡한 엔드포인트

이 엔드포인트 모델은 대개 셀룰러(LPWA 네트워크 포함) 등의 장거리 통신 링크(그림 2 의 '엔드포인트 구성의 예 1' 참조)를 통해 백엔드 서비스와 항구적으로 연결되거나, Wi-Fi 또는 이더넷을 이용해 고객 구내 장비 게이트웨이(그림 2 의 '엔드포인트 구성의 예' 2)를 통해 연결됩니다. 장치 안에 들어 있는 프로세서는 8 비트 마이크로컨트롤러 정도로 작을 수도 있지만 교류와 직접 연결되거나 안에 배터리를 내장하고 배터리 충전 시스템과 주기적으로 연결되므로 처리 능력이 큽니다. 일부 복잡한 엔드포인트는 모세관 프로토콜을 통해 통신하기도 하지만 스트리밍 오디오 장치와 같은 로컬 애플리케이션을 효율적으로 실행하려면 전력이 더 필요합니다.

복잡한 엔드포인트의 예는 다음과 같습니다.

- IoT 와 연결된 조명 시스템

- 냉장고, 세탁기 등 가전제품
- 산업용 제어시스템(예: SCADA)
- 리트로핏 OBD2 셀룰러 "커넥티드 카" 추적 및 모니터링 장치

복잡한 엔드포인트는 전류 인입을 더 많이 할 수 있으며 대개 더 강력한 프로세서를 구현합니다. 또 기판에 보안 기술을 적용할 만한 공간이 많습니다. 따라서 복잡한 엔드포인트로는 더 많은 작업을 할 수 있습니다. 이들 장치에는 거의 모든 종류의 트러스트 앵커를 사용할 수 있습니다. 그러므로 뒤에서 설명할 PSK(Personalized Pre-Shared Key)나 비대칭 신뢰 컴퓨팅 기반(TCB) 모델고 쉽게 구현할 수 있습니다.

3.3 게이트웨이(또는 "허브")

게이트웨이는 대개 전용 전원 장치에 연결돼 경량 엔드포인트와 그것을 구동하는 백엔드 시스템 간의 통신을 관리하는 장치를 말합니다. 게이트웨이는 셀룰러(LPWA)나 위성, 유선, 섬유, 이더넷 같은 장거리 통신선을 관리합니다. 서비스 생태계에 안에 있는 백엔드 시스템에서 명령을 받아 경량 엔드포인트에서 소비할 수 있는 메지지로 변환합니다. 엔드포인트

IoT 게이트웨이의 주된 기능은 경량 엔드포인트로 메시지를 전달하고 받는 것이지만 다음과 같은 중요한 기능도 할 수 있습니다.

- 장치 발견
- 네트워크 드라이버 배포
- 관리 기능
- 런타임 모니터링
- GBA 나 TLS 셋업 등 인증과 보안

게이트웨이도 기술적으로는 엔드포인트지만 최종사용자가 관리하지 않고 IoT 서비스 업체나 네트워크 운영사(아래 참조)의 관리를 받을 수도 있습니다. 어느 쪽이든 게이트웨이를 복잡한 엔드포인트로 설계해 로컬화된 네트워크 내의 여러 경량 엔드포인트로 업링크를 배분하는 작업을 더 효율적으로 할 수도 있습니다.

게이트웨이는 복잡한 엔드포인트와 같이 처리 능력과 전력 인입을 높일 수 있으며 기판 안에 공간도 많습니다. 따라서 IoT 게이트웨어는 복잡한 신뢰 컴퓨팅 기반 솔루션과 GBA 인증 클라이언트 등의 기술을 비교적 쉽게 구현할 수 있습니다.

또한 여러 가지 통신 기술을 내장해 이종 네트워크 장치 간 메시지를 전달할 수도 있습니다. 그러면 통상 메시지 교환을 효과적으로 하지 못하는 엔드포인트 간에도 통신이 가능합니다. 게이트웨이는 이 같은 방식으로 로컬 생태계 안에서 장치의 집합점 역할을 해, 장치 간, 그리고 필요 시 네트워크와 서비스 생태계 간 통신을 지원합니다.

게이트웨이는 일반적으로 "IoT 서비스 게이트웨이"와 "고객 구내 장비(CPE) 게이트웨이", 이렇게 두 가지로 나뉩니다. 차이점은 다음과 같습니다.

1. IoT 서비스 업체에서 "IoT 서비스 게이트웨이"를 제공합니다. 최종 사용자가 게이트웨이를 소유하기도 하지만 관리는 IoT 서비스 업체가 맡습니다. 이 게이트웨이는 대개 경량 엔드포인트를 (유선/셀룰러 연결을 통해 직접 또는 CPE 게이트웨이를 통해) 서비스 생태계와 연결하는 허브 역할을 합니다. 이때 최종 사용자는 IoT 서비스 업체에게 관리 서비스를 구입합니다.
2. 네트워크 운영사에서 "CPE 게이트웨이"를 제공합니다. 이것은 일반적으로 셀룰러 또는 유선망을 통해 인터넷과 연결되는 브로드밴드 라우터를 말합니다. 가정과 회사에서 모두 사용됩니다. 이 구성에서 게이트웨이는 보통 네트워크 운영사가 관리와 구성을 맡습니다.

3.4 포괄 모델

평가 또는 설계 중인 엔드포인트가 어떤 유형이든 하드웨어와 실행의 관점에서 하위구성요소 모델은 모두 비슷합니다.

- 중앙처리장치(CPU)가 애플리케이션 코드를 실행해야 합니다
- CPU 는 데이터와 실행 코드를 항구적 저장장치에서 불러오거나 거기에 저장해야 합니다
- CPU 는 임시 저장장치에서 데이터를 연산해야 합니다
- 신뢰 컴퓨팅 기반을 이용해 환경을 인증해야 합니다
- 장치는 IoT 생태계와 통신해야 합니다

주지하듯, 경량 엔드포인트는 복잡한 엔드포인트나 게이트웨이보다 저장 공간이 적고 연산 능력도 낮습니다. 보안 능력 또한 대체로 떨어집니다.

포괄 모델에서 가장 중요한 측면은 각 유형의 엔드포인트 장치가 한 가지 기본 기능을 수행해야 한다는 점입니다. 바로 특정 애플리케이션을 실행하는, 믿을 수 있는 고품질의 안전한 플랫폼을 지정하는 것입니다. 다시 말하면, 엔지니어링 팀에서는 스마트폰, 클라우드 서버, 메인프레임

등 복잡한 컴퓨팅 플랫폼과 마찬가지로 고품질의 애플리케이션을 안정적으로 실행하거나 피어와 안전하게 공유하기에 앞서 하드웨어가 애플리케이션에게 *믿을 만한* 플랫폼을 제시하게 해야 합니다.

IoT 엔드포인트는 속성상 다른 엔드포인트의 네트워크에 참여합니다. 상위 서비스의 영향이나 참여 없이 어떤 작업을 수행하는 독립 장치가 아닙니다. 어떤 장치의 신뢰도를 높이고 보안 또는 안정성의 허점으로 인한 배상책임의 여지를 없애려면 전체 IoT 시스템의 *신뢰도*는 엔드포인트 하드웨어의 구성에서 *시작*한다는 생각으로 모든 엔드포인트를 설계해야 합니다.

이 같은 시점에서 보면 아주 간단한 엔드포인트 장치조차도 안정성과 고품질, 보안을 갖춰야 한다는 것이 분명해집니다. 수백 만 개에 달하는 장치로 이루어진 네트워크의 일원이 될 것이기 때문입니다. 한 엔드포인트의 거동 방식이 분명 전체 IoT 생태계에 영향을 미치게 됩니다. 그러므로 엔지니어링 팀에서는 내장되는 어떤 장치의 물리적 속성을 너머 아키텍처 설계의 영향을 검토해야 합니다. 전체 IoT 생태계의 보안과 안정성, 품질 니즈의 관점에서 생각해야 합니다.

4 보안 모델

엔드포인트의 보안은 구성요소의 관점에서 평가할 수 있습니다. 엔지니어와 공격자 모두 어떤 엔드포인트를 이루는 각 구성요소를 평가하면 큰 수고 없이 시스템을 침해할 수 있는 유력한 공격 방법을 찾아낼 수 있습니다.

위에서 정의한 포괄 엔드포인트 모델을 이용하면 사용되는 구성요소를 높은 수준에서 평가할 수 있습니다. 각 구성요소를 높은 관점에서 보면 많이 사용되면서도 보안이 취약할 가능성이 높은 기술이 보입니다. 공격의 성공에 필요한 전문성과 장비, 비용이 낮은 것부터 구성요소를 나열해 보면 애널리스트건 공격자건 엔드포인트의 보안 허점을 빠르게 평가하는 공격 모델을 만들 수 있습니다.

엔드포인트 생태계에는 리소스와 인프라 접근성, 전문성에 따라 공격자의 타깃이 되는 위협면(threat surface)이 몇 군데 있습니다. 예를 들면 다음과 같습니다.

- 네트워크 통신
- 접속 가능한 네트워크 서비스
- 콘솔 액세스
- 로컬 버스 통신

- 칩 액세스

4.1 네트워크 통신에 대한 공격

IoT 엔드포인트를 침해하고자 할 때 첫 단계이면서 가장 간단한 조치는 보통 통신 모델의 약점을 노리는 것입니다. 애널리스트는 해당 통신 모델이 통신 보안 모범 사례를 반영하고 있는지를 볼 것입니다. 애널리스트가 서비스 생태계에서 엔드포인트 식별에 사용될 로그인 자격증명이나 통신 토큰, 그 외 식별자를 쉽게 획득할 수 있다면 장치는 이미 침해를 당한 것입니다.

이 전략은 아주 간단할 수도 있고 지극히 어려울 수도 있습니다. 그 이유는 통신 채널을 통해 전달되는 평문 데이터에 대한 애널리스트 또는 공격자의 접근성 때문입니다. 충분히 장비를 갖춘 애널리스트라면 BLE 나 802.15.4, 그 외 대중적인 프로토콜의 통신을 가로채는 기술을 이미 확보하고 있을 것입니다. 어떤 엔드포인트의 통신을 대상으로 '중간자'(man-in-the-middle) 공격을 실시할 때 엔드포인트가 바뀌는 것은 거의 없으므로 공격자는 매우 유리한 위치에 서게 됩니다. 이런 류의 공격은 별다른 수고 없이도 가능합니다.

그러나 통신 모델이 모범 사례를 채용해 데이터의 기밀유지와 무결성을 의무화하고 있다면 공격자가 가치 있는 비밀에 접근하기는 훨씬 더 어렵습니다. 그러면 공격자는 다음으로 쉬운 공격 모델로 향하게 됩니다.

4.2 접속 가능한 네트워크 서비스 공격

IoT 엔드포인트 공격의 다음 단계는 개방된 네트워크 서비스를 평가하는 것입니다. 먼저, 엔드포인트에서 나오는 송신 메시지를 포착해 메시지 안에 즉시 사용 가능한 비밀이 들어 있는지 확인합니다. 이렇게 하면 공격자가 엔드포인트 자체에서 비밀을 추출하느라 들이는 노력을 줄일 수 있습니다. 송신 통신 보안 모델이 건전하다면, 네트워크 서비스를 스캔해 네트워크에서 엔드포인트의 운영체제에 접근할 수 있는지 평가합니다.

NMap[8]이라는 도구로 평가를 실시해 네트워크 포트가 열려 있는지 파악합니다. BLE 나 IEEE 802.15.4 네트워크처럼 네트워크 토폴로지에 IP 기능이 없더라도 공격자는 쉽게 구할 수 있는 도구를 이용해 무선 프로토콜로 엔드포인트에 연결할 수 있습니다.

이후 공격자는 엔드포인트에 메시지를 보내 엔드포인트를 조작해 명령을 실행하게 하거나 운영체제에 대한 원격 콘솔 접속을 제공 받을 수 있는지 파악하려고 합니다. 흔히 쓰는 방법은 SSH(Secure Shell)나 telnet 같은 네트워크 로그인 인터페이스가 있는지 알아보는 것입니다. 기본

로그인 자격증명이 사용되는 경우 공격자가 엔드포인트에 로그인할 수 있을지도 모릅니다. 그렇게 되면 공격자가 로컬 운영체제를 조작할 수 있습니다. 어쩌면 로컬의 취약점을 남용해 권한을 올려 장치에서 비밀을 추출할 수도 있습니다.

또 한 가지 예로는 허술하게 설계된 웹 서비스의 남용이 있습니다. 즉 사용자 입력 필드에서 제어 문자를 제대로 벗겨내지 않는 명령을 공통 게이트웨어 인터페이스(CGI)를 통해 주입하여 로컬 운영체제에서 코드를 실행시키는 것입니다.

4.3 콘솔 액세스 공격

콘솔 액세스는 정확히 말하면 공격이라기보단 전략입니다. 일반적으로, 개발자와 품질관리(QA) 담당자들이 하드웨어와 소프트웨어의 이상을 진단하려면 엔드포인트에서 콘솔을 구동해야 합니다. 그런데 콘솔에서 나오는 정보는 공격자에게 매우 값어치가 있습니다. 또한 콘솔을 통해 공격자가 로컬 및 원격으로 엔드포인트 시스템에 로그인할 수도 있습니다.

대개 로컬 하드웨어는 다음과 같은 방법으로 엔드포인트에서 찾을 수 있습니다.

- 기판에서 TTL 직렬 포트를 나타내는 5 핀 헤더를 찾는 방법
- CPU 또는 MCU 의 사양을 검색해 UART 핀을 찾아내는 방법

TTL 포트 찾기는 멀티미터를 이용하면 됩니다. 핀이 TTL 의 일반적 전압 규격을 준수하기 때문입니다. 논리 분석기를 써서 하드웨어 핀을 지나는 아무 직렬 데이터의 보드율을 추정하는 방법도 있습니다. 애널리스트는 어떤 콘솔이 로컬 하드웨어에서 이용 가능한지 여부를 빠르게 판단할 수 있습니다.

콘솔 포트에 액세스하기만 하면 엔드포인트 장치에서 명령어 프롬프트에 바로 접속되는 경우가 많습니다. 때로 로그인 자격증명을 요구 받기도 하지만 대개 추측 가능합니다. 인터넷 상에서 다른 사람이 그 로그인 자격증명을 찾아냈고 엔드포인트 로그인 자격증명이 모두 같다면 애널리스트는 온라인에서 구글 검색을 실시해 다른 누군가 그 자격증명을 게시했는지 알아보기만 하면 됩니다.

원격 콘솔 액세스는 진단 네트워크 프로토콜이나 콘솔 액세스 프로토콜(예: SSH, telnet), 그 외 수단을 통해 얻을 수 있습니다. 이와 같은 액세스 방법은 평가를 통해 공격자가 액세스 채널을 조작하여 원격 콘솔에 대한 액세스를 부여 받을 수 있는지 판단해야 합니다.

4.4 로컬 버스 통신 공격

콘솔을 통해 명령어 프롬프트를 얻지 못했을 때 공격자 또는 애널리스트의 다음 행선지는 하드웨어를 검사해 엔드포인트가 얼마나 쉽게 침해되는지 판단하는 것입니다. 방법은 여러 가지이나 다음과 같이 간단한 단계가 있습니다.

- 쓰기가 가능한 매체가 있거나 변경할 수 있는가?
- 하드웨어 버스를 통해 크립토그래픽 비밀이 노출된 상태로 전송되는가?
- 하드웨어 회로에 공격자에게 유리하도록 애플리케이션 또는 운영체제의 거동에 영향을 미치는 메시지를 넣을 수 있는가?

가장 단순한 공격법은 쓰기 가능한 매체가 있는지 찾아보는 것입니다. 쓰기 가능한 외부 메모리(SD/MMC) 카드처럼 변경하기 쉬운 매체가 그 대상입니다. NVRAM 칩이나 EEPROM 도 애플리케이션이나 구성의 변경으로 명령 프롬프트 액세스 또는 안전하게 저장된 토큰에 대한 액세스를 허용하도록 바꿀 수 있습니다.

이 벡터에 적절한 보안조치가 마련돼 있다면 애널리스트는 크립토그래픽 상태의 비밀이 하드웨어 버스를 통해 노출 상태로 전송되는지 판단하게 됩니다. 여기에는 논리 분석기를 이용해 EEPROM 과 CPU 간, 마이크로컨트롤러와 SPI 연결 네트워크 어댑터 간, 또는 공격 간 주고 받는 메시지를 가로채는 방법이 있을 수 있습니다. 이런 공격들은 공격과 남용되는 기술의 복잡도에 따라 아주 간단하고 빠를 수도 있고 복잡하고 비쌀 수도 있습니다.

공격자가 위의 방법을 써서 값어치 있는 비밀을 가로채지 못한다면, 하드웨어 버스에 메시지를 주입하여 엔드포인트에서 실행 중인 애플리케이션의 거동을 바꾸려 할 수도 있습니다. 이것은 애플리케이션만의 데이터와 컨텍스트를 평가할 수 있는 고도의 전문성과 장비, 능력을 요하는 어려운 공격입니다.

4.5 칩 액세스 공격

앞서 설명한 공격이 지나치게 복잡하거나 비싸다면, 하드웨어를 상대로 한 공격은 그보다 더 복잡합니다. 여기서는 대개 칩 또는 기판을 이루는 여러 구성요소의 보안을 남용합니다. 주요 사례는 다음과 같습니다.

- 마이크로컨트롤러 또는 CPU 의 디캐핑(decapping)
- 내부 EEPROM 또는 NVRAM 에서 비밀 추출하기
- 내부 SRAM 메시지 가로채기

- X 레이 분석이나 FIB 역설계 실시

위와 같은 공격에는 모두 높은 수준의 기술과 전기공학 지식, 방대한 장비가 필요합니다.
대부분의 조직에서는 공격자가 위와 같은 방법을 써서 자사 제품을 역설계하지 않을까 걱정할 필요는 없지만 한 가지 가능성으로서 검토는 필요합니다. 그 이유는 엔드포인트 장치에 고유한 크립토그래픽 비밀이 프로비저닝돼 있지 않다면 공격을 *한 번*한 해도 되기 때문입니다.

고유한 크립토그래픽 비밀이 없다면 이 등급에서 한 번만 공격해도 전체 제품 라인에 영향을 줄 수 있는 비밀이 추출될 것입니다. 이것은 심각한 문제입니다. 어떤 이유로든 데이터가 대중에게 공개되면 패치가 배포될 때까지 공격과 남용의 대상이 될 것이기 때문입니다.

5 자주 하는 보안 질문

본 문서에서 엔드포인트 보안은 우선순위 별 권고사항으로 나뉩니다. 하지만 실무에서는 실용적인 부분부터 권고사항을 평가하면 됩니다. 엔지니어링 팀에서는 보통 기술적 목표나 사업에 영향을 주는 목표를 기준으로 권고사항을 작성하기 시작합니다. 본 절에서는 엔드포인트의 관점에서 공통의 목표를 기술하고 그 목표를 달성하기 위한 권고사항을 제시합니다.

5.1 클로닝에는 어떻게 대처해야 합니까?

지적재산 보호는 요즘 기업에게 중요한 목표입니다. 엔드포인트 제품을 만드는 데 쓰이는 하드웨어와 펌웨어, 통신 기술은 시간과 전문성, 자금을 들여 탄생한 것으로 기업 입장에서는 그것이 다른 비양심적인 기업의 브랜드나 사업의 구축에 쓰이는 것을 좋아할리 만무합니다. 그렇지만 회사가 아무리 노력을 해도 누군가 똑같은 하드웨어 부품을 가져다가 그럴듯한 "가짜" 또는 "복제품"(clone)을 만들 수도 있습니다. 회사로서는 합법적 계약과 제휴 외에 이것을 막을 도리가 없습니다. 하지만 경제적인 방법으로 누군가 그런 복제품을 *사용하지* 못하게 할 수는 있습니다.

엔드포인트 통신에 인증을 결합하면 각 엔드포인트가 IoT 서비스 업체에서 제조했음을 크립토그래픽하게 입증할 수 있습니다. 백엔드 서비스, 즉 피어 엔드포인트가 엔드포인트 장치와 통신할 때마다 엔드포인트에게 스스로 인증하게 하여 유효한 엔드포인트와 *복제품*을 구별할 수 있습니다. 장치가 그렇게 하지 못한다면 피어 또는 서비스는 그 엔드포인트를 거절하면 됩니다. 이를 위해서는 다음 권고사항이 제 기능을 해야 합니다.

- 엔드포인트 ID 인증
- 제대로 엔지니어링되지 않았거나 구현되지 않은 상호 인증

5.2 엔드포인트 ID 의 보안을 확보하는 방법은 무엇입니까?

엔드포인트를 제대로 인증하려면 엔지니어가 엔드포인트의 크립토그래픽 ID 를 신뢰할 수 있어야 합니다. 이것은 보기보다 복잡하며 목표 달성을 위해서는 프로세스와 정책, 기술의 조합이 필요합니다. *신뢰 컴퓨팅 기반* 권고사항에서 자세히 설명하겠지만, 인증 토큰이 엔드포인트에 인코딩되는 방식에 따라 전체 시스템의 보안 수준이 결정됩니다.

엔드포인트 아키텍처 중에는 공격자가 목표 장치에서 크립토그래픽 토큰(있다면)을 복사해 사칭하기면 하면 되는 것이 많습니다. IoT 서비스 업체가 제조한 엔드포인트가 모두 동일한 크립토그래픽 토큰을 이용한다면 공격자가 어느 한 토큰 세트만 침해해도 어떤 장치든 사칭할 수 있을지도 모릅니다.

그러므로 적절한 TCB 를 구축하려면 다음 권고사항을 구현해야 합니다.

- 신뢰 컴퓨팅 기반(TCB)의 구현
- 트러스트 앵커 활용
- 탬퍼링(tamper)에 강한 트러스트 앵커 활용
- TCB 에 API 활용
- 검증된 무작위 번호 생성기 이용
- 탬퍼링에 강한 제품 케이스 이용
- 트러스트 앵커를 상대로 기밀유지와 무결성의 의무화

5.3 트러스트 앵커에 대한 공격의 여파를 어떻게 줄이면 될까요?

장치가 제조되어 프로비저닝되는 방식도 생산 중인 엔드포인트의 보안에 큰 영향을 미친다는 점 또한 유념해야 합니다. 제조 공정에 따라 엔드포인트가 키로 안전하게 인코딩되거나 되지 않습니다. 이행과 프로비저닝 프로세스에 따라 엔드포인트가 특정 소비자와 연계되는 방식이 결정되며, 장치가 연계 전후에 침해될 가능성은 없는지도 알 수 있습니다.

- 공급망 보안 검토
- 이행 전 각 엔드포인트 장치의 개인화
- 각 엔드포인트의 고유한 프로비저닝
- 프라이버시와 고유한 엔드포인트 식별자

5.4 엔드포인트 사칭의 여지를 낮추는 방법은 무엇입니까?

공격자의 입장에서 사업상의 이유로 장치를 복제한 후 해야 할 일은 어떤 사람이나 *특정* 장치를 사칭하는 것입니다. 이것은 특정 개인의 공격과 직접 관련이 있을 수도 있고 없을 수도 있습니다. 블루투스 기반의 디지털 락 같은 보안 장치를 우회할 목적으로 단순히 어떤 장치를 사칭하는 것일 수도 있습니다.

그 이유가 무엇이든, 이 공격은 TCB 와 개인화, 인증으로 대응할 수 있습니다. 그 외에 다음 방법도 가능합니다.

- 완전 순방향 기밀성(PFS)
- 메모리의 핵심부 잠금

5.5 서비스나 피어의 사칭을 봉쇄하는 방법은 무엇입니까?

어느 IoT 네트워크든 그 안에는 엔드포인트 장치만 있는 것은 아닙니다. 네트워크 서비스와 피어도 존재합니다. 엔드포인트는 서비스의 인증을 받아야 하지만 서비스도 엔드포인트의 인증을 받아야 합니다. 그러면 애플리케이션 업데이트 같은 중요한 서비스가 와해돼 네트워크가 더 침해될 염려가 없습니다.

- 엔드포인트 통신 보안
- 완전 순방향 기밀성(PFS)
- 검증된 무작위 번호 생성기 이용
- 무선 애플리케이션 업데이트
- 제대로 엔지니어링되지 않았거나 구현되지 않은 상호 인증
- 무단 메타데이터 수확(harvesting)

5.6 펌웨어와 소프트웨어의 탬퍼링을 봉쇄할 방법은 무엇일까요?

신뢰 기반이 마련되면 엔드포인트는 믿을 수 있는 구성요소에서 인증을 할 수 있습니다. 그러면 엔드포인트가 신뢰의 근간을 확립하고 다음 단계 애플리케이션이 공격자에게 의도적이든 아니든(예컨대 고장난 NVRAM 을 통해) 변경되지 않게 할 수 있습니다. 그 방법은 다음과 같습니다.

- 최소한 타당성 있는 실행 플랫폼 (애플리케이션 롤백)
- 크립토그래픽한 방법으로 애플리케이션 이미지에 서명
- 내부 EEPROM 밖에서 부트로딩
- 메모리의 핵심부 잠금
- 안전하지 않은 부트로더
- 탬퍼링에 강한 제품 케이스 이용

5.7 원격 코드 실행의 가능성을 낮추는 방법은 무엇일까요?

물리적 펌웨어나 소프트웨어를 이용한 탬퍼링에서 원하는 결과가 나오지 않으면 공격자는 부트로더, 또는 버스나 네트워크 인터페이스를 통해 통신하는 애플리케이션을 상대로 코드 실행처럼 더 복잡한 공격으로 옮겨갈지도 모릅니다. 이 장의 앞에서 설명했듯이 네트워크의 피어가 모두 인증을 받는다면 공격자가 악성 콘텐츠를 주입하기는 훨씬 더 어려워집니다. 그러나 대부분의 장치는 다른 조직의 장치를 상대할 때 공용 통신의 모습이 어느 정도 필요합니다. 그러므로 데이터의 원천에 제대로 제한을 가하지 못할 수도 있습니다.

따라서 원격 인터페이스와 실물 인터페이스 양쪽의 컴퓨터에 데이터를 진입시킬 때에는 면밀한 검토가 필요합니다. 애플리케이션의 익스플로잇 가능성을 제한하고 한 애플리케이션 *이*/침해됐을 때 노출을 제한하려면 다음 항목을 고려해야 합니다.

- 메모리 보호장치의 의무화
- 비밀에 내부 메모리 사용
- 무선 애플리케이션 업데이트
- 적절한 수준의 권한으로 애플리케이션 실행
- 애플리케이션 아키텍처에서 작업의 분리 의무화
- 언어 보안 의무화

- 운영체제 수준의 보안 강화 의무화
- 사용자 인터페이스 보안
- 3 자 코드 감사

5.8 무단 디버깅이나 아키텍처의 계측을 차단하는 방법은 무엇입니까?

아키텍처에 관한 지식과 디버깅 도구를 지닌 공격자라면 십중팔구 표준 디버깅과 진단 유틸리티를 계측(instrument)해 시스템 비밀에 액세스하거나 유익한 코드를 변경 또는 주입하려고 할 것입니다. 공격자의 이 같은 능력을 제한한다면 소비자가 탐지하지 못하는 빠르고 은밀한 공격의 가능성을 낮출 수 있습니다.

- 탬퍼링(tamper)에 강한 트러스트 앵커 활용
- 로깅과 진단
- 메모리의 핵심부 잠금
- 이상 탐지
- 탬퍼링에 강한 제품 케이스 이용
- 디버깅과 시험 기술 사용 안 함
- 사용자 인터페이스 보안

5.9 사이드 채널 공격에 대처하는 방법은 무엇입니까?

일반적인 옵션이 모두 떨어지면 공격자는 장치에서 비밀을 추출하기 위해 더 정교한 공격을 준비할 것입니다. 공격은 하드웨어의 거동을 평가해 거동 속 어떤 패턴이 어떤 값, 이를 테면 1 이나 0, 특정 명령어와 같은지 확인하려고 합니다. 이것이 계속되다 보면 임베디드 시스템에서 처리되고 있는 데이터를 역설계하는 능력이 애널리스트에게 생깁니다.

또한 공격자가 비싼 분석 기술을 동원해 장치에서 비밀을 추출하거나 지극히 작은 회로를 만들어 실리콘에서 보안 레이어를 통해 연결을 구현할 수도 있습니다. 이런 공격은 대처하기가 매우 어렵지만 시행자도 다음과 같은 이유로 공격에 나서기가 쉽지 않습니다.

- 이행 전 각 엔드포인트 장치의 개인화
- 비밀에 내부 메모리 사용
- 탬퍼링에 강한 제품 케이스 이용
- 주변기기 기반의 공격을 통해 손상된 메모리

- 환경 록아웃(lock-out) 임계값 구현
- 전력 경고 임계값 의무화
- 장치의 퇴역과 일몰
- 보이지 않는 컴포넌트와 신뢰하지 않는 브리지 차단
- 콜드 부팅 공격 차단
- 집속 이온빔과 X 레이에 대한 대처

5.10 보안 원격 관리는 어떻게 구현해야 합니까?

원격 관리는 IoT 엔드포인트 라이프사이클의 핵심이 되는 요소로 관리에 사용되는 채널이 남용되지 않도록 보호 대책을 강구해야 합니다. 이것은 알 수 없는 3 자 공격자와의 문제만은 아닙니다. 내부 남용도 소비자의 영향권 내에서 또는 IoT 서비스 업체 안에서 발생할 수 있습니다.

- 엔드포인트 비밀번호 관리
- 원격 엔드포인트 관리
- 로깅과 진단
- 완전 순방향 기밀성(PFS)
- 사설 APN 의 이용

5.11 침해 당한 엔드포인트는 어떻게 탐지합니까?

엔드포인트의 아키텍처에 따라 다르지만, 장치가 정상 작동한다면 하드웨어나 펌웨어가 탬퍼링 당했는지 판단하기가 불가능할 수도 있습니다. 그러나 인프라를 추적하고 로깅하고 이상 발견 시 경보를 발령한다면 장치의 침해는 이상 거동으로 탐지할 수 있습니다. 다음 권고사항을 고려하기 바랍니다.

- 이상 탐지
- 탬퍼링에 강한 제품 케이스 이용
- 전력 경고 임계값 의무화

5.12 백엔드 연결 없이 장치를 안전하게 배포하는 방법은 무엇입니까?

백엔드 환경의 연결이 가능하지도 바람직하지도 않을 때가 있습니다. 이런 환경에서는 보안 확보가 더더욱 어렵습니다. 보안 키와 ID, 동적 인증 메커니즘을 관리할 수 없기 때문입니다. 그러나 어느 정도의 보안은 확보할 수 있습니다. 다음 사항을 생각해보기 바랍니다.

- 신뢰 컴퓨팅 기반(TCB)의 구현
- 조직의 신뢰 기반 정의
- 이행 전 각 엔드포인트 장치의 개인화
- 완전 순방향 기밀성(PFS)
- 엔드포인트 ID 인증
- 백엔드 연결 기능이 없는 환경

5.13 소비자의 프라이버시는 어떻게 확보합니까?

소비자 프라이버시는 엔드포인트 기술뿐만 아니라 IoT 제품 또는 서비스 전체를 심도 있게 분석해야 하는 까다로운 문제입니다. 전체 시스템의 각 구성요소를 분석해 프라이버시 침해 가능성을 확인해야 합니다. 다음 권고사항을 통해 프라이버시 의무화에 관해 더 알아보기 바랍니다.

- 완전 순방향 기밀성(PFS)
- 엔드포인트 통신 보안
- 프라이버스 관리
- 프라이버시 및 고유 엔드포인트 ID
- 사설 APN 이용
- 무단 메타데이터 수확(harvesting)
- 명확하지 않은 보안 위험(시 스루 월)
- 적법한 가로채기

5.14 프라이버시와 보안을 의무화하면서 사용자 안전을 확보하는 방법은 무엇입니까?

안전은 애플리케이션과 그 목적, 그것이 사용될 환경, 소비자의 유형, 사용되는 커뮤니케이션 기술을 감안해 검토해야 합니다. 안전과 보안이 서로 조금씩 양보해야 할 것 같은 때도

있습니다. 하지만 그렇지 않습니다. 오히려 안전과 보안 모두를 지키기 위해 아키텍처 모델을 바꿔야 할지도 모릅니다. 가능하다면 안전을 위해 보안을 버리는 일은 피해야 합니다. 가능하다면 둘 다 지켜야 합니다. 이것은 철학과 관련된 권고사항이지만 엔지니어링 팀에서는 안전을 끊임 없이 검토해야 합니다. IoT 안전에 관해 논의를 시작할 때 다음 권고사항을 검토하기 바랍니다.

- 안전 위주의 분석
- 서비스의 의도적, 비의도적 거부
- 적법한 가로채기
- 공급망 보안 검토

5.15 해결을 기대해서는 안 되는 문제는 무엇입니까?

어느 시스템이든 물리학의 법칙이나 비용, 단순히 기술 솔루션의 부족 때문에 해결이 불가능한 위험이 있기 마련입니다. 그 중 몇 가지를 예로 들자면 다음과 같습니다.

- 서비스의 의도적, 비의도적 거부
- 보이지 않는 컴포넌트와 신뢰하지 않는 브리지 차단
- 명확하지 않은 보안 위험(시 스루 월)
- 집속 이온빔과 X 레이에 대한 대처
- 공급망 보안 검토
- 적법한 가로채기

6 핵심 권고사항

보안 엔드포인트를 개발할 때에는 다음 권고사항을 반드시 구현해야 합니다. 아래 권고사항은 보안 엔드포인트 아키텍처와 직결되는 아주 중요한 것입니다. 이것을 구현하지 않는다면 엔드포인트는 보안이 불완전해 공격자의 먹잇감이 되고 말 것입니다.

6.1 엔드포인트 신뢰 컴퓨팅 기반의 구현

어느 것이든 임베디드 시스템의 보안을 확보하는 첫 단계는 신뢰 컴퓨팅 기반(TCB)을 정의하는 것입니다. 엔드포인트(또는 유사한 임베디드 장치)라는 맥락에서, TCB 란 엔드포인트의 무결성을 지켜주고 네트워크 피어와 상호 인증을 실시하고 통신과 애플리케이션의 보안을 관리하는 하드웨어, 소프트웨어, 프로토콜의 집합체입니다.

TCB 의 핵심은 트러스트 앵커, 즉 사전 공유키(PSK)나 비대칭 키 같은 크립토그래픽 비밀을 저장하고 처리하는 보안 하드웨어 기술을 말합니다. UICC 와 같은 트러스트 앵커는 네트워크 통신 시 피어의 인증에 쓸 수 있을 뿐만 아니라 증강을 통해 엔드포인트 애플리케이션 보안에 유용한 데이터를 저장할 수도 있습니다.

트러스트 앵커를 선택해 엔드포인트 솔루션에 통합하고 나면, 트러스트 앵커를 전체 TCB 패키지에 통합하는 라이브러리를 선택하거나 설계할 수 있습니다. 이 TCB 를 통해 운영체제와 엔드포인트의 주력 애플리케이션이 장치뿐만 아니라 네트워크의 전반적 보안을 더욱 쉽게 관리할 수 있습니다.

여기서, 엔지니어링 팀은 솔루션에 맞는 트러스트 앵커를 선택하는 것이 중요합니다. 트러스트 앵커와 TCB 의 조합에 따라 보안 수준이 달라지기 때문입니다. 일부 조합과 트러스트 앵커 구현체는 잘못된 보안 감각을 낳기도 합니다.

신뢰 컴퓨팅 기반의 보편적 유형을 소개하자면 다음과 같습니다. 순서는 보안이 낮은 것에서 높은 것입니다.

- 구현되지 않음(평문)
- 고정된 사전공유 키(PSK)
- 고정된 공용키
- 개인화된 PSK
- 개인화된 공용키

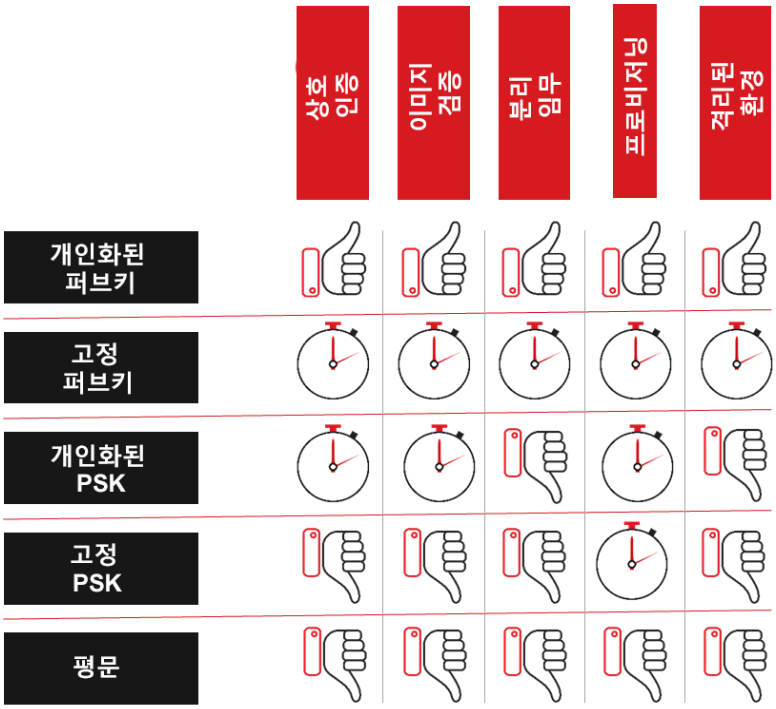

그림 3 - TCB 유형별에 따라 제공되는 보안 대책.

위 그림에서, 각 TCB 종의 기능에 가중치가 부여돼 있습니다. 엄지손가락이 아래를 향하고 있는 아이콘은 해당 TCB 모델이 아래 첫 줄에 명시된 보안 전략을 수용하지 못한다는 뜻입니다. 스톱워치 아이콘은 보안 전략을 쓸 수는 있지만 어느 정도 시간이 지나면 보안이 뚫린다는 뜻입니다. 엄지손가락이 위를 향하고 있는 아이콘은 보안 전략을 탄탄하게 구현할 수 있고 보안 전략의 수명이 길 가능성이 높다는 뜻입니다.

TCB 로 IoT 제품과 서비스의 여러 부분에서 보안을 지킬 수 있지만 본 문서에서는 다음 다섯 가지만을 소개하기로 하겠습니다.

- 실행 가능한 이미지 검증
- 네트워크 피어의 상호 인증
- IoT 보안 아키텍처 내 임무의 분리
- 프로비저닝과 개인화
- 격리된 환경 보안(연결 없는 현장 보안)

*실행 가능한 이미지 검증*을 구현하는 TCB 는 엔드포인트 장치에 로드돼 실행될 실행 가능 이미지 각각을 크립토그래픽 방식으로 검증하여 장치의 보안을 지킵니다. 이 과정은 부트로더에서 시작합니다. 부트로더는 실행 다음 단계, 보통은 운영체제 커널을 크립토그래픽 방식으로 검증합니다. 부트로더는 또 운영체제 이미지, 즉 NVRAM 에 저장된 펌웨어 애플리케이션 이미지도 검증할 수 있습니다.

*네트워크 피어의 다중 인증*을 구현하는 TCB 는 네트워크 구성요소의 인증에 신뢰 기반을 제공하고 자신을 네트워크 피어에게 크립토그래픽 방식으로 인증합니다. 그러면 네트워크상의 피어가 자기가 주장하는 ID 를 그대로 나타낼 가능성이 높습니다. 가령, 네트워크 피어가 펌웨어 업데이트 서비스를 제공한다고 주장한다면, TCB 는 그 피어를 핵심 IoT 서비스 업체 네트워크로 인증한 후 그 피어에게서 펌웨어 업데이트를 받아들입니다.

*임무 분리*를 구현하는 TCB 는 계층으로 된 키(key)를 이용해 IoT 서비스 업체의 서비스 안에서 여러 가지 구성요소나 서비스를 표시합니다. 예를 들면, 일단의 크립토그래픽 키가 펌웨어 업데이트 서비스를 나타내고 또 다른 키 집단이 "푸시" 서비스를 나타내는 것입니다. 이들 서비스는 기능이 서로 완전히 다르기 때문에 통신 시 같은 크립토그래픽 키와 ID 를 사용해서는 안 됩니다. 그런 의미에서 TCB 는 각각의 ID 를 관리하고 검증해 한 서비스나 기능을 다른 것과 분리해야 합니다. 그렇게 되면 공격자가 크립토그래픽 키 가운데 *하나*를 침해하더라도 IoT 서비스 인프라 전체를 침해할 위험은 제한됩니다. 다시 말하면, 공격자가 "푸시 서비스"의 키를 침해하더라도 펌웨어 업데이트 서비스를 사칭할 수는 없다는 것입니다.

*개인화와 프로비저닝*을 구현하는 TCB 는 엔드포인트가 같은 유형의 다른 엔드포인트와는 크립토그래픽적으로 고유한 ID 를 갖게 합니다. 또한 모든 통신 ID 에 보호 대책을 제공해 프라이버시 유출이나 추적 가능성을 낮추는 역할도 합니다.

*격리된 환경 보안*을 구현하는 TCB 는 프로세스에 도와줄 백엔드 서비스가 없어도 피어의 진위와 데이터의 기밀성 및 무결성을 검증하는 정책과 절차를 집행합니다. 다시 말하면, 백엔드

서비스에 대한 통신이 장시간 두절되어도 로컬화된 IoT 생태계는 높은 수준의 보안을 유지한 태계속 기능을 할 수 있다는 것입니다. 격리된 환경의 무결성은 시간이 지나면서 약화되지만 잘 정의돼 *격리된 환경 보안*을 실행하는 TCB 는 네트워크의 회복력을 높여주고 환경이 안전하다고 간주되는 시간을 늘려줄 수 있습니다.

이 맥락에서 *개인화*란 특정 트러스트 앵커와 연계된 고유 키의 집합을 말합니다. 개인화 과정은 고유한 키를 생성하고 설치하는 단계, 키와 고유한 칩을 연결하는 단계, 이 정보와 관련 메타데이터를 관련 당국에게 배포하는 단계로 진행됩니다. 이렇게 하면 각 칩이 고유한 크립토그래픽 ID 를 갖게 됩니다. 여기서 *고정*이란 매 엔드포인트에서 사용되는 동일한 키의 집합을 말합니다.

TCB 는 임베디드 시스템에 존재할 수도 있는 보안 문제를 거의 모두 다 해결할 수 있지만 TCB 가 반드시 해결해야 하는 핵심적인 문제가 몇 가지 있습니다

- 엔드포인트 애플리케이션 이미지 검증
- 네트워크 인증 및/또는 피어 인증
- 임무의 분리
- 프로비저닝과 개인화
- 격리된 환경(연결 없는 현장) 프로비저닝과 통신
- 무작위화

TCB 를 구현하지 않으면 보안이 부실해지지만 그 외 보편적 TCB 구현체에게는 해결해야 하는 미묘한 문제가 있습니다. 이 미묘한 문제를 해결하지 않으면 보안에 큰 허점이 발생할 수도 있습니다.

6.1.1 트러스트 앵커 주요 모델

6.1.1.1 정적인 키

정적인 키 구현체는 PSK 든 비대칭 키든 모든 엔드포인트가 동일한 크립토그래픽 비밀을 이용해 문제를 해결하는 솔루션이라고 정의됩니다. 문제마다 다른 키를 써서 해결할 수도 있지만, 이 키는 엔드포인트마다 동일합니다.

이 모델은 TCB 가 해결하는 문제를 효과적으로 실행할 수 있다는 점에서 안전해 보입니다. 그러나 전체 솔루션의 수명은 매우 길기도 하고 지극히 짧기도 합니다. 트러스트 앵커의 보안과

크립토그래픽 알고리즘, 선택한 키의 크기에 따라 공격자가 솔루션을 거의 즉시 파괴할 수 있을지도 모릅니다.

문제는 키가 한 번 침해되면 엔드포인트 시스템이 모두 다 노출 위험에 놓인다는 점입니다. 이렇게 되면 TCB 구현의 의미가 퇴색하고 엔드포인트와 IoT 아키텍처에 솔루션을 구현하느라 들인 시간과 돈이 무의미하게 됩니다. 그러므로 이 모델은 구현하기에 위험한 TCB 입니다. 사실상 시한폭탄이기 때문입니다.

6.1.1.2 개인화 키

PSK 솔루션이나 비대칭 솔루션 어느 것을 구현하더라도 개인화는 TCB 의 효과적 작동에 중요합니다. 개인화는 공격자가 침해 당한 트러스트 앵커를 이용해 IoT 생태계 전체의 보안을 전복하는 능력을 무력화합니다. 공격자가 한 번에 엔드포인트 하나만 침해할 수 있고 그것도 물리적으로 접근해야만 가능하다면 IoT 기술을 폭넓게 침해하는 일은 느리고 비싸고 복잡한 과정이 될 것입니다. 이것은 기업에게 큰 비교우위입니다.

지난 몇 십년 사이 셀룰러 통신의 표준이 발전한 덕분에 네트워크 운영업자는 UICC 같은 트러스트 앵커의 개인화를 위한 PSK 모델을 완성했습니다. 그 결과 UICC 는 때로 IoT 엔드포인트의 애플리케이션 트러스트 앵커 역할을 해 IoT 애플리케이션을 위한 경제적인 보안 솔루션 형성에 기여하도록 프로비저닝이 가능합니다. 가까운 장래에 eUICC 가 나오면 이 기능을 이미 현장에 배포된 eUICC 에서도 구현할 수 있습니다.

현재로서는 개인화 키 기술이 트러스트 앵커에게 가장 효과적인 보안 솔루션입니다. IoT 에 구현된 TCB 는 개인화 TCB 솔루션을 기반으로 해야 합니다. IoT 서비스 업체는 네트워크 운영업자와 협의해 UICC 또는 SIM 를 애플리케이션급의 트러스트 앵커로 구현할 수 있는지 판단해야 합니다.

6.1.2 TCB 프로토콜과 기술

TCB 는 트러스트 앵커외에 프로토콜과 정책, 소프트웨어 정책까지 갖춰 IoT 제품이나 서비스 전반에 대한 보안을 담당해야 합니다. 셀룰러 기반의 표준 트러스트 앵커를 이용할 때 좋은 점 한 가지는 네트워크 운영업자에게 이미 존재하는 프로비저닝 및 개인화 소프트웨어를 이용할 수 있다는 점입니다. 다음과 같은 기술과 프로토콜, 패키지가 TCB 와 함께 네트워크에 대한 엔드포인트의 인증을 지원합니다.

- oneM2M TS-0003 에 명시된 oneM2M SM UICC 애플리케이션

- 범용 부트스트래핑 아키텍처(GBA) 3GPP TS 33.220 (부록 A 참조)

위와 같은 기술을 이용하면 프로비저닝과 개인화의 구현이 빨라집니다. 노련한 엔지니어와 보안 애널리스트들이 라이브러리와 프로토콜을 이미 조사해 놓았기 때문입니다. 그러나 이들 프로토콜 아래에서 엔드포인트가 애플리케이션을 제대로 검증하지 못하거나, 엔드포인트가 메시지를 인증하거나 작업을 승인하지 못할 가능성도 있습니다. TCB 는 다른 프로토콜을 내장해 펌웨어 검증이나 무선 업데이트 메시지 검증과 같은 작업을 완료해야 합니다.

가까운 장래에 eUICC 와 같은 기술이 애플리케이션 측면에서 기능을 보강할 것입니다. 앞선 UICC 는 2 중 사용 기술을 통해 네트워크 보안을 관리하면서도 엔드포인트 자체를 부트스트랩할 수 있습니다. 이것은 중요한 개선입니다. 네트워크 운영업자가 IoT 서비스 업체를 대신해 eUICC 장치를 원격으로 안전하게 관리할 수 있기 때문입니다. 또한 GlobalPlatform Card Specification[15]에 명시된 기밀 카드 콘텐츠 관리 기능을 통하면 네트워크 운영자의 허락 하에 IoT 서비스 생태계에서 여러 주체가 독립적으로 자기 애플리케이션을 관리할 수도 있습니다.

6.1.3 위험

TCB 를 구현하지 않는 것은 전체 IoT 아키텍처에게 결정적인 실패의 원인이 됩니다. 잘 정의된 TCB 가 없다면 트러스트 앵커와 핵심 애플리케이션 간 상호작용이 느슨하게 정의됩니다. 이것은 공격자에게 공격의 여지가 될 수도 있습니다. TCB 는 트러스트 앵커와 핵심 애플리케이션, 네트워크 피어 간 통신의 보안과 신뢰도, 최신 업데이트를 보장합니다. TCB 가 없다면 엔드포인트의 보안 수명주기를 관리할 구심점이 사라집니다.

6.2 트러스트 앵커 활용

엔드포인트가 생태계에 참여하려면 자체 플랫폼의 무결성을 검증할 수 있어야 하고 피어의 신원을 인증할 수 있어야 합니다. 이를 위해서는 트러스트 앵커가 신뢰 컴퓨팅 기반에 내장되어야 합니다.

트러스트 앵커는 별도의 하드웨어 요소로 별도의 칩이거나 CPU 내 보안 코어를 말하며 크립토그래픽 비밀을 안전하게 저장하고 처리할 수 있습니다. UICC 또는 eUICC 장치는 인증 비밀을 저장하는 트러스트 요소로도 쓸 수 있는 보안 기술의 한 종류입니다.

트러스트 요소를 이용하려면 사실상 데이터를 저장하고 검증하고 업데이트하고 처리해야 합니다. 데이터는 크립토그래픽 방식으로 검증이 필요한 비밀 정보이거나 공개 정보가 될 수

있습니다. 어느 경우든, 트러스트 앵커는 메시지와 신원이 인증 가능한지 안전하게 판단할 수 있어야 하고 TCB 에게 인증 또는 크립토그래픽 작업의 결과를 모두 안전하게 말해줄 수 있어야 합니다. 그러면 애플리케이션과 TCB 가 전체 엔드포인트의 보안에 영향을 미칠 중요한 결정을 내릴 수 있습니다. 예컨대, 트러스트 앵커는 엔드포인트를 도와 어떤 네트워크 피어가 패치 배포 서버 등 주요 리소스를 사칭하고 있는지 판단할 수 있습니다. 트러스트 앵커가 어떤 네트워크 피어를 검증하지 못한다면 엔드포인트의 TCB 와 애플리케이션은 그 피어를 상대하지 말아야 하며 가능하다면 사용자에게 해당 네트워크 리소스를 알려야 합니다.

부품 원가의 하락과 수요의 급증으로 트러스트 앵커의 공급은 그 어느 때보다도 풍부합니다. 여기에는 실제 트러스트 앵커 기술뿐만아니라 그 기술과 병행 사용 승인을 받은 라이브러리와 인터페이스도 포함됩니다. 이로써 아주 짧은 시간에 트러스트 앵커 솔루션을 가동할 수 있으며 커스텀 소프트웨어나 허술하게 구현된 표준으로 기술의 수명이 짧아지는 것을 막을 수 있다. 가능하다면 표준을 적용해 보안에 허점이 생겨날 여지를 차단해야 합니다.

경량 엔드포인트에서 트러스트 앵커를 구현할 때 또 한 가지 난제는 구성요소의 크기입니다. 외부 트러스트 앵커를 사용할 경우, 최소 구성요소 프로파일을 유지할 필요가 있습니다. 폼팩터에 UICC 와 같은 기술이 들어 있으면 이 프로파일을 완수하기는 어렵습니다. 그러나 ETSI TS 102 671 표준이면 문제가 되지 않습니다. 크기가 6mm x 5mm 로 아주 작은 폼팩터를 적용하기 때문입니다. UICC 스마트 카드 폼팩터에 이 “MFF1”와 “MFF2” 증강을 적용하면 UICC 가 지원하는 기술을 충분히 이용할 수 있습니다. 동시에 물리적 요건은 최소한에 그칩니다. 여기에 장치에 솔더링되는 필드 프로비저닝 폼팩터를 이용하면 보안은 더 높아져 공격자가 장치의 ID 를 다른 장치로 옮기기가 더욱 어려워집니다.

트러스트 앵커를 개발하고 배포하는 데 따른 비용은 다음과 같습니다.

- 기본 기술(CPU 에 임베드되거나 별도 칩)의 원가
- 필요 시 회로에 기술을 통합하는 비용
- OS 와 TCB 에 드라이버를 엔지니어링하거나 통합하는 비용
- 애플리케이션이 트러스트 앵커를 이용하는 비용
- 필요 시 트러스트 앵커 유지보수
    - 보안 키, 취소 키, 퇴역 ID 유지보수

- 키와 메타데이터의 보안과 관리에 필요한 인프라 유지보수

- 서비스 측에서 앵커 ID 모니터링
  - 필요 시 장치 블랙리스트 실행
- 가용할 경우 캐리어 서비스를 통합해 UICC 와 같은 트러스트 앵커 모니터링과 관리

6.2.1 위험

트러스트 앵커를 이용하지 않는 데 따른 위험은 많지만 모두 한 가지 기본 문제에서 파생되는 것입니다. 공격자가 전체 IoT 생태계와 관련된 키를 훔치는 능력이 그것입니다. 그로 인해 공격자는 다음과 같은 행위가 가능합니다.

- 엔드포인트 ID 복제
- IoT 서비스 사칭
- 허가 받지 않은 패치 또는 업데이트 배포
- 엔드포인트 소프트웨어에 무단 변경 실시

이 같은 보안의 허점은 시간이 지나면서 회사에 큰 비용을 발생시킬 수도 있으며 공격자뿐만 아니라 경쟁사까지도 인프라를 자기 이익을 위해 이용할 수 있게 됩니다.

6.3 탬퍼링(tamper)에 강한 트러스트 앵커 활용

트러스트 앵커 중에는 FIB, 사이드 채널 분석, 글리칭 등 특정 부류의 공격에 대비해 추가로 물리적 보안을 구비한 것도 있습니다. FIB 의 활용처럼 비용 측면에서 보호가 거의 불가능한 공격도 있기는 하지만 트러스트 앵커 제조사는 최신 기술을 이용해 공격을 더욱 비싸게 만들 수 있습니다. 공격의 비용이 클수록 무작위 엔드포인트 장치를 그 목표물로 할 확률은 낮아집니다. 대신 공격은 보상이 비용보다 큰 목표물에 집중할 것입니다.

일부 트러스트 앵커 제조사는 가까운 장래에 Federal Information Processing Standards(FIPS)[10]와 EMVCo[11], Common Criteria 승인을 받은 기술의 변종을 보급할 계획을 세워 두고 있습니다. 신기술을 개발하는 엔지니어는 현재 설계가 가까운 장래에 그에 부합하는 모델로의 전환을 지원할 것인지 판단해야 합니다.

각 표준의 최신 버전을 참고하기 바랍니다. 제조사가 어느 수준의 능력을 제공하는지 확인하기 바랍니다. 구현의 비용과 복잡성 때문에 소비자 기반 장치에게는 불가능에 가까운 보안 수준도 있으니 유의해야 합니다.

6.3.1 위험

탬퍼링에 강한 트러스트 앵커를 쓰지 않는 데 따른 위험은 매우 큽니다. 예컨대, 트러스트 앵커가 단순히 NVRAM 에 임베드된 크립토그래픽 키라면, 그 키를 추출할 도구와 기술을 지닌 공격자 누구라도 잠재적으로 전체 인프라를 전복시킬 수 있습니다. 그러나, 비밀이 탬퍼링에 강한 트러스트 앵커에 저장돼 있다면 비밀을 추출하는 비용이 높아져 비밀이 추출될 가능성은 크게 낮아지며, 잠재적 공격 목표로서 해당 트러스트 앵커의 가치는 저하됩니다.

만일 트러스트 앵커 구현이 약하다면 비밀의 추출로 침해가 발생할 여지는 매우 높아집니다. 침해가 발생한다면 엔지니어링과 아키텍처, 생산, 실행에 소요된 비용은 의미를 잃게 될 것입니다. 이는 막대한 금전적 손실로 이어질 수도 있습니다. 그러므로 조직이 바르게 구현을 하게 하는 것이 중요합니다.

6.4 TCB 에 API 활용

TCB 안에 신뢰기반이 확립되면 TCB 의 기능과 신뢰기반을 효과적으로 수용하는 프로토콜을 사용해야 합니다. API 의 다음을 실현을 담보해야 합니다.

- 서명 인증은 모두 TCB 에서 실시한다
- TCB 에서 개인의 키가 노출되지 않는다
- TCB 가 애플리케이션을 대신해 중요한 교환을 실시할 수 있다
- TCB 가 암호해독을 할 수 있다
- TCB 에서 암호화를 할 수 있다
- TCB 에서 메시지 서명을 실시할 수 있다
- TCB 에서 보안 메시지 패딩을 할 수 있다
- TCB 와 애플리케이션 간 기밀유지와 무결성

이와 같은 기능으로 TCB 는 핵심 보안 자산을 안전하지 않은 애플리케이션이나 하드웨어 환경에 노출하지 않을 수 있습니다. 이것은 이와 같은 요건을 통일성 있게 적용하는 기존 규격을 이용해 가능합니다. 다음의 평가를 고려하기 바랍니다.

- SIM 얼라이언스 공개 모바일 API [12]

- 글로벌플랫폼 보안 요소 액세스 제어 [13]
- 글로벌플랫폼 신뢰 실행 환경(TEE) API 규격 [14]
- 신뢰 컴퓨팅 그룹 (TCG)
- oneM2M TS-0003 [20]

TCB 로 구현 가능한 소프트웨어 라이브러리가 포함된 트러스트 앵커가 많이 나올 것입니다. 이들 라이브러리에는 엔지니어가 TCB 를 상대할 때 쓸 수 있는 API 가 있을 것입니다. 트러스트 앵커가 제공하는 라이브러리가 있다면 선호를 받습니다. 트러스트 앵커 개발 분야의 전문가에게 조사를 받았을 가능성이 높기 때문입니다. 그러나 엔지니어링 팀에서는 본 권고사항에 명시된 요건 목록을 평가하고 라이브러리가 이와 같은 우려를 적절히 감안하고 있는지 확인해야 합니다.

또한 TCB 는 엔드포인트에서 실행되는, 권한이 있는 애플리케이션에서 *만* 액세스가 가능해야 합니다. TCB 인터페이스는 엔드포인트에서 실행되고 있는 애플리케이션 중 권한이나 신뢰가 없는 것(3 자의 것)에서는 액세스가 가능해서는 안 됩니다. 액세스는 모두 요청을 평가하여 신뢰할 수 없는 애플리케이션이 수상한 요청이나 개인정보 위주의 요청을 하면 옵션으로 사용자에게 경고하는 신뢰할 수 있는 서비스를 통해 프록시되어야 합니다.

이 프로토콜을 구현할 때 난제는 해당 데이터의 원점과 TCB 간에 어떤 메시지도 탬퍼링 당하지 않게 하는 것입니다. 그 반대로 마찬가지입니다. 애플리케이션에서 호출 가능한 어떤 EEPROM 세그먼트가 애플리케이션을 대신해 이 기능을 수행할 수 있다면 가장 효과적입니다. API 코드의 핵심부를 내부 EEPROM 으로 격리하고 내부 RAM 을 이용해 메시지를 처리하면 외부 버스에는 덜 중요한 데이터가 노출됩니다.

6.4.1 위험

애플리케이션 프로토콜 인터페이스가 제대로 정의되지 않으면 TCB 의 이용은 의도하지 않은 결과나 부작용을 낳을 수도 있습니다. 미리 프로토콜을 정의하고 논리 또는 보안 문제를 조사해두면 엔지니어링 팀이 나중에 보안 문제로 이어질 수도 있는 결함을 쉽고 효과적으로 찾아낼 수 있습니다. 이렇게 프로토콜의 정의에는 IoT 서비스 업체의 니즈를 포함해 기존 API 의 평가가 포함되어야 합니다. 기성 기술을 찾아낼 수 있다면 그것이 커스텀 솔루션보다는 항상 더 좋습니다.

6.5 조직의 신뢰 기반 정의

조직의 신뢰 기반이란 ID 와 애플리케이션, 통신의 암호화 보안 방법을 관장하는 암호화된 정책과 절차를 말하며, 통신은 크립토그래픽 방식으로 보안을 확보할 수 있고 그렇게 해야 합니다. 강력한 크립토크래피를 고유 대칭 키나 인증서, 공용 키 형태로 사용해야 합니다. 이것은 TCB 에서 이용 가능한 모델과 트러스트 앵커의 기능, 그리고 엔지니어링 팀에게 의미가 있는 것에 따라 달라집니다.

대칭이든 비대칭이든 루트(root) 개인 키를 이용하여 계층에서 사용 중인 다른 키에 디지털 서명을 해야 합니다. 예컨대, Example IoT Company LLC 이라는 가상의 조직이 조직의 신뢰 기반을 만들고자 한다면 믿을 수 있는 머신에 루트 키를 생성할 것입니다. 이 키는 조직의 기반(root)을 나타냅니다. 이어서 독립된 보안 계층을 갖춰야 하는 하위 조직을 대표하는 새 키를 생성할 것입니다. 예는 다음과 같습니다.

- 코드 서명 키
- 서버 통신 키
- 피어 간 통신 키
- 엔드포인트 ID 키
- 마스터 취소 키

위의 키 각각에는 조직의 루트키로 서명을 해야 합니다. 이들 키 모두와 그것에 상응하는 서명, 그리고 루트 키는 TCB 가 이용하는 트러스트 앵커에 저장해야 합니다. 그러고 나면 특정 키와 연동된 애플리케이션이 사용될 때마다 그 애플리케이션은 그 특정 키를 이용해 통신 채널을 통해 전송된 메시지를 검증합니다.

이 모델로 모든 메시지가 크립토그래픽 계층을 통해 보안을 확보할 수 있습니다. 키 형식 별로 임무를 분리하면 동일한 통신 프로세스를 통해 침해 당한 키를 취소할 수 있습니다.

이 방법의 채용에 도움이 되는 기존의 몇 가지 프로토콜은 다음과 같습니다.

- TLS(Transport Layer Security) - 최신 유효 규격
- SSH2(Secure Shell)
- OSCP(Online Certificate Status Protocol) IETF RFC 2560
- GBA(Generic Bootstrapping Architecture) (부록 A 참조) 3GPP TS 33.220

크립토 키가 필요한 서비스를 배포해야 할 때 어려움은 커집니다. 서버 커뮤니케이션 키처럼 보안이 중요한 자산을 인터넷과 연결된 웹서버에 놓을 게 아니라, 해당 서버 티어를 위해 별도의 인증서 또는 키쌍을 만들어야 합니다. 이어서 이 인증서를 서버 통신 키로 서명할 수도 있습니다. 이렇게 하면 어떤 엔드포인트든 서비스가 신뢰기반의 인증을 받았는지 확인할 수 있습니다. 그러면서 중요한 조직의 키는 공격자에게 노출되지 않습니다.

키가 침해를 당하더라도 취소 마스터키로 취소를 인증하여 사용을 중단할 수 있습니다.

두 말할 나위 없이, 조직의 신뢰 기반 내 핵심 키는 모두 인프라의 안전에 매우 중요합니다. 이들 키는 철저히 보호해야 하며 핵심 팀의 믿을 수 있는 내부 구성원만 이용해야 합니다. 승인 받은 하드웨어 보안 모듈(HSM)을 이용해키를 저장하고 액세스하고 이용하기를 강력히 권장합니다.

HSM 은 기술 배포 초기에 가격이 상당히 높기도 하지만 장기적인 재무 효과는 매우 긍정적입니다. 나중에 TCB 와 HSM 으로 해결할 수도 있었던 어떤 위험을 진단하고 대처하느라 포렌식 분석과 엔지니어링에 들어가는 큰 비용을 감안하면 초기 비용은 상대적으로 낮은 편입니다.

6.5.1 위험

조직의 신뢰 기반을 이용하지 않을 경우 어느 한 키의 침해가 전체 생태계의 침해로 이어질 위험이 있습니다. 조직을 계층(hierarchy)으로 나누고 계층에 별도 키를 배포하면 키가 일정 주기로 또 애플리케이션의 우선순위나 키가 관련된 하위조직에 따라 순환합니다. 이렇게 되면 조직의 부문 간 임무가 분할되고 침해 당한 키가 인프라 전체의 보안을 전복시킬 위험도 낮아집니다.

6.6 이행 전 각 엔드포인트 장치의 개인화

엔드포인트 장치는 반드시 크립토그래픽 방식으로 구동해야 공격자와 경쟁사, 애호가가 생산 환경에서 다른 사용자나 장치를 사칭하지 못합니다. 이를 제대로 구현하기 위해서는 제조 단계에서 개인화 프로세스를 실시해야 합니다. 이것은 특정 TCB 솔루션의 제조사를 통하거나 인쇄회로기판 조립체(PCB/A) 공정에서 가능합니다.

개인화 과정을 해결하기 위해서는 다음 항목을 실시해야 합니다.

- 고유한 크립토그래픽 키 생성
- 조직의 엔드포인트 서명키(또는 그 파생물)로 키에 서명
- TCB 의 트러스트 앵커에 키 저장

- 해당 엔드포인트의 고유 내부 식별자 생성(또는 이용)
- TCB 의 트러스트 앵커에 고유한 식별자 저장
- 고유한 식별자, 키, 서명을 IoT 서비스 백엔드 인증 시스템에 저장

엔드포인트 플랫폼의 개인화는 네트워크 ID 의 개인화와 분리돼 있음에 유의하기 바랍니다. 네트워크 인증에 UICC 를 이용하면 여러 모로 유익하며, 가능하다면 UICC 를 트러스트 앵커로 이용해도 됩니다. 그러나 네트워크 트러스트 앵커가 네트워의 인증에만 사용 가능하다면 그 애플리케이션 트러스트 앵커의 개인화는 반드시 별도로 실시해야 합니다. 애플리케이션 트러스트 앵커는 크립토그래픽적으로 고유해야 합니다. 그래야 애플리케이션 플랫폼이 엔드포인트 애플리케이션 실행 전에 검증되기 때문입니다.

UICC 는 네트워크 운영업자나 기타 발급 주체와의 계약을 통해 인도 전 프로비저닝 돼 애플리케이션 중심의 트러스트 앵커 역할을 하기도 합니다. 엔드포인트 개발자는 가까운 장래에 eUICC 기술이 IoT 제품과 서비스에 사용되기 적합한지 평가해야 합니다. 이들 기술로 애플리케이션 중심의 트러스트 앵커와 비슷한 방법으로 크립토그래픽 비밀을 인필드(in-the-field) 프로비저닝할 수 있을 것입니다. 모바일 기술이 개인화와 프로비저닝 프로세스의 리더이므로 eUICC 를 트러스트 앵커로 이용하는 것에 상당한 이점이 있을 수도 있습니다.

또한 이들 기술에는 원격 프로비저닝 능력이 포함돼 있고 애플리케이션과 eUICC 트러스트 앵커 간 보안 통신을 위한 보안 채널도 들어 있습니다. 이들 기능은 인필드(in-field) 개인화가 가능해 엔드포인트 별로 개인화와 프로비저닝의 전반적 비용이 줄어드는 결과로 이어질 것입니다.

IoT 서비스 생태계 내 UICC 카드 활용에 관한 짤막한 튜토리얼이 부록 B 에 제시돼 있습니다.

어려운 부분은 엔드포인트 ID 와 서명 프로세스의 관리입니다. 각 ID 는 반드시 탬퍼링이 불가능한 시스템에 상응하는 고유 식별자와 함께 목록화해 저장해야 합니다. 이 프로세스는 대개 PCB/A 시설에서 진행되지만, ID 데이터를 안전하게 이송하려면 그 시설과 회사가 연결돼 있어야 합니다.

이 솔루션을 확산하는 것은 크립토그래픽 개인화에 익숙한 시설에게는 간단 명료할 수도 있습니다. 그 외 제조 시설은 이것을 위한 시설이 없을 수도 있습니다. 모바일 업계는 UICC 와 같은 임베디드 기술의 제조와 이행을 제어하는 능력이 있어 그 동안 이 방식에서 성공을 거둘 수 있었습니다. 모바일 업계가 이 프로세스에서 꽤 오랫동안 리더 역할을 하긴 했지만, IoT 애플리케이션 엔드포인트 개인화와 프로비저닝 프로세스는 여전히 걸음마 단계입니다.

엔드포인트의 ID 를 게이트웨이나 업링크가 관리해야 하는지 또는 할 수 있는지 판단할 준비를 해야 합니다. IoT 제품이나 서비스의 아키텍처를 평가하면 ID 관리의 속성이 개인화 프로세스에 영향을 미칠지 어떨지 어느 정도 판단할 수 있습니다. 트러스트는 게이트웨이에 분산될 수도 있지만, 조직은 통신과 인증 시스템의 전체적 보안을 해치지 않으면서 트러스트를 적절히 위임할 수 있을지 판단해야 합니다.

개인화에 수반하는 비용은 대체로 다음과 같습니다.

- 칩 제조사에서 개인화 공정을 구현하는 비용
- 제조사와 IoT 서비스 업체 모두에서 고유한 개인화 값을 조정하거나 전달하는 비용
- 개인화된 ID 를 구현하고 관리하는 비용

6.6.1 위험

조직이 엔드포인트 장치를 개인화하지 않을 경우, 엔드포인트를 서로 구별하지 못할 위험에 놓이게 됩니다. 모든 키가 엔드포인트 시스템 전역에서 동일하다면 일련번호가 고유하든 아니든 중요하지 않습니다. 어느 엔드포인트에서든 키가 추출되기만 하면 공격자는 어느 엔드포인트라도 사칭할 수 있기 때문입니다.

개인화를 적용하면 공격자가 복제 또는 사칭을 원하는 엔드포인트마다 크립토그래픽 비밀을 추출해야 하므로 위와 같은 사태를 막을 수 있습니다. 이 프로세스의 비용은 대개 매우 높기 때문에 트러스트 앵커를 이용한 개인화는 복제와 사칭을 막는 방법 중에서 가장 강력합니다.

6.7 최소한 타당성 있는 실행 플랫폼 (애플리케이션 롤백)

MVeP(Minimal Viable execution Platform)란 트러스트 앵커와 통신하는 믿을 수 있는 실행 환경을 만들기 위해 실시해야 하는 최소한의 일을 말합니다. 이것은 일반적으로 다음을 뜻합니다.

- 내부 클럭 또는 오실레이터의 구성
- 핵심 주변기기(메모리, 저장장치)의 구성
- 각종 하드웨어 브리지 또는 주변장치 구동
- CPU 에서 실행될 다음 코드 묶음 인증
- 다음 단계 코드의 실행
- 애플리케이션 이미지 롤백의 관리

이 MVeP 가 정의되면 최소 부트로더가 트러스트 앵커를 이용해 더 건강한 부트로더를 검증하거나 외부 애플리케이션 검증 후 부트로더의 나머지를 시행할 수 있습니다. 그러면 최소한의 수고로 애플리케이션 플랫폼을 정의할 후속 코드 체인을 인증하는 일관성 있는 환경이 실현됩니다.

또한 MVeP 모델을 이용할 경우 내부 NVRAM 이나 EEPROM 의 양이 적은 프로세서도 내부 또는 외부의 트러스트 앵커를 이용해 신뢰할 수 있는 아키텍처를 부트스트랩할 수 있다는 장점도 있습니다.

마지막으로, MVeP 는 특정 플랫폼의 안정적 버전으로 롤백할 때에도 중요합니다. 애플리케이션 펌웨어 이미지의 무결성을 검증하고 실행 환경을 구성하는 데 필요한 최소한의 기능을 지닌 MVeP 를 정의할 수 있다면 그 기능은 애플리케이션의 핵심 기능과 분리할 수 있습니다. 따라서, 펌웨어 업데이트가 어떤 이유로든 실패하더라도 MVeP 를 이용해 백엔드 네트워크와 다시 연결하고 다른 펌웨어 이미지(같은 이미지 또는 이전 이미지)를 다운로드 할 수 있습니다. 또한 NVRA 칩이 손상된 엔드포인트도 백엔드 서비스와 계속 통신하며 진단 정보를 제출할 수 있습니다.

6.7.1 위험

MVeP 를 정의하는 것은 사소한 문제처럼 보일 수도 있지만 그로 인해 전체 아키텍처가 부팅 프로세스의 각 단계를 크립토그래픽 방식으로 검증할 수 있게 됩니다. 이것은 엔드포인트가 네트워크와 피어에게 자신을 인증하는 능력을 갖추는 데 매우 중요합니다. MVeP 의 아키텍처가 허술하면 부팅 과정에서 보안의 허점이 발생해 공격자에게 익스플로잇의 빌미를 주고 보안 아키텍처가 무효가 될 수도 있습니다.

6.8 각 엔드포인트의 고유한 프로비저닝

개인화가 제조 시점에서 각 장치의 고유성을 보장한다면 프로비저닝은 고유한 장치가 활성화되고 업데이트뙥 특정 고객 ID 와 연동되게 합니다. 프로비저닝 과정은 제조된 장치와 구매됐거나 IoT 환경에 배포된 장치를 구별할 때 유용합니다. 이를 통해 IoT 서비스 업체가 누릴 수 있는 장점은 다음과 같습니다.

- 활성 상태인 장치와 비활성 장치의 구별
- 엔드포인트를 네트워크 또는 특정 고객과 연동된 리소스와 연계

- 고객의 니즈에 따라 엔드포인트 커스터마이징
- 특정 고객 또는 엔드포인트의 침해 여부를 더욱 쉽게 판단

프로비저닝 프로세스는 제조 중에 일어나지 않지만 제조에 도입된 개인화 프로세스에 의존합니다. 프로비저닝은 대개 활성화 프로세스를 시작하는 고객을 기준으로 현장에서 일어납니다. 그러나 프로비저닝은 이 프로세스의 보안 확보를 위해 개인화 프로세스 중 고유한 보안 토큰을 이용하여 고유한 엔드포인트가 고유한 고객과 연동되게 합니다. 이렇게 하면 공격자가 엔드포인트 세부사항을 추측하여 엔드포인트 장치를 임의로 등록하거나 등록해지 하지 못합니다. 여기에는 대신 개인화 과정에서 생성되고 설정된 각각의 고유한 크립토그래픽 토큰이 필요한데, 이는 연산적으로 실행 불가능합니다.

이렇게 하면 IoT 서비스 업체는 공격자가 엔드포인트 서비스를 고의로 임의 스푸핑하거나 등록할 가능성을 수학적으로 매우 낮은 수준으로 유지할 수 있습니다. 그 결과 IoT 환경은 더 안전하고 안정적이 되며 고객과 장치의 관계에서 신뢰도는 더 높아질 수 있습니다.

6.8.1 위험

프로비저닝 프로세스를 구현하지 않으면 조직과 그 엔드포인트 노드 간에 비동기화가 일어날 수도 있습니다. 그러면 조직이 엔드포인트를 추적하고 어떤 장치가 생태계에서 구동되거나 퇴역하였는지 찾아내기가 어려워집니다. 또한, 어떤 장치가 특정 고객과 연동돼 있는지도 알기 어렵게 돼 현장에서 문제가 있거나 침해 가능성이 있는 장치를 추적하기도 더욱 어려워집니다.

6.9 엔드포인트 비밀번호 관리

사용자 인터페이스가 있는 장치는 비밀번호를 효과적으로 관리할 수 있어야 합니다. 이를 위해서는 몇 가지 항목이 필요합니다

- 무차별 대입공격 대응
- 기본 또는 하드코드 비밀번호 사용 차단
- 비밀번호 모범 사례 의무화
- 로그인 인터페이스에 사용자 자격증명 게시 불허
- 유효하지 않은 비밀번호 입력 시 임계값과 점증적 지연 의무화

사용자는 가장 단순한 형태의 공격, 즉 다른 사용자가 비밀번호를 추측하는 행위로부터 보호 받아야 합니다. 이것은 무차별 대입공격의 가능성을 차단하는 것만으로 대응이 가능합니다.

방법은 비밀번호 입력 시도 간 시간 제한을 늘리는 것입니다. 로그인 시도가 실패할수록 다음 비밀번호 입력 가능 시점까지 걸리는 시간도 늘어나야 합니다. 한 번에 N 회까지 시도할 수 있다고 상한선을 둬야 합니다. 또는 적당한 잠금 기간을 의무화해야 합니다. 올바른 자격증명 입력 시 사용자에게 무차별 대입공격 사실을 알려야 합니다.

IoT 시스템에서는 기본 비밀번호나 하드코드 비밀번호는 쓰지 *말아야* 합니다. 관리자가 시스템에 들어갈 수 있는 "뒷문 비밀번호"도 *없어야* 합니다. 기본 자격증명을 동반하는 특권 계정도 *없어야* 합니다. 이것은 인터넷을 돌며 허술한 보안을 찾아 무단 침입을 시도하는 사용자로부터 사용자 장치를 보호하는 데 기본이 되는 조치입니다.

비밀번호는 현행 정보 보안 모범 사례를 대표하는 최소 품질 요건을 충족해야 합니다. 그러면 무차별 대입공격이 어려워지고 절도로부터 사용자를 보호할 수도 있습니다. 비밀번호 보안에 관한 OWASP 또는 SANS 가이드라인을 참고해 애플리케이션이 최신 모범 사례를 따르게 해야 합니다.

비밀번호가 사용자 화면에 표시되면 안 됩니다. 비밀번호는 항상 눈꽃송이 문자나 기타 상형문사로 숨겨야 합니다.

또한 비밀번호를 입력하는 인터페이스는 모두 무차별 대입공격에 대응하는 기술을 갖춰야 합니다. 이때 비밀번호를 *검증하는* 기술이 그 대응을 맡는 것이 중요합니다. 예컨대, 웹브라우저에 렌더링된 웹페이지에 임베드된 자바스크립트가 무차별 대입공격에 대한 대응을 맡으면 *안 됩니다*. 웹을 잘 아는 공격자라면 누구라도 인터넷을 통해 백엔드 인증 서버와 상대하여 이 통제장치를 우회할 수 있습니다. 이 모델에서 대응 기술은 서버측에 구현해야 합니다. 로컬 핀이나 비밀번호가 애플리케이션의 보안 저장 구역에 임베드되는 모바일 애플리케이션의 경우, 모바일 장치가 인터페이스에서 일어나는 무차별 대입공격에 대응해야 합니다.

또한 유효하지 않은 비밀번호가 입력된 후에는 대응 시스템이 허용된 횟수 간에 필요한 지연 시간을 늘려야 합니다. 또한 유효하지 않은 비밀번호를 입력하는 횟수에 제한이 있어야 합니다. 이 제한 횟수에 도달하면 사용자는 2 요소 인증이나 더 침습적인 모델의 적용을 받아야 합니다. 난이도

이 프로세스는 구현하기가 매우 쉬우며 엔지니어링 팀에게 부담도 거의 없습니다.

6.9.1 위험

이 권고사항을 구현하지 않을 때 지게 되는 위험은 다음과 같습니다.

- 무차별 비밀번호 추정을 통해 도난 당한 장치가 무력화될 위험
- "드라이브 바이" 인터넷 공격이 단순히 하드코드 비밀번호를 이용해 IoT 시스템의 보안을 전복할 위험
- 사용자 인터페이스에 입력되는 실제 비밀번호가 표시될 경우 "쇼울더 서핑"을 통해 사용자가 침해를 당할 위험

6.10 검증된 무작위 번호 생성기 이용

사용자는 TCB 가 진정으로 무작위 번호 생성을 할 수 있는지 판단해야 합니다. 이 기능이 없다면 크립토그래픽 검증 프로세스가 손상을 입어 암호화된 데이터의 추정이 쉬워지고 데이터 무결성이 약화될 수도 있습니다.

이것은 또 고유한 크립토그래픽 키 생성에도 중요합니다. 여러 가지 환경 조건을 감안할 때, 공격자가 환경에 영향을 미쳐 TCB 가 키 생성이나 서명, 크립토그래픽 메시지 패딩 중 예측 가능한 번호를 생성하지 못하게 해야 합니다.

이 프로세스는 TCB 가 FIPS [10], EMVCo [11] 또는 Common Criteria 의 승인을 받은 무작위 번호 생성이 가능한지 확인해 보는 것으로 충분합니다.

6.10.1 위험

강력한 무작위 번호 생성기 없이 크립토그래피를 이용하는 것은 여로 모로 위험합니다. 그 이유는 열거하기 어려울 정도로 많지만, 주목해야 하는 약점을 몇 가지만 소개하자면 다음과 같습니다.

- 크립토그래픽 키 생성이 침해를 당해 약하거나 예측 가능한 키가 생성될 수도 있음
- 1 회용 비밀번호/패드 또는 키가 약하거나 예측 가능할 수도 있음
- 메시지 리플레이 가능성을 없애기 위해 사용되는 메시지 패딩이 침해를 당할 수도 있음

위와 같은 문제는 전체 IoT 시스템 내 크립토그래픽 보안의 전반적 무결성에 심각한 장애를 유발할 수도 있습니다. 이 위험은 엔드포인트 장치뿐만 아니라 전체 네트워크에 영향을 미칩니다.

6.11 크립토그래픽한 방법으로 애플리케이션 이미지에 서명

CPU 의 핵심 EEPROM 밖에 저장된 애플리케이션은 모두 크립토그래픽 방식으로 인증을 받아야 합니다. 방법은 다음 절차를 따르기만 하면 됩니다.

- 애플리케이션 이미지의 버전을 나타내는 메타데이터를 찾는다
- 메타데이터를 포함해 애플리케이션 이미지의 크립토그래픽 해시를 생성한다
- 애플리케이션 메타데이터가 내부 메타데이터와 일치하는지 확인한다
- 해시 값이 트러스트 앵커 내부의 값과 일치하는지 확인한다
- 애플리케이션 서명 키로 서명을 크립토그래픽 방식으로 검증한다
- 애플리케이션 서명 키가 조직 기반에서 서명을 한 것인지 크립토그래픽 방식으로 검증한다

이 프로세스는 휘발성이 가장 높은 활동을 먼저 실시하고 페일 가능성이 가장 작업을 마지막에 실시하도록 구성되었습니다. 이렇게 하면 적은 노력으로 가능성이 높은 위험을 찾아낼 수 있습니다.

이 프로세스는 구현하기가 매우 쉽습니다. 특히 TCB 가 애플리케이션 대신 처리 작업을 담당할 능력이 있을 때 더욱 그렇습니다. 진짜 난제는 *어떤 애플리케이션*이 그 작업을 실행하느냐입니다.

크립토그래픽 방식으로 검증을 받지 않은 애플리케이션은 이 작업을 실시할 수 없습니다. 자기 코드가 공격자에게 무력화되었는지 알 길이 없기 때문입니다. NVRAM 에서 코드를 변경하는 것은 공격자들이 임베디드 시스템, 특히 애플리케이션을 검증하지 않는 임베디드 시스템을 조작할 때 자주 이용하는 방법입니다.

내부 EEPROM 애플리케이션은 외부 저장장치에서 어떤 애플리케이션 이미지를 상대하든 이 절차를 먼저 실시해야 합니다. 이어서, 작업 자체를 실시하거나 내부 EEPROM 에 인코딩된 애플리케이션에게 대신 시험을 실시하도록 요청할 수 있습니다.

6.11.1 위험

엔드포인트 펌웨어(NVRAM)에 저장된 애플리케이션 이미지에 크립토그래픽 방식으로 서명이 없다면, 시스템은 승인 받은 코드와 공격자가 주입한 코드를 구별할 수 없게 됩니다. 그러면 공격자가 실행 코드를 남용해 물리적을 침해 당한 엔드포인트를 조작할 수 있을 뿐만 아니라 경쟁 업체도 엔드포인트에 자기 소프트웨어를 설치할 수 있습니다.

6.12 원격 엔드포인트 관리

엔드포인트라고 모두 다 원격 관리가 필요한 것은 아니지만, 원격 관리가 필요한 엔드포인트라면 반드시 3 자가 관리 자격증명을 남용해 필드의 엔드포인트를 일부 또는 전부 침해하지 못하게 해야 합니다. 적절한 해결책은 엔드포인트의 기능에 따라 달라집니다. 그러나 다음 가이드라인은 적용해야 합니다

- SSH, 개인 키, TLS 개인 키, 비밀번호 등 개인 크립토그래픽 요소는 엔드포인트에서 보안이 확보되지 않은 저장장치에 넣지 않는다
- 가능하다면, 엔드포인트별로 관리 토큰(크립토그래픽 키 또는 비밀번호)을 만든다
- 비밀번호를 사용하는 경우, 그 복잡도와 길이를 모범 사례에 맞게 의무화한다
- 가능하다면, 관리자를 대상으로 2 요소 인증을 의무화한다
- 관리자가 엔드포인트에 원격으로 접속하면 최종 사용자가 알도록 한다
- 가상 사설망(VPN)에는 관리자 접속을 제한하는 것을 고려한다
- 공용 접속이 가능한 애플리케이션이나 API 에 원격 관리 기능을 임베드하지 않는다. 별도로 구분된 통신 채널을 이용한다
- 관리 통신 채널에 기밀유지와 무결성을 의무화한다
- 업계 표준 통신 프로토콜로 통신 프로토콜에 적절한 엔트로피를 실현함으로써 관리 명령의 리플레이 가능성을 낮춘다

6.12.1 위험

원격 관리에 관한 정책을 정의하고 구현하고 의무화하지 못하면 엔드포인트가 원격 침해를 당할 수도 있습니다. 엔드포인트 장치에 대한 슈퍼 사용자 액세스에 적용되는 엄격한 보안 모델이 없다면 공격자가 그 기술을 역설계하거나 엔드포인트에서 보안 키를 추출해 생태계 내 모든 엔드포인트에 액세스하게 될 수도 있습니다. 관리 액세스가 임베디드 시스템에서 공격자의 첫 목표물이 되는 경우가 많은데 이는 구성에 오류가 있거나 기술적으로 취약할 때가 종종 있기 때문입니다.

6.13 로깅과 진단

IoT 서비스 업체가 엔드포인트 장치의 문제를 평가하려면 엔드포인트의 거동을 끊임 없이 평가해 엔드포인트가 승인 받은 거동 내에서 작동하고 있는지 판단해야 합니다. 이를 위해서는 다음 세 가지 전략을 구사해야 합니다

- 이상 탐지
- 엔드포인트 로깅
- 엔드포인트 진단

엔드포인트는 저마다 자기의 거동을 로그하고 이따금 그 로그를 백엔드 서비스에 업로드해야 합니다. 로그는 커넬 로그, 애플리케이션 로그, 기타 메타데이터 등 일반적 활동으로 구성해야 합니다.

진단 정보도 주기적으로 관찰해 일반 로그와 함께 또는 별도로 백엔드 서비스에 전달해야 합니다. 진단 메시지에는 온도, 배터리 수명, 메모리 활용도, 실행 시간, 프로세스 목록(해당할 경우) 등 엔드포인트에 대한 환경 데이터를 최대한 수록해야 합니다. 이 정보는 문제가 있거나 이상이 있는 이벤트와 언제 그리고 어떤 서비스가 관련이 있는지 찾아야 할 때 유용합니다.

네트워크에서 이상의 탐지는 로그나 진단 분석으로 드러나지 않는 문제를 찾을 때 유용합니다. 또한 로그나 진단에서 관찰되는 문제들을 분류하고 문제의 원인으로서 현실 세계에서 제대로 반응하지 못하는 구성요소를 찾을 때에도 유용합니다. 네트워크와 재연결을 계속하는 셀룰러 모듈이나 불량 데이터를 생성하는 센서가 그 예입니다.

종합하면, 이 정보는 현장에서 기술의 결함을 찾아낼 때뿐만 아니라 이상 거동이 보안 이벤트의 징후인지 확인할 때에도 유용합니다.

6.13.1 위험

로깅과 진단을 구현하지 않으면 조직이 핵심 정보를 놓칠 수도 있습니다. 핵심 정보는 생태계의 보안에 영향을 미칠 뿐만 아니라 제품 엔지니어링의 중요한 결함을 진단할 때 유용합니다.

6.14 메모리 보호장치의 의무화

임베디드 시스템에 MMU(Memory Management Unit)나 MPU(Memory Protection Unit) 같은 고급 기술이 없는 마이크로컨트롤러가 삽입되는 경우가 많습니다. 그러나 이 기술들은 *반드시* 다음과 같은 기능이 필요한 플랫폼에 사용해야 합니다.

- 권한이 없는 애플리케이션의 실행
- 신뢰할 수 없는 (3 자) 앱 또는 애플리케이션의 실행
- 권한이 없는 프로세스에서 에뮬레이터 또는 가상 머신의 실행

권한이 없는 애플리케이션의 실행이 필요한 환경은 불량 애플리케이션 또는 침해 당한 애플리케이션으로부터 자신을 보호할 수 있어야 합니다. 그러면 불량 애플리케이션이나 침해 당한 애플리케이션이 TCB, 트러스트 앵커 드라이버, 하드웨어 주변기기 레지스터 등 권한이 있는 리소스를 제공하는 메모리 영역에 들어오지 못합니다.

이 영역에서는 8 비트 마이크로컨트롤러 플랫폼을 더 견고한 플랫폼, 즉 32 비트 마이크로컨트롤러나 풀(full) 프로세서 아키텍처로 전환하는 것이 종종 문제가 되기도 합니다. 그러나 MPU 또는 MMU 로 메모리 보호를 바르게 구현하며 무료로 또는 저렴한 라이센스료로 이용 가능한 임베디드 시스템용 운영체제가 많이 나와 있습니다.

6.14.1 위험

위와 같은 기술들을 적용하지 않으면 불량 애플리케이션이나 침해 당한 애플리케이션이 드라이버나 주변기기 레지스터 같은 핵심 리소스, 심하면 커넬이나 기타 애플리케이션 등 권한 있는 서비스를 변경하더라도 제한할 방법이 없습니다. 메모리 보호장치가 없다면 아무 애플리케이션이나 마이크로컨트롤러 또는 프로세스의 메모리에 제한 없이 액세스할 수 있습니다. 권한이 없는 애플리케이션이 이들 리소스를 이용하는 것은 *반드시* 제한해야 합니다.

6.15 내부 EEPROM 밖에서 부트로딩

부트로더 코드는 대부분 CPU 안에 있는 EEPROM(Electrically Erasable Read-Only Memory) 안에 임베드됩니다. 하지만 항상 그런 것은 아닙니다. CPU 가 외부 소스에서 부트로더를 불러오는지 확인해 봐야 합니다. CPU 에 부트로더 코드를 검증할 수 있는 EEPROM 이 없다면 공격자에게 조작 당해 공격자에게 유리하게 CPU 를 구성할 수도 있습니다.

칩 또는 부트로더를 호스팅하는 메모리의 영역에 적용된 보호 수준에 따라 공격자가 로컬 버스(직렬 주변기기 인터페이스(SPI) 등) 또는 원격 API(펌웨어 OTA 등)를 이용해 임베드된 코드를 조작할 수도 있습니다. 그러면 공격자가 커스텀 코드를 가장 신뢰할 수 있는 실행점, 즉 실행 가능 코드의 첫 단계에 심어 컴퓨팅 플랫폼을 무력화할 수 있게 됩니다. 그런가 하면 공격자가 부트로더 칩을 떼어내고 커스텀 명령어가 들어 있는 새 칩을 끼워 넣는 방법으로 공격을 할 수도 있습니다. 사용자는 외부 코드의 무결성을 검증할 방법이 없어 승인 받은 소프트웨어와 그렇지 않은 소프트웨어를 구별할 수 없게 됩니다.

공격자가 부트로더를 커스터마이징하려면 부트로더를 개발하거나 개발을 외부에 맡겨야 합니다. 이 작업의 난이도는 가용한 자원과 목표 프로세스에 따라 아주 쉬울 수도 있고 지극히 어려울 수도 있습니다.

내부 EEPROM 이나 잠금 가능 NVRAM 이 탑재된 CPU 또는 MCU/MPU 를 이용해 부트로더를 저장하는 방안을 고려해 보기 바랍니다. 그렇게 하면 적어도 아키텍처에서 로드하고 실행한 첫 실행 파일만큼은 검증할 수 있어 장치의 신뢰도를 높일 수 있습니다.

6.15.1 위험

CPU 가 로드한 첫 코드을 대상으로 신뢰 체인을 평가하지 않거나 무결성 검증을 의무화하지 않으면 시스템 침해가 확대될 수도 있습니다. 이 같은 조치는 IoT 엔드포인트 장치, 나아가 생태계의 보안을 위해 매우 중요합니다.

6.16 메모리의 핵심부 잠금

메모리 중 실행 가능 영역에 저장된 핵심 애플리케이션(첫 단계 부트로더, 신뢰 컴퓨팅 기반 등)은 읽기 전용으로 저장해야 합니다. 그러면 장치가 공격자의 개입 없이 유효한 구성으로 부팅이 가능합니다. 이와 같은 안전 장치가 없다면 실행 첫 단계 이후 로드되는 실행 코드는 자기가 유효한 구성이나 상태로 부팅되었다고 믿을 수 없게 됩니다.

물론 여전히 공격자가 메모리의 이 핵심부를 자체 코드로 교체해 시스템을 전복시킬 수는 있지만, 여기서 필요한 해당 소프트웨어의 커스텀 버전을 만드는 일은 복잡하고 까다롭습니다. 그러면 공격 비용이 증가하게 되고 성공에 필요한 기술도 늘어납니다. 또한 개인화와 프로비저닝이 사용된다면, 공격자는 엔드포인트마다 프로세스를 다시 만들고 로컬 시스템의 고유한 크립토그래픽 특성에 맞춰 자기 솔루션을 커스터마이징해야 합니다. 그러면 공격자의 총 비용이 크게 높아지고 타당성은 떨어지게 됩니다.

메모리의 핵심부에 잠금 기능이 있는지 확인하기만 하면 위와 같은 위험에는 쉽게 대처할 수 있습니다. 아니면, 잠금 기능이 있는 EEPROM 기술로 시작해도 됩니다.

잠금장치가 사용된다면 소프트웨어에서 설정되면 안 됩니다. 소프트웨어가 지정하는 잠금장치는 소프트웨어가 각 기능을 실행해 잠금장치를 활성화한 후에만 구동합니다. 이때 몇 밀리초 정도 잠기지 않은 상태가 나타나는데 이것을 공격자가 이용할 수도 있습니다. 그러므로 가능하다면 항상 퓨즈나 로크 비트 같은 하드웨어 잠금장치를 이용해야 합니다.

6.16.1 위험

잠금장치나 읽기 전용 상태가 없다면 메모리의 핵심부는 공격자가 쉽게 변경할 수 있습니다. 그러면 공격자가 추가 조치 없이 엔드포인트 플랫폼 전체를 침해해 시스템에서 사용되는 후속 보안 장치를 모두 전복할 수 있는 권한을 갖게 될 수도 있습니다.

6.17 안전하지 않은 부트로더

부트로더의 임무에는 주요 애플리케이션의 실행에 맞춰 CPU 를 구성하는 것 외에도 실행 제어장치를 불러와 애플리케이션에게 전달하는 것도 있습니다. 부트로더는 이를 위해 대개 메인 애플리케이션을 찾아 CPU 메모리로 불러옵니다. 문제는 기본 부트로더가 특정 형식의 시스템에 사용될 때 발생합니다.

마이크로컨트롤러 벤더가 이용하는 부트로더 중 다수가 예컨대 외부 펌웨어를 CPU 메모리로 불러와 실행하거나 직렬 인터페이스를 통해 펌웨어 업데이트하는 것을 허용하고 있습니다. 그런가 하면 애플리케이션 이미지가 들어 있는 위치를 사용자에게 알려줘 사용자가 원하는 애플리케이션이면 아무거나 실행할 수 있게 하는 부트로더도 있습니다.

이런 기능은 데스크톱이나 노트북, 서버 등의 환경에서는 무방하나 임베디드 시스템에서는 허용되지 않습니다. 부트로더가 미확인의 신뢰할 수 없는 애플리케이션을 로드해 실행할 경우, 실행된 애플리케이션의 안정성이나 보안을 보장할 길이 없어 임베드된 장치의 상태가 의문으로 남기 때문입니다.

그러므로 이 문제를 해결하기 위해서는 다음 항목을 따라야 합니다.

- 부트로더는 실행될 애플리케이션 이미지를 크립토그래픽 방식으로 검증할 수 있어야 한다.
- 기본/표준 부트로더가 대체 이미지 또는 펌웨어 플래싱을 허용한다면 사용하면 *안* 된다
- 부트로더가 임의의 저장 위치에서 로드된 애플리케이션 이미지를 허용해서는 안 된다

- 첫 단계 부트로더에서 실행 가능한 미지는 EEPROM 에 잠기며 보안 절차를 통해서만 업데이트 되어야 한다

또한, 부트로더의 설계는 3 자 보안 애널리스트에게 검토를 받아야 합니다. 소프트웨어 버그의 조작으로 부트로더가 침해를 당하면 커스텀 코드가 실행되거나 무결성 검증 확인이 무력화될 수도 있습니다. 이것은 *탈옥*으로 이어지기도 하는데, 회사에게는 좋지 않을 수도 있습니다. 시스템에 사용되는 부트로더는 예외 없이 보안 위험으로 이어질 수도 있는 소프트웨어 프로그래밍 오류 여부를 철저히 검사해야 합니다.

6.17.1 위험

보안이 결여된 부트로더는 아키텍처가 허술한 부트로딩 프로세스만큼이나 큰 손상을 유발할 수도 있습니다. 부트로더에 대한 보안 확보는 IoT 엔드포인트의 무결성 확보를 위해 매우 중요한 조치입니다.

6.18 완전 순방향 기밀성(PFS)

PFS(Perfect Forward Secrecy)는 두 엔드포인트 간 통신 셋업 시 교환되는 크립토그래픽 키의 공개를 담당합니다. 일반적으로 이들 엔드포인트에는 각자의 신원을 인증할 때 사용되는 인증서가 위치합니다. 인증 단계가 완료되면, 대칭 키가 생성되고 비대칭 암호화를 통해 합의돼 키 협상을 보호합니다. 이 키가 생성돼 합의에 이르면, 두 엔터티 간 나머지 세션의 보안은 모두 이 키를 이용하게 됩니다. 이것은 비대칭 크립토그래피에서 발생하는 컴퓨팅 비용을 낮추는 역할을 합니다. 대칭 크립토그래피는 컴퓨팅 측면에서 저렴합니다. 즉 임베디드 기술이나 저출력 기술에서 더 빠르고 전력 소모가 절 집약적입니다.

그러나 한 가지 단점이 있습니다. 많이 쓰이는 이 키 합의 모델은 비대칭 키가 *항상 비밀을 유지한다*고 가정하는데, 그렇지 않을 수도 있습니다. 앞으로 누군가는 충분한 자금 지원 하에 어떤 공용 비대칭 키에 대해서도 개인 키를 컴퓨팅할 수 있을지도 모릅니다. 공격자가 목표 엔터티와 그 피어 간의 *통신 세션을 모두* 저장한다면, 그 엔터티는 미래 어느 시점에 개인 키를 만들어 *과거의 통신 메시지 전부*를 해독할 수 있을 것입니다.

또 서버의 크립토그래픽 키가 무명의 3 자 또는 내부자에게 침해를 당할 수도 있습니다. 그런 일이 실제로 벌어지면, 도난 당한 비대칭 키로 보호를 받던 통신 메시지를 저장하고 있던 사람은 누구나 그 메시지를 해독할 수 있게 됩니다.

이 문제의 한 가지 해결 방법은 키 협상 과정에서 단기 비대칭 키쌍을 만드는 것입니다. 이 단기 키쌍의 공용 키만 통신 링크의 각 측으로 전달되므로 대칭 키의 운반에 쓸 수 있습니다.

이 단기 키에는 엔트로피가 충분해야 하며, 키도 적당한 기간 안에 연산 고갈 공격의 가능성을 해소할 만큼 커야 합니다. 그러면 키 협상 프로세스가 지속 가능해지고 앞으로 공격을 받을 가능성도 낮아집니다.

또 이 방법으로 피어가 항구적인 비대칭 키를 기밀유지와 무결성이 아닌 인증 목적으로만 사용되게 할 수도 있습니다. 이 비대칭 키가 도난 당하거나 외부에 노출되더라도 인증 프로스만 영향을 받고 통신 채널의 기밀유지와 무결성은 영향을 받지 않습니다.

이 프로세스의 공격에 대한 회복력을 높이려면 인증에 사용되는 비대칭 키에 대해 엔드포인트가 키의 침해 여부를 검증하는 보안 취소 프로세스를 적용해야 합니다. 침해가 발생했다는 통보를 받으면 엔드포인트는 그 키에게 더 이상 인증을 맡기지 말아야 합니다.

6.18.1 위험

PFS 를 구현하지 않았을 때 공격자가 통신 채널의 보안 확보에 사용되는 개인 키를 입수하면 네트워크 통신 전체가 공격자에게 노출될 수도 있습니다. 미래의 어느 시점에 공격자가 개인 키를 손에 넣는다면 공격자가 과거에 포착한 통신 모두가 해독됩니다. 이것은 심각한 결과를 낳습니다.

6.19 엔드포인트 통신 보안

본 가이드의 다른 권고사항과 위험에서 이미 얘기한 사항이지만, 엔드포인트 IoT 에게 가장 큰 위협은 엔드포인트 통신 보안입니다. 공격자가 통신 채널을 조작하는 것이 엔드포인트를 침해하는 방법 중에서 가장 간단합니다.

따라서 엔드포인트 설계자는 다음과 같은 관점에서 통신 보안을 구현해야 합니다.

- 네트워크 피어의 인증
- 데이터의 기밀유지
- 메시지의 무결성

다른 조직에서 설계한 엔드포인트와 상대하기 위해 평문 메시지를 주고 받을 수도 있겠지만, 명령어나 사용자 프라이버시 데이터, 중요한 시스템 메시지가 들어 있는 데이터는 *어느* 채널을

통하든 *반드시* 보안 대책을 강구해야 합니다. 첫 단계는 피어 장치를 인증해 그것의 진위를 가려내는 것입니다. 이것은 피어가 시스템 서비스를 표시하고 있을 때 특히 중요합니다.

다음으로 데이터 기밀유지를 확보해 3 자가 통신 채널을 통해 전송되는 핵심 데이터를 읽지 못하게 해야 합니다.

마지막으로 메시지의 무결성을 확보해 비밀 메시지가 공격자에게 탬퍼링 당하지 않게 해야 합니다.

이 세 가지 속성이 결합하면 엔지니어링을 거의 바꾸지 않아도 몇년 동안 살아남을 수 있는 통신 모델이 탄생합니다.

여기에 다음을 비롯해 기존에 잘 분석된 보안 프로토콜을 접목하면 그 과정은 더욱 단순해집니다.

- 최근 승인된 TLS 표준
- 최근 승인된 DTLS 표준
- 인증 및 키 교환용 SSH2
- 키 생성 및 교환용 GBA
- 승인용 OAuth2
- BEST, 저산출 머신형 통신(MTC) 장치를 위한 배터리 효율 보안[21]

엔지니어링 팀에서는 전술한 요건에 부합하는 패키지라면 어느 것이나 이용해도 되지만, 표준 통신 프로토콜 패키지를 이용하면 현장에서 관찰되는 오류를 줄일 수 있습니다. 이는 정보 보안과 크립토그래피의 전문가들이 표준화된 프로토콜의 개발에 참여하고 있기 때문입니다.

표준화된 LPWA 네트워크 기술인 NB-IoT 와 LTE-M 을 비롯해 3GPP 기반 셀룰러 통신 기술의 보안 속성은 GSMA PRD CLP.14 [4]에서 찾을 수 있습니다.

6.19.1 위험

통신 보안은 의무사항임은 말할 필요도 없지만, 그것이 *왜* 의무사항인지를 두고서는 때로 혼선이 발생하기도 합니다. 통신 보안의 역할이 단순히 공격자가 데이터를 읽지 못하게 하는 데서 그치는 것은 아닙니다. 그 외에 다음과 같은 기능도 합니다.

- 엔드포인트의 사칭 차단

- 핵심 서비스의 사칭 차단
- 남용된 메시지의 탐지
- 소프트웨어 또는 보안 구성의 안전한 변경

통신 보안이 없다면 IoT 제품이나 서비스의 품질, 안정성, 프라이버시가 보장될 수 없습니다.

6.20 엔드포인트 ID 인증

엔드포인트마다 크립토그래픽적으로 고유한 ID(예: 고유 일련번호)를 지닌다면, 장치는 자기가 *진정으로 그 일련번호를 나타낸다는 것*을 증명할 수 있어야 합니다. 이를 위해 TCB 는 TCB 와 IoT 백엔드 서비스가 알고 있는 키로 메시지에 크립토그래픽 방식으로 서명을 해야 합니다. 이것은 GBA 와 같은 기술로 관리할 수 있는 복잡한 과정입니다. 메시지에는 엔드포인트와 비교해 고유한 ID(일련번호, 기타 토큰)와 메타데이터가 들어 있어야 합니다.

TCB 가 서명할 메시지에는 또 백엔드 시스템에서 발급한 질문(challenge)도 들어 있어야 합니다. 그러면 이미 TCB 에서 백엔드로 제출한 이증 메시지를 공격자가 *리플레이*하지 못하게 됩니다. 질문에 *엔트로피*가 충분히 들어 있다면 메시지 리플레이의 여지는 차단됩니다.

엔드포인트의 ID 를 질문하는 방법:

- 고유한 ID 토큰이 들어 있는 엔드포인트에서 요청 접수
- 고유한 질문 생성 후 엔드포인트로 발송
- 엔드포인트에서 서명과 메시지가 들어 있는 질문 회신 접수
- 공유 키를 이용해 서명이 올바른지 검증
- 서명된 메시지에 올바른 ID 토큰과 기타 관련 메타데이터가 들어 있는지 확인
- 검증 받은 서명 인정

질문 처리 방법:

- 백엔드 시스템에 연결
- 백엔드 시스템의 크립토그래픽 ID 접수
- TCB 를 이용해 백엔드 시스템의 ID 를 크립토그래픽 방식으로 인증
- 엔드포인트 ID 와 기타 메타데이터가 들어 있는 메시지를 백엔드로 발송
- 백엔드에서 질문 접수
- 고유한 ID 토큰과 메타데이터, 질문이 들어 있는 메시지를 생성

- 메시지에 서명
- 메시지와 서명을 백엔드로 발송
- 백엔드 시스템이 서명된 메시지를 승인했는지 검증

6.20.1 위험

이 권고사항을 구현하지 않을 경우 엔드포인트가 복제를 당하거나 사칭 공격의 먹잇감이 될 위험이 있습니다. 그렇게 되면 조직의 인프라가 경쟁사와 공격자 모두의 공격에 놓이게 됩니다. 경쟁사는 엔드포인트 ID 인증의 부재를 틈타 동일한 자재명세서에서 더 낮은 원가로 경쟁 플랫폼을 만들 수도 있습니다.

아니면, 인증의 허술함을 이용해 회사의 인프라에 기생하는 하드웨어를 판매할 수도 있습니다. 이런 문제들은 회사에게 수익의 상실과 운영비 증가를 유발할 수도 있습니다. 경쟁사가 무료로 회사의 네트워크 인프라를 이용할 수 있기 때문입니다. 네트워크 대역폭에는 측정 가능한 비용이 따르고, 클라우드 서버와 CPU 활용, 디스크 활용, 기타 리소스 역시 측정 가능한 비용이 발생하므로 이와 같은 기생 회사는 취약한 조직에게 큰 손해를 끼칠 수 있습니다.

7 높은 우선순위 권고사항

우선순위 권고사항이란 엔드포인트 아키텍처가 요구할 때에만 구현해야 하는 권고사항을 말합니다. 가령, 엔드포인트 아키텍처 중에는 훼손에 강한 제품 케이스가 필요 없는 것도 있습니다. 이들 권고사항은 평가를 통해 구현해야 하는 사업적 타당성이 있는지 판단해야 합니다.

7.1 비밀에 내부 메모리 사용

가능하다면, 프로세서는 내부 CPU 메모리를 이용해 트러스트 앵커 안에 들어 있지 않은 핵심 비밀과 크립토그래픽 키를 처리해야 합니다. 그러면 공격자가 메모리 버스를 모니터링하거나 설령 조작할 수 있더라도 핵심 비밀은 얻지 못하며, 비밀의 이용이 실행 중인 애플리케이션에 미치는 효과만 볼 뿐입니다.

이 모델은 크립토그래픽 비밀의 수명 연장을 낳아 공격자가 비밀 입수를 단념하게 만드는 효과가 있습니다. 그러면 공격자는 비밀의 이용 *효과*와 동일한 RAM 속 비트를 조작해야만 합니다. 이를 위해서는 비밀이 내부에서 사용될 때마다 메모리 속 비티를 바꿔야 해 공격의 복잡도가 크게 늘어납니다.

운영체제 중에는 내부 RAM 을 이용해 비밀을 처리하는 모델을 정의하지 않는 것도 있습니다. 그러므로 엔지니어링 팀에서는 이것을 직접 구현해야 할 수도 있습니다. 그 프로세스는 어렵지 않지만 간단하지도 않습니다. 실행 가능 코드는 메모리 루틴이 반드시 내부 프로세서 메모리를 표시하도록 보장된 특정 영역을 이용해야 합니다. 이것은 사용되는 운영체제와 컴파일러 툴체인에 따라 추가 작업이 필요할 수도 있습니다.

7.1.1 위험

대부분의 마이크로프로세서와 일부 CPU 에는 내부 EEPROM 이나 내부 NVRAM 에서 실행되는 코드를 전담하는 작은 내부 SRAM 이 있습니다. 이 SRAM 은 DMA 와 같은 기술을 통해 일부러 노출되지 않는 한 보통 외부 주변기기에는 접속할 수 없습니다. 이 코드로 처리된 크립토그래픽 비밀은 프라이빗하게 유지되는 한 RAM 통신을 가로챌 능력을 지닌 공격자에게 노출될 확률이 매우 낮습니다.

높은 위험은 아니지만, 크립토그래픽 비밀은 공격 가능성이 있으므로 대중이 접속 가능한 버스를 통해서는 전송하면 안 됩니다. 좋은 장비로 RAM 통신을 빠른 속도로 가로챌 수 있는 공격자라면 크립토그래픽 비밀 같은 데이터를 손에 넣을 *수도 있습니다*. 그러나 RAM 을 오가는 메시지를 가로채기 위해서는 크립토그래픽 운영에 정통한 숙련된 역설계자가 필요합니다.

따라서, 이것은 중요한 권고사항이지만 물리적 보안의 확보에 필수불가결하지는 않을 수도 있습니다. 핵심 크립토그래픽 키가 트러스트 앵커 안에 저장돼 있고 세션 키만 애플리케이션에서 처리된다면, 외부 RAM 에서 키가 처리되더라도 즉시 침해로 이어질 가능성은 높지 않습니다. 그러나, 이것은 그립토그래픽 아키텍처가 노출된 키를 핵심 IoT 운영에 중요하지 않은 것, 즉 키 회전, 세션 키 생성, 인증 철회 등으로 한정한다는 가정에 기초하고 있습니다.

7.2 이상 탐지

엔드포인트의 거동을 모델링하는 것은 IoT 보안에서 중요한 부분입니다. 장치와의 성공적 상호작용만이 기록되고 분석된다면 침해 당한 엔드포인트와 정상인 엔드포인트를 구별할 길이 없기 때문입니다. IoT 환경 전체를 대상으로 장치의 거동 흔적을 모두 기록해야 공격자의 거동으로 의심되는 이상 현상을 찾아낼 수 있습니다.

엔드포인트의 이상 거동은 다음과 같습니다.

- 일정하지 않은 재부팅 또는 장치 리셋

- 불규칙하게 통신 네트워크를 떠나거나 합류
- 정상이 아닌 서비스 엔드포인트에 연결하거나 부적절한 시점에 서비스 엔드포인트에 연결
- 정상과 크게 다른 네트워크 트래픽의 양
- 형식이 허술한 메시지가 엔드포인트에서 서버 엔드포인트로 다수 발송

엔드포인트의 정상 거동이 IoT 서비스 업체에게 기록된다면 조직은 이상 거동을 나타내는 거동 패턴을 찾을 수 있을 것입니다. 거동의 기준을 설정하고 이탈 여부를 계속 모니터링하면 생산 환경에서 보안 및 성능 문제를 모두 더 빠르게 진단할 수 있습니다.

거동의 흔적을 기록하면 조직에서 문제의 원인이 되는 기능이나 환경 상태를 더 빠르게 찾아낼 수 있습니다. 그러면 거동 데이터가 수집되지 않을 때보다 엔지니어링 솔루션을 더 빠르게 확보할 수도 있습니다.

7.2.1 위험

이상 탐지를 하지 않는다면 IoT 생태계 안에서 침해 당한 엔드포인트를 탐지하는 데 과도하게 많은 시간이 들 수도 있습니다. 엔드포인트의 이상 거동이 정상 작동 밖에서만 보인다면 관리팀에서 그 엔드포인트를 불신할 이유가 없을지도 모릅니다. 그러나 이상 탐지가 생태계 전역에서 구현된다면 악성 거동을 훨씬 더 빨리 찾아 봉쇄할 수도 있습니다.

7.3 훼손에 강한 제품 케이스 사용

물리적 장치는 칩 수준에서 탬퍼링에 강해야 할 뿐만 아니라 제품 수준에서도 탬퍼링에 강해야 합니다. 케이스는 공격자 또는 호기심 많은 사용자로부터 제품을 보호해야 합니다. 그 방법에는 다음과 같이 몇 가지가 있습니다.

- 케이스가 열리면 NVRAM 을 무효로 만드는 회로
- 빛이 감지되면 보안 퓨즈를 끊어버리는 센서
- 물리적으로 고정된 장치가 자리를 옮겼을 때 경보를 발령하는 센서
- 핵심 회로 부품을 덮고 있는 에폭시
- 내부 또는 탈착식 부품이 장치와 분리되었을 때 울리는 경보

위와 같은 방법을 적용하면 물리적 엔드포인트의 탬퍼링 저항력을 보강할 수 있습니다. 다만 회로 자체의 설계를 개선하는 것이 비용 대비 큰 효과를 낼 수 있기는 합니다. 위 방법들은 아마추어 해커나 초보 공격자의 침해 가능성은 상당히 낮출 수 있지만 좋은 장비를 갖추고 경험이 많은 보안 애널리스트를 막기에는 역부족입니다.

그러므로, 이 방법들은 제품이 소비자의 소유를 벗어났을 때 탬퍼링을 차단하는 조직의 능력을 높여주는 역할을 합니다. 다시 말하면, 소비자가 장치를 집이나 현장에 놓아두었을 때 공격자가 그 장치를 침해하려면 장치에 물리적으로 접근해야 할 뿐만 아니라 탬퍼링을 막는 보안 장치도 무력화해야 합니다. 그러면 장치가 빠르게 침해돼 교체될 여지가 줄어듭니다. 물리적 장치의 보안이 크게 높아지는 것입니다.

그러나 위협 모델이 이 같은 측면을 무시한 채 장비를 갖춘 첨단 공격자를 포함해 일반적인 물리적 공격만을 막는 데 집중한다면 위협에 제대로 대처하기 어렵습니다. 그러면 이 탬퍼링 대응 장치는 공격자를 늦출 수는 있어도 시간과 돈이 있는 공격자는 멈추지 못합니다.

그러므로 비용 대비 효과와 해당 장치의 위협 모델 간에 균형을 찾아야 합니다.
자동현급지급기가 좋은 예입니다. ATM 보안을 위해서는 케이스에 탬퍼링 차단 기능이 있어야 합니다. 공격자가 케이스를 실제로 열어, 예컨대 마그네틱 띠 데이터를 편취하고 액세스 번호를 기록할 수 있기 때문입니다. 그러나 노련한 공격자들은 이미 기존 ATM 위에서 적응하는 로컬식 구성요소, 즉 스키머(skimmer)를 고안했습니다. 그러므로 물리적인 탬퍼링 차단 장치는 원하는 결과 중 일부만 실현할 수 있습니다. 애플리케이션과 하드웨어 설계는 물리적 공격의 차단을 위해 더 노력이 필요합니다.

엔지니어와 기업 수뇌부는 특정 제품 또는 서비스의 위협 모델을 평가하고 공격의 위험과 장치에 구현되는 탬퍼링 차단 대책 간에 균형점을 찾아야 합니다. 탬퍼링 차단장치는 프로세스와 엔지니어링, 사용되는 소재에 따라 종류마다 비용이 발생합니다. 그러나 노력이 원하는 수준의 보안으로 이어지지 않을 수도 있습니다.

예를 들면 칩을 에폭시로 코팅했을 때가 그렇습니다. 이 프로세스는 나름대로 가치가 있지만 공격자가 다음과 같이 손쉬운 두 가지 조치로써 에폭시를 우회할 수 있습니다.

- 에폭시로 싸인 부품에서 나오는 회로를 태핑한다
- 에폭시를 제거한다

에폭시로 감싸면 부품을 보이지 않게 숨길 수는 있지만 에폭시로 코팅된 칩에서 나와 회로를 이동하는 전자까지 막지는 못합니다. 따라서 핵심 비밀이 하드웨어 버스를 통해 전송된다면 에폭시만으로 공격자가 이 데이터를 가로채는 것을 막을 수는 없습니다.

더구나 에폭시 자체도 간단히 떼어낼 수 있습니다. 최근 몇 년 사이 집에서 가정용 화학제품과 자체 공정을 이용해 에폭시를 제거하는 방법까지 등장했습니다. 이 공정은 부식성이고 자칫 위험하기도 하지만 노련한 리버스 엔지니어들이 만든 방법은 탄탄하며 제대로 환기가 되는 실험실이나 사무실을 갖춘 사람이라면 누구나 실행에 옮길 수 있습니다.

따라서 위험 평가를 실시해 탬퍼링 대응 기술의 편익을 침해 난이도를 기준으로 평가해야 합니다. 무작위로 어떤 장치를 쉽게 조작하거나 남용하기를 원하는 공격자로부터 각 장치를 보호하고자 한다면 탬퍼링 차단 대책을 도입해야 합니다. 공격자가 하드웨어 버스를 오가는 메시지를 가로채지 못하게 해야 하는 이유가 있다면 탬퍼링 차단 외에 애플리케이션과 운영체제를 대상으로 더 견고한 보안 아키텍처를 고려해야 합니다.

7.3.1 위험

앞 절에서 언급한 바와 같이 탬퍼링 차단 조치를 도입하지 않았을 때 나타나는 위험은 장치의 요건에 따라 크게 달라집니다. 실물 장치가 열리거나 부서졌거나 변경됐을 때 사용자에게 경보를 울리는 것이 목적이라면 탬퍼링 차단 조치가 중요합니다. 아마추어, 프로를 불문하고 보안 연구자나 공격자의 분석으로부터 장치를 보호해야 한다면 아키텍처 보안이 위험을 막는 올바른 해결책일 것입니다.

어느 경우든, 탬퍼링 차단 대책을 마련하지 않으면 사용자가 실물 장치의 탬퍼링 여러부를 판단할 방법이 없습니다. 튼튼하고 보강된 하드웨어와 애플리케이션 보안 아키텍처가 완비된 애플리케이션에게는 별 일 아닐지 몰라도 의료기기, 텔레매틱스 시스템, 가정 보안, 자동화 시스템 등 사용자에게 중요한 서비스를 제공하는 제품에게는 *매우* 중요한 문제입니다.

7.4 트러스트 앵커를 상대로 기밀유지와 무결성의 의무화

트러스트 앵커와 주고 받는 통신에 대해서는 예외 없이 인증을 실시하고 기밀유지와 무결성을 의무화해야 합니다. 다만 트러스트 앵커가 프로세스의 코어 안에 있을 때에 한해 예외로 할 수 있습니다. UICC 같은 외부 앵커는 받고 보내는 메시지를 신뢰할 수 있을 때에만 신뢰해야 합니다.

이를 위해서는 인증과 암호화 기능이 있는 트러스트 앵커를 골라 질문에 대한 답이 들어 있는 메시지가 안전하게, 가능하다면 검증 가능한 무결성과 함께 발송되는지 검증해야 합니다.

보안 채널로 관리 가능한 UICC 는 기밀유지와 무결성 확보 능력이 있습니다. IoT 서비스 업체는 네트워크 사업자와 UICC 보안 채널 기술을 애플리케이션 보안의 지원에 사용해도 되는지 협의해야 합니다. 장래에는 eUICC 도 애플리케이션 보안이 가능할 것입니다. 그러면 보안 채널을 이용해 부트로더 단계부터 네트워크 인증 단계까지 엔드포인트 애플리케이션의 보안을 활성화할 수 있을지도 모릅니다.

쉬운 일 같아 보이겠지만 이 과정에는 중요한 요소가 숨어 있습니다. 우선 통신 레이어의 각 측면을 테스트해야 합니다. 여러 트러스트 앵커에서 나오는 메시지 중에는 기밀이 아니거나 무결성이 훼손된 것이 있을 수도 있습니다. 예컨대, 작업이 성공하거나 실패했다는 메시지가 사소해 보일지도 모르지만, 공격자가 맞춤형 응답을 보내 애플리케이션을 속일 수도 있으므로 보호해야 합니다.

트러스트 앵커 중에 통신 채널에서 무결성을 확보할 능력이 없는 것이 있을 수도 있습니다. 무결성은 상대적으로 더 좋고 메시지가 탬퍼링 당하지 않았음을 확인하는 수단으로서 도입해야 합니다. 그러나 이것은 호스트 프로세서와 트러스트 앵커 간에 신뢰가 바탕이 되어야 하는데, 그것이 애플리케이션에게는 합리적이지 않을 수도 있습니다.

임베디드 시스템은 어느 것이나 장비를 갖춘 물리적 공격자에게 침해를 받을 가능성을 안고 있기 때문에 단순히 로컬 버스 통신을 위해 두 프로세스에 모두 신뢰 기반을 요구하는 것은 오버킬(overkill)이 될 수도 있습니다. 그러나, 물리적 보안이 중요한 애플리케이션에서는 무결성을 구현해야 합니다.

7.4.1 위험

기밀유지와 무결성을 구현하지 않았을 때 나타나는 위험은 흥미롭습니다. 시스템이 완전히 침해 당할 위험부터 사소한 정보 수집까지 다양하기 때문입니다. 이것은 일부 메시지가 게임의 대상이 *될 수 있기* 때문입니다. 예컨대, TCB 에서 트러스트 앵커에게 메시지의 무결성 검증을 요청하자면 하드웨어 버스를 통해 트러스트 앵커에게 메시지를 전달해야 합니다.

만일 트러스트 앵커가 CPU 내부에 있다면 공격자가 정교하고 값비싼 장비 없이 이 메시를 변경할 가능성은 낮습니다. 그러나, 트러스트 앵커가 회로 기반의 별도 칩이라면 공격자가 회로를 스플리한 후 자기 하드웨어를 넣어 메시지를 변경할 여지가 있습니다. 예컨대 트러스트

앵커가 그 메시지를 받고 무결성 없이 "예, 유효한 메시지입니다"라고 응답해 버린다면 TCB 는 그 메시지가 버스에 침입한 공격자에게 조작되었는지 검증할 수 없게 됩니다.

또한 응답 *이* 무결성 검증 완료라고 해도 버스에 접근할 수 있는 공격자가 회로를 침범해 TCB 의 메시지 요청을 거둬들이고 자체 신뢰 메시지를 트러스트 앵커에 보내 실제 트러스트 앵커의 응답이 TCB 에게 전달되게 할 수도 있습니다. 하드웨어 통신 버스의 보안이 완벽하지 않으면 이와 같은 공격도 가능하며 이 작업을 실시하는 트러스트 앵커의 능력은 무력화됩니다.

그러나, CPU 와 트러스트 앵커가 모두 내부 트러스트 앵커를 갖기를 기대하는 것은 모순입니다. 부팅 가능한 CPU 가 공격자에게 침해를 당했는데 그 CPU 가 자기 EEPROM 으로 트러스트 앵커의 무결성을 검증해야 한다면 그 CPU 는 자신을 어떻게 신뢰할 수 있을까요? 이것은 어려운 문제지만, 해결책이 없는 것은 아닙니다.

그 중 한 가지는 CPU 의 ROM 에 공용 키를 입력하는 것입니다. 이 키로 트러스트 앵커에서 보낸 메시지의 무결성을 검증할 수 있습니다. 임의의 메시지(검증 대상)가 하드웨어 버스를 통해 트러스트 앵커로 전송되면, 트러스트 앵커는 *원 메시지가* 회신의 일부로 들어 있는 메시지에 서명을 해서 응답할 수 있습니다. 이것은 그 메시지가 정말로 해당 트러스트 앵커에서 왔고 처리 중인 메시지가 실제로 처리되어야 하는 그 메시지임을 확인하는 것입니다. 유일한 문제라면 메시지 패딩에 사용된 난스(nonce)가 해당 크립토그래픽 메시지의 *리플레이*를 차단하게 하는 것입니다.

이상을 기억한다면 크립토그래피가 크립토그래피 안에 있는 미묘한 문제뿐만 아니라 크립토그래픽 통신을 지원하는 알고리즘 안에 있는 문제 때문에도 페일(fail)한다는 것을 쉽게 알 수 있습니다. 기밀유지와 무결성의 (올바른) 구현이 그토록 중요한 이유가 바로 여기에 있습니다.

7.5 무선 애플리케이션 업데이트

원격으로 엔드포인트의 *애플리케이션* 이미지를 업데이트하는 것은 단순하고 간단명료한 작업입니다. 복잡한 요소는 실제로는 보안 결함을 상대하지 않게 솔루션을 오버엔지니어링하는 데서 옵니다. 항구적 저장장치라는 관점에서 볼 때 엔지니어링 프로세스는 매우 간단합니다.

- 액티브 애플리케이션 이미지에 위치를 지정
- 백업 애플리케이션 이미지(있다면)에 위치를 지정

- 긴급 애플리케이션 이미지에 위치를 지정
- 백업 애플리케이션 이미지 공간이 존재할 경우 이 공간을 액티브 이미지로 업데이트
- TCB 에 저장된 서명으로 액티브 이미지를 크립토그래픽 방식으로 검증
    - 이로써 저장 매체가 오염되지 않았으며 공격자가 쓰기 과정에서 비트를 수정하지 않았음이 확인됨
- 새 이미지(전부 또는 델타로)와 그 메타데이터, 신호를 다운로드
- 액티브 이미지를 델타로 패치
- TCB 를 이용해 크립토그래픽 서명 검증
- 새 이미지로 재부팅

프로세스가 어느 지점에서든 페일한다면, 시스템은 백업 이미지로 돌아가 애플리케이션이 필요에 맞게 가동하게 하거나 긴금 시스템을 이용해 *원점을 호출*한 후 IoT 생태계에게 고장이 발생했음을 알려야 합니다.

난관은 다음 두 가지 문제에 대처할 수 있는 저장 모델을 만드는 데 있습니다.

- 업데이트 프로세스를 조작하려고 하는 공격자
- 하드웨어 이상

백업 시스템이나 비상 파티션이 없다면 장치는 페일하는 수 밖에 없습니다. 임베디드 시스템은 대개 견고한 사용자 인터페이스가 없기 때문에 이것은 회사와 고객에게 심각한 부담이 될 수도 있습니다. 페일을 잘 하는 것은 고객 신뢰의 문제일 뿐만 아니라 시스템 안정성과도 직결된 문제입니다.

공격자 중에는 업데이트 프로세스를 오염시켜 시스템을 지속적으로 취약한 상태로 만들려고 하는 이들도 있습니다. 예컨대 애플리케이션의 액티브에서 익스플로잇 가능한 취약성이 발견되었지만 그 애플리케이션의 최신 버전에 패치가 있는 경우를 가정해 보겠습니다.

이 모델의 장점은 공격자가 네트워크 협상 프로세스를 오염시키더라도 백엔드 시스템이 이 이벤트를 인지할 기회가 있다는 점입니다. 백엔드 시스템에서 노드가 업데이트를 *제외하고* 정상 통신을 하고 있음을 발견한다면, 관리자에게 해당 엔드포인트의 남용 여부를 확인하라는 경보를 발령해야 합니다.

7.5.1 위험

OTA 애플리케이션 업데이트 프로세스의 아키텍처가 허술하면 공격자가 엔드포인트에 실행 가능한 코드를 원격으로 주입할 수도 있습니다. 공격자가 네트워크에서 특권의 위치를 점한다면 한 번에 수천 개 엔드포인트에 영향을 미칠 수도 있습니다. 이 같은 공격의 결과는 단순한 코드 실행부터 서비스의 거부(엔드포인트 가로막기), 엔드포인트 장치의 목적 변경까지 다양하게 나타날 수 있습니다.

7.6 제대로 엔지니어링되지 않았거나 구현되지 않은 상호 인증

통신 환경에서 피어는 프로토콜의 *ID* 셈블런스(semblance)를 통해 서로 소통합니다. 이것은 컨텍스트마다 다른 의미를 지니지만 어떤 환경이든 메시지의 행선지를 나타내는 *주소*가 있습니다. 특정 프로토콜을 구현하는 통신 모듈은 어느 것이든 그것이 *특정 주소*의 소유자임을 표시할 수 있습니다. 어떤 프로토콜의 특정 *구현체*가 설계상 또는 외부의 힘 때문에 어떤 로컬 무선 모듈의 하드웨어 주소를 이용더라도 사용자가 그 모듈의 EEPROM 을 물리적으로 변경하고 하드웨어 주소를 바꿀 수 있다고 명시하는 규칙은 없습니다. 설령 그 구현체가 사용자의 하드웨어 주소 변경을 허용하지 않더라도 조작을 당해 주소를 변경할 여지는 남아 있습니다. 이 같은 기능의 결과는 기본적으로 스푸핑, 즉 다른 컴퓨터의 ID 를 빼앗아 그 컴퓨터로 향하는 메시지를 가로채는 것입니다.

7.6.1 클라이언트 인증

환경은 모두 스푸핑에 취약합니다. 예컨대, 셀룰러 무전기는 사실이든 아니든 자기가 어떤 국제 모바일 가입자 ID 의 소유자라고 신호를 보낼 수 있습니다. 노트북은 이데넷 주소를 바꿔 LAN 에서 다른 컴퓨터를 사칭할 수도 있습니다. 토폴로지가 물리적 공간을 지나든 전파 공간을 지나든 통신 엔드포인트의 *ID* 는 사칭이 가능합니다.

이것을 막는 방법은 인증입니다. 예컨대, 셀룰러 네트워크에서는 누구든 장비만 갖추면 자기가 선택하는 IMSI 를 소유하고 있다고 주장할 수 있습니다. 그러나 셀룰러 서비스 업자는 가입자 ID 모듈(SIM)에 크립토그래픽 키를 인코딩하여 *인증*을 의무화합니다. 이 키는 가입자마다 고유합니다. 셀룰러 장치가 기지국과 통신하며 자기가 특정 IMSI 를 대표한다고 말하면 기지국에서는 SIM 카드에 저장된 고유한 크립토그래픽 키(해당 ID 에 프로비저닝된 것)를 지니고 있는 사람만 풀 수 있는 크립토그래픽 질문을 보냅니다. 공격자가 이 크립토그래픽 질문을 풀지 못하면 기지국에서는 공격자가 문제의 IMSI 를 대표하*지 않음*을 확인하고 그 사용자의 네트워크 사용을 불허할 수 있습니다.

위에서 설명한 모델은 *클라이언트 기반 인증*입니다. 클라이언트(엔드포인트)가 크립토그래픽 방식으로 ID 를 인증할 수만 있다면 서버 시스템(기지국 포함)을 자유롭게 들어오고 나갈 수 있는 모델입니다 그러나 클라이언트가 조작의 위험에 노출되는 정반대의 문제가 있습니다. *서버 인증*이 그것입니다.

7.6.2 서버 인증

3GPP 모델에서는 엔드포인트(3GPP 내 사용자 장비라고 함)만 인증을 받습니다. 엔드포인트는 자기가 연결된 기지국은 인증하지 않습니다. 따라서 어느 기지국이든 셀룰러 업자를 대신해 서비스한다고 주장할 수 있습니다. 셀룰러 기지국을 조작하거나 만들 수 있는 사람이라면 어느 셀룰러 업자든 사칭할 수도 있는 것입니다. 셀룰러 기지국을 주문해 만드는 비용은 현재 1,000 달러도 되지 않지만 거기서 얻을 수 있는 힘은 로컬 영역에서 메시지를 가로채는 데 그칩니다. 가짜 타워가 세워지고 나면, 기지국은 로컬 셀룰러 업자를 사칭해 인근 지역 내에 있는 엔드포인트에서 전화와 문자 메시지, 데이터를 가로챌 수 있습니다.

UMTS, LTE 등 최근에 나온 3GPP 네트워크 프로토콜은 두 엔터티의 상호 인증을 의무화하고 있습니다. 그러면 기지국이 정말로 자기가 주장하는 셀룰러 업자를 대신해 서비스하고 있는 것인지 엔드포인트에서 확인할 수 있습니다. 이제 공격자가 기지국을 사칭하려면 셀룰러 업자의 크립토그래피까지 깨뜨려야 합니다. 그 결과 공격의 복잡함과 난이도, 비용은 크게 높아지게 됩니다.

7.6.3 셀룰러 질문기 또는 가짜 기지국

그러나 이 규칙에는 예외가 있습니다. 셀룰러 질문기가 그 중 하나입니다. 이 장치는 정부 하청업자나 정부, 첩보기관에서 주로 사용하는 것으로 이 안에는 셀룰러 업자가 국가 안보 목적으로 이들 기관에 제공한 크립토그래픽 키가 들어 있습니다. 이들 시스템은 이 키를 이용해 양방향 통신을 수동적으로 가로채거나 능동적으로 특정 목표물을 대상으로 '중간자' 공격을 실시합니다.

그러나 현대의 통신 위협 모델에서는 이 기술이 정부 기관과 첩보 당국의 전유물은 아닙니다. 몇 백 달러 밖에 안 되는 부품들을 조립하면 이런 시스템을 만들 수 있습니다. 그러면 비용 대비 효과가 큰, 셀룰러 통신을 가로채거나 사칭할 수 있는 가짜 기지국이 탄생하는 것입니다.

7.6.4 통신 보안은 게이트 간 보안

셀룰러 질문기는 이 절의 요점과 맥이 닿아 있습니다. 즉 통신 보안은 절대적이지 않다는 것입니다. 통신 보안은 두 엔터티 사이의 통신 채널만 보호할 뿐입니다. 그러나 이들 엔터티는 데이터를 연결된 생태계로 보내고 또 그 생태계에서 내보내는 *문*에 불과합니다.

가령, 어떤 SIM 카드를 프로비저닝하여 유정 모니터링 장치 같은 산업용 제어 시스템에 쓰도록 만들었다고 하겠습니다. SIM 카드는 설계상 탈착이 가능한 부품입니다. 유정 모니터링 장치에 접근할 수 있는 사람이라면 누구나 그 SIM 카드를 빼서 노트북에 끼울 수 있습니다. 그 노트북에 그 유정 장치의 기능을 모사하는 소프트웨어가 설치돼 있다면 백엔드 서버가 실제 유정 장치와 노트북을 구별할 길은 없습니다. 하지만 노트북은 SIM 카드 때문에 셀룰러 네트워크에 인증을 받게 됩니다! 셀룰러 네트워크는 노트북이 아니라 SIM 카드를 인증한 것입니다.

7.6.5 상호 인증을 위한 문제해결

IoT 생태계의 각 피어는 그 생태계에 참여하는 다른 피어를 전부 인증해야 합니다. 이를 위해서는 TCB 를 이용해 적절한 크립토그래픽 아키텍처가 통신 기술을 주도하게 해야 합니다. 키가 공격자에게 쉽게 노출된다면 상호 인증은 불가능합니다. 자세한 사항은 본 문서의 TCB 편을 참고하기 바랍니다.

인증을 마치면 각 피어는 네트워크 안에서 다른 피어에게 보낸 메시지를 암호화하고 서명해야 합니다. 메시지를 받는 피어는 저마다 그것을 실행하기 전에 데이터를 크립토그래픽 방식으로 검증해야 합니다. 통신 프로토콜이라고 모두 다 상호 인증을 하거나 강력한 크립토그래피를 갖고 있는 것은 아니므로, 애플리케이션 엔지니어는 통신 프로토콜에 의지하기보다 기밀유지와 무결성을 의무화하기에 충분한 프로토콜을 설계해야 합니다.

LTE 와 같이 상호 인증을 내장한 더 강력한 프로토콜조차도 셀룰러 통신망 밖에서는 보안을 책임지지 못합니다. 더 상위의 프로토콜 보안만이 셀룰러 업자의 통제 범위 밖에서 인프라의 취약점을 해결할 수 있습니다.

7.6.6 위험

강력한 애플리케이션 보안을 적용하지 않으면 엔드포인트가 통신 레이어의 보안을 믿어야만 한다는 위험이 있습니다. 본 권고사항에서 설명한 바와 같이 네트워크만을 믿고 애플리케이션의 보안 문제를 해결하기는 충분하지 않을 수도 있습니다. MNO 는 믿을 수 있더라도 메시지는 데이터가 IoT 서비스 업체 소유의 서버에 도달하기 전에 MNO 가

소유하거나 통제하지 않는 네트워킹 인프라의 여러 부분을 통해 전파될지도 모릅니다. 따라서, IoT 서비스 업체는 그와 같은 시스템을 통제하는 누군가가 엔드포인트를 오가는 메시지를 가로채거나 변경하거나 만들어낼 위험 앞에 처이게 됩니다.

7.7 프라이버스 관리

IoT 기술의 중요한 측면 가운데 하나는 실제 세상을 디지털 세상과 연결하는 능력입니다. 그로 인해 프라이버시에는 허점이 생기게 됩니다. 사용자의 실제 환경이 사용자가 온라인에서 좋아하고 보는 것과 직접 연결되기 때문입니다. 그 결과 시간이 지나면서 바람직하지 않은 결과가 나타나기도 합니다.

그러므로 IoT 서비스 업체는 소비자의 프라이버시를 고려하여 엔드포인트와 가능하다면 제품 또는 서비스의 웹 인터페이스에 모두 통합되는 프라이버시 관리 인터페이스를 개발해야 합니다.

이 기술로 사용자는 본인 프라이버시의 어떤 속성이 시스템에서 이용되고 있고 서비스의 약관은 무엇인지 알 수 있고 회사나 그 파트너에게 이 정보의 노출을 차단할 수 있어야 합니다. 이 같은 세분화(granularity)와 탈퇴 제도를 통해 사용자는 본인 자신과 그 실제 세상에 관한 정보의 공유를 관리할 권리와 능력을 확보할 수 있습니다.

7.7.1 위험

소비자 프라이버시를 보호하지 않을 때 따르는 위험은 많습니다. 스토킹, 희롱, 프로파일링, 위협 등에서 나오는 문제는 사용자의 데이터를 보호하지 않았을 때 나타나는 현실적이고 실질적인 결과입니다.

7.8 프라이버시 및 고유 엔드포인트 ID

각 엔드포인트는 지문을 통해 디지털 방식으로 식별됩니다. 이 지문은 엔드포인트 별로 고유한 *주소*, *일련번호*, *크립토그래픽 ID* 로 구성됩니다. 그러나 이들 토큰은 어떤 장치를 특정 고객이나 위치, 서비스와 직접 연결할 수도 있습니다. 이것은 바람직하지 않을 때가 많습니다. 예컨대, 스마트폰은 추적이 가능합니다. 802.11 액세스 포인트를 능동적으로 스캐닝할 때 폰에 내장된 Wi-Fi 주소가 사용되었기 때문입니다. 위치를 이동할 때마다 이 주소를 추적하면 됩니다. 그러면 누구나 특정 Wi-Fi 주소를 특정 사용자와 연결지어 세계 어디서나 동선을 들여다볼 수 있습니다. 이것을 막기 위해 스마트폰 소프트웨어 제조사들은 액세스 포인트를

스캐닝할 때 무작위 Wi-Fi 클라이언트 주소를 만들어 이 같은 방식으로는 전화기 추적이 거의 불가능하게 만들었습니다.

IoT 엔드포인트도 비슷하게 BLE(Bluetooth Low Energy) 주소나 802.15.4 주소, Wi-Fi, 셀룰러를 통해 추적이 가능합니다. 심지어 IMSI 를 통해서도 가능합니다. IoT 서비스 업체는 엔드포인트 기술을 개발할 때 가능하다면 무작위 무선 주소를 통해 새 환경에 연결하여 사용자의 프라이버시가 온전하게 보호를 받게 해야 합니다.

이것은 SSH 공용키 등 크립토그래픽 키에게도 해당하는 사항입니다. 사용자는 대개 본인의 공용 키가 대중에게 알려지기를 원하지만, 엔드포인트의 크립토그래픽 공용 키는 특정 엔드포인트의 사용자 ID 를 노출합니다. 이것은 바람직하*지 않*습니다. 대신 사용자가 새로운 환경과 연결될 때 ID 공개 여부를 선택할 수 있어야 합니다.

7.8.1 위험

이 위험에 적절히 대처하지 못하면 모바일 엔드포인트를 지닌 사용자가 본인 장치의 망 진입/진출을 추적 당하게 됩니다. 그러면 법무팀과 입법기관, 보험사들이 현재 분석 중인 프라이버시에 큰 허점이 생깁니다. 프라이버시를 제대로 구현해 추적의 가능성을 낮추지 못하면 새 IoT 서비스 업체는 가까운 장래에 법적 책임을 질 수도 있습니다.

7.9 적절한 수준의 권한으로 애플리케이션 실행

엔드포인트에서 실행되고 있는 애플리케이션은 대개 슈퍼 사용자 권한이 필요 없습니다. 애플리케이션은 장치 드라이버나 네트워크 포트에 접속해야 할 때가 많습니다. 이런 장치나 포트, 기타 객체 중에는 최초 접속 시 슈퍼 사용자 권한이 필요한 것도 있지만 후속 동작을 수행할 때에는 슈퍼 사용자 권한이 없어도 됩니다. 따라서 애플리케이션 시작 시점에만 슈퍼 사용자 권한을 이용해 접속하는 것이 바람직합니다. 그러고 나면 슈퍼 사용자 권한은 버려야 합니다.

슈퍼 사용자 권한을 버리는 것은 흔한 프로세스로서 관련 문서도 많습니다. 또한 SSH, apache2 등과 같은 애플리케이션에서 아주 잘 구현되기도 했습니다. 이 프로세스의 대상은 다음과 같습니다.

- 높아진 권한으로 애플리케이션 시작하기
- 높아진 권한이 필요한 리소스에 모두 액세스하기

- 애플리케이션 실행 시 사용될 사용자 ID (예: UNIX 사용자 ID, 그룹 ID) 식별하기
- 프로세스의 ID 를 대상 사용자/그룹 ID 로 완전히 변경해 슈퍼 사용자 권한을 실행 중인 애플리케이션에서 제거하기

privsep 의 SSH 구현체에서 더 복잡한 모델을 볼 수 있습니다. 여기서는 대상 사용자/그룹 ID 아래에서 메인 애플리케이션을 부트스트랩하는 것만이 목적인 특권 서비스가 실행됩니다. 이렇게 하면 서비스가 종료되더라도 권한을 지닌 리소스의 침해 없이 쉽게 다시 시작할 수 있습니다.

자세한 사항은 다음을 참조하십시오. SSH Privilege Separation: http://www.citi.umich.edu/u/provos/ssh/privsep.html

7.9.1 위험

권한을 높여 애플리케이션을 실행하면 한 애플리케이션이 침해됐을 때 전체 시스템이 침해되는 결과로 이어질 수도 있습니다. 어떤 애플리케이션이 슈퍼 사용자 권한을 부여 받으면 실행 중인 전체 시스템에 제한 없이 액세스할 수 있게 되므로 그 애플리케이션이 침해를 당했을 때 공격자를 봉쇄할 길이 없습니다. 권한을 낮추면 공격자를 봉쇄하기에도 좋고 임베디드 시스템 안에서 권한을 높이지 못하게 하는 효과도 있습니다. 이것이 시스템 전체의 침해와 사소한 성가심을 가르는 편수가 될 수도 있습니다.

7.10 애플리케이션 아키텍처에서 작업의 분리 의무화

엔드포인트를 실행 중인 애플리케이션은 각각의 고유한 프로세스에 대해 서로 다른 사용자 ID 를 지녀야 합니다. 이렇게 되면 한 애플리케이션이 침해를 당해도 같은 엔드포인트의 별도 애플리케이션은 두 번째 공격이 성공하지 않는 한 안전합니다. 공격자에게 요구되는 이 추가 단계가 전체적인 익스플로잇 개발 과정에 큰 장애물로 작용해 엔드포인트에 대한 공격의 비용과 복잡성을 높일 때가 많습니다.

예컨대, 엔드포인트의 상태에 관한 정보를 사용자가 불러올 수 있는 네트워크 서비스라면 같은 프로세스를 통해 TCB 를 조작할 수 있어서는 안 됩니다. 그 기능은 서비스의 목적에 비춰 *범위를 벗어난 것*일 수도 있습니다. 이 두 가지 작업은 서로 다른 애플리케이션에서 처리하고 로컬 운영체제에서 별도의 사용자 ID 아래에서 실행되어야 합니다. 이는 애플리케이션의 임무를 분리하고 한 구성요소가 침해를 당했을 때 남용의 위험을 낮추기 위함입니다.

이것을 제대로 구현하려면 기초가 되는 하드웨어 아키텍처에서 메모리 보호장치가 활성화되고 운영체제에 권한 수준이라는 개념이 있어야 합니다. 권한이 없는 소프트웨어는 드라이버, 구성 파일, 그 외 객체 등 권한이 있는 리소스에 접근하지 못하게 해야 합니다.

서비스는 시스템 호출처럼 제약이 있는 API 를 통해 요청을 한 후 권한 있는 리소스에 액세스해야 해야 합니다. 이는 모든 메시지가 형식을 갖추고 보안 아키텍처의 요건에 부합하게 하기 위함입니다.

권한의 멀티 티어라는 개념은 나온지 반 세기가 넘었습니다. 그러나 임베디드 시스템에서는 사용자가 콘솔에 로그인해 자체 애플리케이션을 실행할 수 없어 이 개념이 자주 간과되기도 합니다. 그래서 서비스 전체가 권한 있는 사용자로 배포되는 일이 잦습니다. 하지만 여기에는 결함이 있습니다.

애플리케이션이나 서비스는 저마다 다른 권한으로 따로 구현해야 합니다. 대부분의 환경에서 이것은 별도의 *사용자 ID* 가 됩니다. 서로 다른 사용자 ID 를 의무화하여 임무를 분리하면 한 서비스가 침해를 당하더라도 같은 시스템의 다른 서비스에서 이용 중인 리소스에는 직접 영향이 없습니다. 다른 서비스와 다른 사용자를 침해하려면 로컬 운영체제에서 따로 익스플로잇을 찾아 권한을 높여야 합니다.

이를 위해서는 계획이 필요하고 권한 분리를 제대로 이용하는 견고한 애플리케이션이 필요합니다.

7.10.1 위험

임무의 분리를 의무화하지 않으면 엔드포인트의 어느 한 서비스가 침해를 당할 경우 장치 전체의 침해로 이어지게 됩니다. 해당 장치에서 실행 중인 서비스나 애플리케이션이 모두 같은 사용자/그룹 ID 를 공유하기 때문입니다. 이 권고사항 *이* 구현되면, 권한이 낮은 서비스가 네트워크를 통해 침해를 당하더라도 전체 시스템의 침해로 즉시 이어지지는 않습니다.

이 권고사항은 구현하기가 간단하기 때문에 IoT 엔드포인트의 보안에는 매우 중요합니다. 네트워크 서비스를 원격으로 침해하려면 막대한 비용이 들 때가 많다는 사실을 기억해야 합니다. 공격자가 시스템 전체를 장악하기 위해서 커널급 익스플로잇이나 두 번째 익스플로잇을 구현해 권한을 높여야만 한다면 공격자에게는 공격을 실행할 만한 시간이나 기술, 장비가 부족해질 수도 있습니다.

이처럼 간단하게 구성을 바꿔 공격의 난이도를 높인다면 장치의 수명 연장에도 큰 도움이 될 것입니다.

또한, 프로세스 모니터링과 그 외 분석을 통해 침해 당한 서비스를 탐지할 수 있으므로 어떤 서비스 침해가 발생해도 서비스 생태계에 경보가 발령됩니다. 이렇게 되면 시스템 전체가 침해를 당하기 전에 관리자가 시스템에 대해 보안 조치를 강구할 수 있습니다. 또한 관리자가 어떤 취약점이 남용되기 전에 취약한 소프트웨어를 진단하고 패치할 수도 있습니다. 결국 회사가 노련한 공격자에 대해 큰 무기를 갖게 되는 것입니다.

7.11 언어 보안 의무화

프로그래밍 언어는 그 목적과 수준에 따라 보안 수준이 저마다 다릅니다. 언어 중에는 원시 메모리에 대한 액세스를 제한하고 메모리의 사용을 제약하는 것도 있습니다. 엔지니어링 팀은 애플리케이션 런타임이나 거기서 나오는 바이너리를 보호할 수 있는 언어를 찾아야 합니다.

컴파일러나 런타임은 가능하면 보안을 철저히 하여 공격자가 취약점을 남용할 여지를 낮춰야 합니다. 잘 정의된 런타임 환경에서는 쉽게 트리거할 수 있는 프로그래밍 결함조차도 완전히 익스플로잇하기가 매우 어렵습니다. 다만 이를 위해서는 보안 강화가 적용돼 애플리케이션이 실행하고 메모리에 액세스하고 운영체제의 보안 강화의 지원을 받는 과정이 보호를 받아야 합니다.

7.11.1 위험

프로그래밍 언어와 그로 인한 애플리케이션을 보강하지 않으면 애플리케이션이 익스플로잇의 순쉬운 목표일이 되고 맙니다. PHP 와 같은 일부 프로그래밍 시스템은 버그로 악명이 높아 전문 엔지니어링 팀에서는 절대로 사용하면 안 됩니다. 그 외 Python 과 같은 시스템은 생산 환경에는 적합하지만 반드시 평가를 받아야 하는 민감한 보안 위험이 있습니다. 이렇듯 보강하지 않을 때의 위험은 치명적인 수준부터 약한 수준까지 천차만별입니다. 엔지니어링 팀에서는 *반드시* 위험 평가와 위협 모델링 프로세스를 적용해 어떤 언어가 생산 환경에 가장 적합한지 충분히 평가해야 합니다.

7.12 수시로 모의침투 실시

수시로 새 엔드포인트가 현장에 배포되고 구성되는 IoT 환경에서는 배포 시점에 실시하는 보안 점검만으로는 충분하지 않습니다. 수시 모의침투 방식을 도입해 취약한 엔드포인트 소프트웨어와 안전하지 않은 구성을 조기에 탐지하기를 권장합니다.

수시 모의침투 전략을 도입하면 확인된 위협을 빠르게 탐지해 조기에 관리할 수 있어 대응 속도를 단축하고 위험 노출 기간을 줄일 수 있습니다.

수시 모의침투 전략이 빈틈 없이 갖춰지면 자산 발굴을 통한 접속 가능한 자산 집단 구축, 자산 평가 및 분석, 드러난 취약성 검증과 탐구, 보안이 결여된 구성의 점검, 대응 조치를 위한 보고와 경보가 계획에 따라 자동으로 진행됩니다.

7.12.1 위험

수시 모의침투 전략을 구현하지 않으면 배포 시점에만 보안 점검이 실시되고 새 엔드포인트와 구성은 평가를 받지 않을 위험이 있습니다. 이는 공격자에게 침해를 받을 때까지 취약한 엔드포인트가 전혀 발견되지 않는 상황으로 이어질 수도 있습니다.

8 중간 우선순위 권고사항

중간 우선순위 권고사항은 엔드포인트 기술의 설계 방식에 따라 연관되는 권고사항을 모아 놓은 것입니다. 예컨대, 운영체제 수준의 보안 강화를 의무화하는 것은 엔드포인트에서 실행 중인 운영체제가 있을 때에만 유효합니다. 엔드포인트가 모놀리식 커널이나 임베디드 애플리케이션이 하나 밖에 없는 임베디드 실시간 운영체제(RTOS)로 구성될 경우 이 권고사항은 적용되지 않을 수도 있습니다. 권고사항이 엔드포인트 설계에 적용될 경우 구현해야 합니다.

8.1 운영체제 수준의 보안 강화 의무화

운영체제에서 실행되는 애플리케이션은 (투명하게 또는 의도적으로) 기초가 되는 운영체제와 커널의 보안 강화요소를 이용하도록 설계해야 합니다. 여기에는 다음과 같은 기술이 포함됩니다.

- ASLR
- 비실행 메모리(Stack, Heap, BSS, ROData 등)
- UDEREF(User-Pointer Dereference Protection)
- 구조 유출(정보 공개) 차단

임베디드 시스템에 사용되는 각 운영체제는 이들 기술을 다양하게 바꾸거나 조합해 제공합니다. 때로는 이름을 다르게 붙이기도 합니다. 운영체제와 커널이 무엇을 제공할 수 있는지 파악해 이들 기술을 통해 애플리케이션의 보안을 높여야 합니다.

어려운 부분은 각 운영체제의 능력을 찾는 것입니다. 예컨대, MMU(Memory Management Unit)가 없는 플랫폼에서 실행되는 애플리케이션은 ASLR 을 하지 못할 수도 있습니다. 그러나 MPU(Memory Protection Unit)가 하나 밖에 없는 환경에서도 UDEREF 에 버금가는 요소를 의무화할 *수 있습니다*. 어떤 기술이 사용되고 있는지, 그 기능은 무엇인지 평가하고 아키텍처와 커널, 운영체제, 애플리케이션 보호장치를 조합해 어느 수준의 보안을 성취할 수 있는지 판단해야 합니다.

8.1.1 위험

이 권고사항을 구현하지 않으면 익스플로잇하기가 대단히 쉬운 애플리케이션 런타임 환경이 될 위험이 있습니다. 위의 강화요소들을 적용하면 취약한 서비스를 안정적으로 익스플로잇 할 수 있는 능력을 지닌 공격자의 수가 크게 줄어듭니다.

조직에서 개발한 애플리케이션에 원격 코드 실행 기능의 확보를 위해 남용될 수도 있는 보안 결함이 있다면 ASLR, NX, UDEREF 등을 의무화함으로써 남용의 여지를 낮출 수 있습니다. 그러면 공격자가 적당한 시간 안에 익스플로잇을 개발하기가 어려워집니다. 익스플로잇 개발자가 목표물 별로 까다로운 첨단 기법을 따로 만들어야 하기 때문입니다. 이렇게 되면 난이도가 높아질 뿐만 아니라 완벽히 작동하는 익스플로잇을 만들기까지 시간과 비용이 듭니다.

보안 강화요소가 없다면 무료 기성품 소프트웨어를 가지고 몇 시간 안에 완벽한 익스플로잇을 만들 수 있습니다.

8.2 디버깅과 시험 기술 사용 안 함

어떤 제품을 개발할 때 엔지니어링 공정의 편의를 위해 디버깅 기술과 시험 기술을 적용하기도 합니다. 여기에는 문제될 게 없습니다. 그러나 어떤 장치가 양산을 앞두게 되면 승인 구성(Approved Configuration)을 정의하기 전에 그와 같은 기술을 모두 제거해야 합니다.

제품과 함께 배포되는 승인 기술에는 공격자가 남용할 수도 있는 디버깅 인터페이스나 진단, 시험 인터페이스가 없어야 합니다. 그런 인터페이스의 예를 들자면 다음과 같습니다.

- 명령 라인 콘솔 인터페이스

- 장황한 디버깅, 진단 또는 오류 메시지가 있는 콘솔
- JTAG, SWD 등 하드웨어 디버깅 포트
- 디버깅이나 진단, 시험에 사용되는 네트워크 서비스
- SSH 나 Telnet 등 관리 인터페이스

위 기술은 모두 승인 구성에서 꺼둬야 합니다.

시스템이 제거할 수 있는 직렬 포트도 회로 기판에서 떼어내야 합니다. 그러나 UART/USART 처럼 마이크로컨트롤러나 프로세서에서 하드웨어 핀을 통해 켤 수 있는 직렬 포트도 많습니다. 이런 핀들이 콘솔로 켜진다면 공격자가 간단히 핀을 이용해서 콘솔을 조작할 수 있습니다. DB9 인터페이스 등 물리적 직렬 포트 자체를 제거해도 콘솔은 꺼지지 않습니다.

또한 JTAG 와 SWD 와 같은 디버깅 포트도 소프트웨어를 통해서는 끄지 못합니다. 이런 장치들은 보안 퓨즈나 잠금장치를 변경하여 꺼야합니다. 이런 기술들을 소프트웨어를 통해 끄면 공격자가 그 직전에 JTAG 나 SWD, 기타 유사한 하드웨어 디버깅 인터페이스에 연결할 여지가 생깁니다. 그 짧은 순간이면 공격자가 공격을 성공시키기에 충분합니다.

8.2.1 위험

이 권고사항을 구현하지 않으면 중앙처리장치에서 조직의 핵심 비밀이 추출 당할 위험이 있습니다. 그러면 공격자가 자기 펌웨어를 NVRAM 이나 EEPROM 에 로드한 후 핵심 비밀을 추출 또는 변경하여 IoT 네트워크나 장치를 더 크게 침해할 수도 있습니다.

디버깅 포트 끄기는 IoT 제품이나 서비스의 무결성 확보를 위해 매우 중요한 조치입니다. 그러나 조직에서는 이런 기술을 사용하지 않는 데 따르는 위험을 평가하고 필드에서 확인된 문제를 진단하고 디버그하는 데 따르는 편익과 비교해야 합니다. 실행 중인 시스템을 디버그할 방법이 없다면 제품에 존재하는 생산 수준의 결함을 해결하기는 훨씬 더 어려울 수도 있습니다.

8.3 주변기기 기반의 공격을 통해 손상된 메모리

처리 시스템은 일관성이 생명입니다. 그래야 특정 입력에 대하여 알고리즘의 출력이 예측 가능하기 때문입니다. 처리 시스템은 구성요소에 대해서도 안정적으로 거동할 것을 예상하며 작성되는 비트 모두가 프로세서에게 변경될 때까지 안정적이고 변하지 않으리라 예상합니다. 닫힌 시스템에서는 이 이론이 적용됩니다. 이 모델에 이상이 발생하면 그로 인해 처리 환경이 침해를 당하거나 단순히 손상을 입을 수도 있습니다.

정보 보안은 정상 상황에서는 액세스가 불가능한 객체에 액세스할 목적으로 유도된 이상상황의 집단을 표시합니다. 공격자에게 이로운 이상 거동을 유도하기에 좋은 기회는 DMA(Direct Memory Access)입니다. 간단히 말해 DMA 란 외부 구성요소(주변기기)가 CPU 의 간섭을 받지 않고 메인 프로세서 메모리에 액세스할 수 있도록 프로세서가 사용하는 기술입니다. 다시 말하면, CPU 가 주변기기에게 메모리 영역에 대한 직접 액세스를 부여하는 것입니다. 그러면 이 주변기기는 메모리의 그 영역을 읽거나 거기에 쓸 수 있습니다.

프로세서가 주변기기가 이용 가능한 메모리 영역을 제대로 제한하지 않으면 주변기기는 본래 기능에게 필요한 것보다 메인 메모리를 더 이용할 수도 있습니다. 다시 말하면, 주변기기(예컨대 이더넷 컨트롤러)가 수신된 이더넷 프레임에 대한 원형 버퍼 역할을 하도록 DMA 영역을 할당 받았는데, 할당 받은 그 영역이 메인 메모리 전체라면, 이더넷 컨트롤러의 펌웨어가 전체 시스템 메모리를 마음대로 읽고 쓸지도 모르는 것입니다. CPU 는 이더넷 컨트롤러 펌웨어가 메모리에 쓰기를 해도 막을 방법이 없습니다.

이 공격의 영향은 두 가지입니다. 하나는 데이터가 메인 메모리에서 유출돼 비밀 탈출이나 즉시 탈출 목적으로 네트워크 패킷이나 애플리케이션 정보에 인코딩되는 것입니다. 다른 하나는 공격자가 애플리케이션의 실행 코드를 덮어써 메인 메모리에 은밀하게 백도어(맬웨어)를 심는 것입니다.

프로세스로서는 과도하게 허용된 메모리 영역이 악성 주변기기에게 남용을 당했는지 확인할 길이 없습니다. 이 공격에 대처하려면 엔드포인트 시스템에 사용된 프로세서가 DMA 를 메모리의 작은 영역으로 제한할 능력이 있는지를 확인해야 합니다. 그럴 능력이 있다면 주변기기 별로 메모리의 각 영역을 지정해야 합니다. 주변기기가 메모리를 마음대로 이용하게 해서는 안 됩니다.

프로세서 중에는 선형 또는 가상 메모리에서 DMA 구역의 크기나 위치를 미세하게 제한하지 못하는 것도 있을 수 있습니다. DMA 공격을 핵심 애플리케이션의 IoT 엔드포인트에 대한 실질적 위협으로 고려해야 하는 만큼, 더 정교한 기능을 갖춘 다른 프로세서로 교체하는 것이 옳은지 검토해야 합니다.

IEEE1394 나 Thunderbolt, Express Card, 그 외 PCI(Peripheral Component Interconnect) DMA 에 대한 직접 액세스를 허용하는 포트가 노출돼 있는 플랫폼은 이미 비용 대비 효과가 큰 공격의 타깃이 된 것이나 마찬가지입니다.

DMA 기반 공격 시 로컬 하드웨어 구성요소의 남용이 필요한 플랫폼의 경우, 공격의 난이도는 확실히 올라가지만 주변기기의 펌웨어를 리플래쉬(reflash)하여 DMA 의 전복을 통한 로컬 엔드포인트의 침해를 노리는 공격 기반의 보안 교전에서 완전히 벗어난 것은 아닙니다. 그러나 비용과 시간, 전문성이 하나의 인자가 되어 청부 공격자가 동원될 가능성이 높아집니다.

8.3.1 위험

외부 구성요소가 DMA 를 남용할 여지를 줄이지 않으면 플랫폼이 전면 침해를 당하거나 최소한 엔드포이트에서 핵심 비밀이나 프라이버시 데이터, 지적재산이 추출될 수도 있습니다.

8.4 사용자 인터페이스 보안

터치 스크린이나 리치 디스플레이, 대체 인터페이스 기술 같은 사용자 인터페이스가 있는 IoT 엔드포인트는 반드시 안전하게 정보를 사용자에게 제공하고 사용자에게 받아야 합니다.

비밀번호 등 사용자 인터페이스의 속성은 이미 본 자료에서 다뤘지만, 논의가 필요한 미묘한 문제가 몇 가지 더 있습니다.

- 경보 시스템
- 조치 확인

물리적 탬퍼링이나 의도하지 않게 움직이는 애플리케이션처럼 이상이 발생하면 사용자에게 눈에 보이는 경보가 발령되어야 합니다. 아니면, 사용자가 사용자 인터페이스 안에서 시스템의 경보를 확인할 수 있어야 합니다.

또한 장치가 실시하는 동작 중에서 인코딩 또는 한 인터페이스에서 다른 인터페이스로의 원활한 전환이 밑바탕이 되는 것은 모두 다 사용자가 확인해야 합니다. 예를 들면 장치 카메라가 QR 코드를 읽거나 NFC 또는 RFID 상호작용이 장치의 URL 연결을 요청할 때가 그렇습니다. 이 경우 사용자는 동작에 대해 확인 요청을 받고 그 동작의 실행이 바람직하다는 것을 검증해야 합니다. 사용자에게는 동작을 취소할 선택권이 있어야 합니다. 사용자는 연결될 URL 을 포함해 어떤 동작에 관한 세부내용을 모두 볼 수 있어야 합니다.

8.4.1 위험

본 권고사항을 구현하지 않으면 사용자는 탐지가 안 되는 공격에 취약한 상태가 됩니다. 어떤 시스템 설계자들은 RFID 칩에서 예컨대 상응하는 제품 사이트로 자연스럽게 전환되는 것을 높이 평가하지만 이 거동에는 바람직하지 않은 효과도 있을 수 있습니다. 즉 사용자가

동의하지도 않았는데 억지로 바람직하지 않은 자료를 보게 될 수도 있고, 또 속아서 보안이나 프라이버시를 약화시키는 사이트를 방문하거나 어떤 동작을 하게 될 수도 있습니다.

또한 경보를 검토하는 데 애로가 있는 사용자라면 탬퍼링됐을지도 모르는 장치를 이용하는 위험을 알지 못할 수도 있습니다. 그러면 사용자는 물리적 보안이 약화되고 위험에 빠질 수도 있습니다.

8.5 3 자 코드 감사

언제든 코드의 일부, 예컨대 부트로더가 보안 런타임 플랫폼 구축에 중요한 요소가 된다면 위험 여부에 대해 감사를 실시해야 합니다. 공격자가 부트로더를 조작해 신뢰할 수 없는 코드를 실행하거나 인증 시퀀스를 우회하게 할 수 있다면 기술은 무용지물이 되고 맙니다. 그러면 이 기술의 도입을 위해 조직에서 투입한 자금과 시간, 경험은 허사가 되고 엔지니어링 비용만 낭비하고 마는 것입니다.

이 영역에서 보안에 허점이 생기면 경쟁사가 스푸핑이나 API 남용, 데이터 가로채기, 장치 복제, 심지어 장치 리브랜딩을 통해 이익을 편취할 수도 있습니다. 따라서 코드에서 중요한 부분은 승인 받은 3 자가 점검하여 기술이 남용의 위험에 놓이지 않게 해야 합니다. 감사를 담당할 적합한 정보 보안팀을 찾으려면 먼저 어떤 유형의 코드에게 감사가 필요한지 파악해야 합니다. 대개 이 모델에서는 C 와 Assembly 이며 C++나 Java 도 가능성이 있습니다.

이들 언어뿐만 아니라 기초가 되는 아키텍처에 정통한 팀을 찾아야 합니다. 소스 코드 감사를 실시하는 정보보안팀은 많지만 IoT 기업이 사용 중인 특정 플랫폼을 감사하는 팀은 많지 않을 수도 있습니다. 플랫폼마다 조금씩 다르므로 사용 중인 플랫폼을 잘 아는 팀을 찾는 것이 최선입니다.

8.5.1 위험

외부 컨설턴트를 동원해 내부적으로 개발된 기술을 평가하는 것이 쉽지만은 않겠지만 보안에게는 반드시 필요한 일입니다. 기술을 개발하는 엔지니어들은 자기네 아키텍처가 검증 가능하다는 것을 보여줄 수 있어야 하기 때문입니다. 아키텍처를 개발한 엔지니어들이 아키텍처를 검토한다면 결코 쉬운 일이 아닐 것입니다. 엔지니어들은 *실제 구현체*가 아니라 자기네가 설계하고 실행하*고자* 했던 아키텍처에서 나온 코드를 시각화하려는 경향이 있습니다. 그러므로 아키텍처와 구현체에서 보안의 허점을 유발할 수도 있는 미묘한 차이를 찾아내기 위해서는 3 자의 눈이 필요합니다.

8.6 사설 APN 이용

3GPP 셀룰러 네트워크에서는 APN(Access Point Name)이 인증 받은 장치만을 위해 구성된 사설 네트워크 역할을 합니다. 일반적으로, 사설 APN("보안 APN"이라고 함)이란 특정 기업과 관련 있는 인증 받은 장치만 접속할 수 있는 개인(private) 네트워크를 말합니다. 기업은 APN 을 이용해 셀룰러 네트워크를 통해 자사 서비스 인프라에 연결할 수 있는 엔드포인트를 제한할 수 있습니다. 이렇게 하면 백엔드 인프라에서 IoT 서비스에 직접 액세스할 수 있는 사용자의 수를 줄일 수 있습니다.

사설 APN 의 다른 속성으로는 불량 엔드포인트가 IoT 생태계를 남용할 가능성을 제한한다는 것도 있습니다. 방화벽은 APN 으로 또 APN 에서 연결할 수 있는 서비스와 컴퓨터를 제한할 수 있습니다. 잘 구성된 APN 은 엔드포인트끼리 직접 연결하는 것을 허용하지 않습니다. 그러면 침해 당한 엔드포인트가 네트워크 인프라를 타고 다른 엔드포인트로 가지 못합니다.

회사가 함께 일하고 있는 셀룰러 사업자나 모바일 가상 네트워크 사업자(MVNO)를 참여시켜 보안 APN 안에서 어떤 기술이 이용 가능한지 파악해야 합니다. 모니터링이나 이상 장치의 블랙리스팅, 사용자 ID 와 동작의 연동 등 다른 서비스도 가능할지 모릅니다.

8.6.1 위험

사설 APN 을 이용하면 여러 유형의 공격을 막아낼 수 있습니다. 예컨대, 사설 APN 을 이용하면 엔드포인트와 인터넷의 직접 연결 수가 줄어듭니다. 엔드포인트는 신뢰할 수 없는 인터넷 리소스와 직접 연결해서는 안 됩니다. 파트너 조직만 신뢰해야 하며 서비스는 인증을 받아야 합니다.

사설 APN 을 이용하지 않으면 침해 당한 엔드포인트가 제한 없이 인터넷 서비스나 프로토콜과 통신할 수 있습니다. 그러면 공격자가 엔드포인트를 남용해 다른 인프라에 2 차 공격을 가할 수 있습니다. 가령 DoS 공격을 동원하거나 다른 회사나 정부, 민간을 상대로 더 위험한 공격을 지원할 수도 있습니다.

그러나 유념해야 할 사실은 사설 APN 이 공격자가 엔드포인트와 사설 APN 간 통신 링크를 침해할 위험을 완화하지는 못한다는 것입니다. 더구나 사설 APN 은 백엔드 서비스에 대한 게이트 역할에 그칠 뿐이며 IoT 서비스 업체의 사설 네트워크상에 존재하는 APN 과 백엔드 서비스 사이에 어떠한 보안도 의무화하지 않습니다. 이와 같은 잠재적 보안의 허점은 사설 APN 의 이용에 따라 제공되는 개선사항과 관계 없이 반드시 별도로 해결해야 합니다.

8.7 환경 록아웃(lock-out) 임계값 구현

임베디드 시스템의 구성요소는 일정한 환경 임계값과 함께 사용되도록 설계돼 있습니다. 전압, 전류 인출, 외기 또는 작동 온도, 습도 등이 그것입니다. 각 구성요소는 대개 일정한 승인 수준에 맞춰 등급을 받습니다. 장치의 상태가 특정 구간을 초과하거나 미달하면 구성요소는 불규칙하게 움직이거나 공격자에게 유익하게 거동할 수도 있습니다.

그러므로 이들 환경 수준의 변화를 탐지하여 장치를 계속 가동해야 하는지, 전원을 차단해야 하는지 파단하는 것이 중요합니다. 그러나 전원 차단이 원하는 효과이고 공격자는 이 같은 엔지니어링 결정을 틈타 DoS 를 이용하려 할 수도 있음을 유념해야 합니다. 엔지니어링 팀은 이 모델을 평가해 전원을 차단하는 것이 유익한지, 온라인 상태를 유지하는 것이 유익한지 판단해야 합니다.

어느 쪽이든 이 모델에는 보통 다음 요소가 들어 있습니다.

- 전압이 지나치게 낮아질 때 브라운아웃(brown out)과 블랙아웃 탐지
- 전압이 임계값을 초과하지 않게 하는 전압 제한 회로 보호기능
- 전류 소모가 일정 수준을 미달하거나 초과하지 않게 하는 전류 제한 회로
- CPU, MCU 등의 내부 수준을 모니터링 하는 내부 온도 모니터링 기능
- 옵션으로, 습도를 평가하여 환경이 지나치게 습하거나 건조하지 않은지 판단하는 기능

고온은 사용자나 환경, 하드웨어나 소프트웨어 문제 때문에 회로에 문제가 생겼음을 나타내므로 온도는 매우 중요합니다. 온도를 모니터링 하면 운영체제나 애플리케이션이 리소스(또는 전체 장치)를 차단해 엔드포인트에게 화재나 기타 문제가 발생하지 않게 할 수 있습니다.

낮은 온도 또한 장치의 거동에 영향을 줍니다. 회로가 느려지거나 구성요소가 예상 외로 반응하기도 합니다. 온도가 예측 가능한 이상으로서 애플리케이션이나 회로에 유리하게 작용하는 예측 가능한 이상을 유발할 수 있다면 이 같은 현상은 공격자에게 유리합니다.

온도와 습도를 분석할 때 잠금 임계값의 난이도가 드러납니다. 전압과 전류 수준은 회로 기판이나 프로세서에 있는 브라운아웃 또는 블랙아웃 회로로 다스려야 합니다. 엔지니어는 어떤 칩의 전압과 전류의 임계값과 관련된 수치를 찾아볼 수 있으므로 이 같은 문제에 대한 방호조치를 쉽게 구현할 수 있습니다.

온도와 습도는 대응 조치를 결정하기가 더욱 까다롭습니다. 공격자가 실물 장치를 만지지 않고도 수치를 만들어낼 수 있기 때문입니다. 온도의 경우, 안전 사건이 머지 않았음을 나타내는 수치에 이르면 장치는 적절한 조치를 취해 온도를 낮춰야 합니다. 그러나 산업용 제어 시스템이나 의료 기기처럼 중요한 환경에서는 장치가 가능하면 핵심 동작을 계속해서 실시해야 합니다. 수치가 엔지니어와 회사 수뇌부가 합의해 정한 수준을 넘었을 때에만 장치를 꺼야 합니다.

8.7.1 위험

전압과 전류 소모의 경우, 남용의 위험은 글리칭, 그리고 위 수치를 바꾸면 유익해지는 기타 사이드 채널 공격과 관련이 있습니다. 브라운아웃이나 블랙아웃 탐지 기능이 프로세서에 구현돼 있다면 남용의 위험은 낮아집니다. 그렇지 않다면 전압 또는 전류의 급등이 실물 장치에 안전 문제를 유발할 수도 있다. 또한 급등으로 인해 공격자가 글리칭(또는 유사) 공격을 실행해 구성요소의 보안을 무력화할 수도 있습니다.

이 같은 문제는 전압이나 전류의 이상 급등으로부터 구성요소를 보호하는 회로를 PCB 에 적용하여 대응해야 합니다.

급격한 환경 수치 변화의 경우 위험은 사용자의 안전과 관련이 있습니다. 과도한 CPU 사용이나 그 외 이상으로 온도가 높아지면 화상이나 화학약품 화상, 화재가 발생할 수도 있습니다.

8.8 전력 경고 임계값 의무화

사용자에게 핵심 서비스를 제공하는 엔드포인트는 전력 관련 이벤트를 표시하는 경고 임계점과 함께 가동해야 합니다. 전력 관련 이벤트란 다음을 말합니다.

- 배터리 부족 상태
- 배터리가 매우 부족한 상태
- 블랙아웃 이벤트
- 브라운아웃 이벤트
- 배터리 전환 백업 이벤트

사용자에게는 전력 손실을 보충할 수 있도록 충분한 시간을 두고 경고가 전달되어야 합니다. 전력 상태를 표시하는 LED 를 켜는 것이 한 가지 방법이 될 수 있습니다. 예컨대 정상은 녹색, 부족은 주황, 매우 부족은 빨강으로 하는 것입니다.

교류 주전력에 연결되는 시스템은 블랙아웃이나 브라운아웃 발생 시 사용자에게 경고를 하도록 구성해야 합니다. 또한 엔드포인트는 항구적 메모리에 이들 이벤트를 기록하여 사용자와 관리자가 나중에 확인할 수 있게 해야 합니다. 정보에는 시각이 표시되어야 합니다.

이 프로세스에서 어려운 점은 배터리의 고갈 속도와 전력 상태의 변동을 사용자에게 통보할 때 필요한 추가 에너지를 확인하는 것입니다. 이것은 전기 엔지니어링으로 해결할 수 있으며 노련한 엔지니어링 업체에게는 그다지 어려운 일이 아닙니다.

8.8.1 위험

잘 정의된 전력 경고 시스템이 없다면 사용자가 중대한 전력 변동에 제대로 대비할 방법이 없습니다. 속도 카운터나 타이머, 기타 웨어러블 장치처럼 단순한 장치에서는 별 문제가 안 될 수도 있지만 개인 추적기, 텔레매틱스 시스템, 홈 시큐리티 시스템 등 더 중요한 장치는 전력이 끊어질 경우 심각한 타격을 입을 수도 있습니다.

8.9 백엔드 연결 기능이 없는 환경

8.9.1 방법

엔드포인트, 특히 게이트웨이나 게이트웨이 역할을 하는 엔드포인트 백엔드 네트워크에 연결이 불가한 환경에서도 통신 보안을 확보할 수 있어야 합니다. 백엔드 네트워크와의 연결 부재가 일시적이든 아니든 게이트웨이나 엔드포인트는 마치 백엔드 시스템이 있는 것처럼 보안을 확보할 수 있어야 합니다.

이를 위해서는 TCB 를 동원해 엔드포인트가 비공개 데이터, 구성 데이터 또는 명령 데이터를 보내야 하는 피어 전체를 인증해야 합니다. TCB 를 이용하면 피어와 주고 받는 메시지가 같은 조직에서 프로비저닝한 엔터티에서 들고 나가게 할 수 있습니다. 이렇게 되면 공격자 장치와 통신을 할 가능성이 낮아집니다.

인증이 불가능한 다른 장치와 통신은 하므로 상호운용성은 유지됩니다. 그러나 그런 장치와 주고 받을 수 있는 정보는 상호운용 및 비민감성 데이터로 한정해야 합니다.

난제는 어떤 엔드포인트를 인증하고 어떤 엔드포인트와 평문으로 통신해야 하는지 결정하는 데 있습니다. 조직은 어떤 유형의 데이터를 기밀로 분류해 인증 받지 않은 피어로부터 보호할지 결정해야 합니다. 이렇게 데이터 분류를 마치면 조직에서는 핵심 IoT 서비스의 도움 없이도 어떤 피어가 신뢰할 만한 것인지 판단할 수 있습니다.

8.9.2 위험

통신이 없는 환경에 솔루션을 도입할 때에는 경쟁사가 인프라를 남용할 여지가 발생한다는 위험이 있습니다. 경쟁사는 상호운용성을 제공하고 외부 연결이 없는 사이트를 시험장으로 이용하여 회사를 약화시킬 수 있습니다.

대신 조직에서는 상호운용성을 허용하되 특정 포인트로는 불허하도록 선택할 수 있습니다. 그러면 일정한 핵심 지적재산과 서비스가 TCB 를 통해 검증 받는 인증 피어에게 한정됩니다. 이렇게 되면 회사가 지적재산 문제와 공격적 경쟁사에게 덜 노출됩니다.

8.10 장치의 퇴역과 일몰

앞서 언급했듯이 엔드포인트 장치에게는 수명주기가 있습니다. 사용자가 구독을 취소해 퇴역을 시켜야 하는 장치가 있는가 하면 이상 또는 공격 행동으로 인해 퇴역을 시켜야 하는

것도 있습니다. 이유가 무엇이든 회사는 TCB 및 통신 모델을 이용해 장치를 안전하게 퇴역시킬 준비가 돼 있어야 합니다.

앞서 언급했듯 일몰(sunsetting)은 이들 장치를 지원하는 장치와 서비스 네트워크 전체를 퇴역시키는 과정을 말합니다. 회사에서 효용 가치를 다한 제품이나 서비스, 또는 폐업을 결정한 회사는 그 장치와 네트워크를 일몰시켜 공격자가 그 네트워크를 접수해 남용할 위험을 일소해야 합니다.

이를 위해서는 TCB 와 지원 프로토콜을 이용해야 합니다. 일반적으로 그 프로세스는 다음과 같습니다.

- 서비스 생태계에서 퇴역 메시지 생성
- 그 메시지를 받을 엔드포인트 별로 메시지 내용 조정
- 퇴역 PSK 또는 비대칭 키로 메시지에 서명
- 메시지를 엔드포인트로 전송
- 엔드포인트에서 퇴역을 확인하는 메시지를 크립토그래픽 방식으로 수신
- 인증 받은 장치 목록에서 해당 엔드포인트 효력 말소
- 그 엔드포인트와 추가 통신 불허

장치 측에서는 소프트웨어서 실행 중인 애플리케이션이 다음과 같이 조치합니다

- 서비스 생태계를 통해 주요 백엔드 서비스에 연결
- 서비스에 중요한 메시지 여부를 질의
- 퇴역 메시지 수신
- TCB 와 트러스트 앵커를 이용해 메시지 서명 검증
- 확인 메시지를 생성한 후 개인화 PSK 또는 비대칭 키로 크립토그래픽하게 서명
- 퇴역 작업 실시
- 주요 서비스에 메시지 재발신

여기서, 퇴역 전에 메시지에 서명하여 전송 준비를 마쳐야 한다는 점이 중요합니다. 퇴역 프로세스에 트러스트 앵커에서 보안 키를 무효화하고 제거하는 작업이 들어 있기 때문입니다. 이 프로세스 때문에 퇴역 메시지에 서명할 때 사용된 키가 사용 불가 상태가 됩니다. 서비스는 무결성 검증이 가능한 메시지를 수신해야 합니다. 엔드포인트가 실제로 메시지를 받아서 처리했는지 확인하기 위함입니다.

이 프로세스에서 어려운 점은 침해 가능성이 있는 어떤 장치를 퇴역시킬 때 그 장치가 퇴역 명령을 거부할 정도로 침해를 받지는 않았다고 가정한다는 데에 있습니다. 만일 침해가 심각하다면 퇴역 명령을 따르지 못할 수도 있습니다.

그러므로 서비스 생태계에서 실행 중인 백엔드 시스템이 해당 엔드포인트에 대해 주요 서비스와 통신하지 못하게 하는 것이 중요합니다. 만일 해당 장치가 네트워크상의 피어나 주요 서비스와 소통하고자 한다면 백엔드 시스템은 경보를 발령하고 관리자에게 이상 이벤트의 발생 사실을 알려야 합니다.

8.10.1 위험

퇴역과 일몰을 구현하지 않을 때 따르는 위험은 공격자에게 네트워크 전체를 장악 당하는 것부터 침해 당한 장치가 네트워크상의 서비스를 계속 이용하게 하는 것까지 다양합니다. 가장 흔한 위험은 사용자가 IoT 서비스 업체의 이용을 중단했을 때 발생합니다. 이 사용자를 네트워크에서 퇴역시키지 않으면 이 사용자가 IoT 엔드포인트 네트워크에서 다른 피어와 계속 통신하거나 더 이상 이용하면 안 되는 서비스를 이용할 수도 있습니다. 그러면 IoT 서비스 업체가 서비스 생태계에서 대역과 CPU 시간, 저장장치 비용을 부담해야 하는 상황이 됩니다.

8.11 무단 메타데이터 수확(harvesting)

현대의 IoT 는 현실 세계를 디지털 세계와 연결하도록 설계돼 있습니다. 이 모델에서는 기술의 영향이 과거보다 훨씬 더 깊고 넓습니다. 기업과 개인은 메타데이터를 이용해 불특정 또는 특정 소비자의 행동을 의도적으로 추적하고 모니터링합니다.

두 네트워크 엔터티 간의 통신이 암호화되었을 때에는 메타데이터 분석이 이용되지만, 메시지의 유형이나 발신자/수신자의 신원을 표시하는 프로토콜 구조는 노출됩니다. 이 메타데이터에서 의도를 추출할 수 있습니다.

가령, 자동차에서 특정 소비자에 관한 메타데이터가 들어 있는 메시지를 내보낸다고 합시다. 이 메타데이터를 (로컬 또는 원격으로) 추적할 수 있는 능력을 지닌 사람이라면 그 소비자의

이동을 모니터링해 그 이동에서 행동 패턴이나 의도를 추출할 수도 있습니다. 만약 자동차의 텔레매틱스 시스템에 익스플로잇할 수 있는 보안 결함이 존재한다면 특정 소비자의 텔레매틱스 시스템을 추적하여 물리적 위해를 가할 수 있을지도 모릅니다.

사법 당국과 보험사에서는 이와 같은 위험이 자동차 금융에 어떤 영향을 미칠지 주시하고 있으며 텔레매틱스 장비의 설계 방식을 결정할 법과 표준의 제정에 참여하기 시작했습니다. 이 같은 변화는 결국 기술의 발전에 따라 낮은 단계의 IoT 부분까지 전파될 것입니다.

메타데이터 수집(harvesting)에 대처하는 길은 데이터를 최대한 암호화하고 통신 모듈에 고유한 바이너리 식별자를 이용하는 것입니다. 외부 사용자가 IoT 시스템의 API 를 이용하여 사용자 프로파일에서 하드웨어 일련번호와 기타 추적 가능한 신원 정보를 추출하지 못하게 하는 정책을 의무화해야 합니다. 가능하다면, 메시지의 구조가 3 자에게 노출되지 않게 해야 합니다. 동작이나 활동, 거동이 3 자에게 노출되지 않게 해야 합니다. 사용자 프라이버시와 관련된 데이터 전체를 대상으로 기밀유지와 무결성을 의무화해야 합니다.

8.11.1 위험

통신 보안이 약하면 최종 사용자를 위험에 빠뜨리거나 최종 사용자의 프라이버시를 침해하는 데이터나 메타데이터 수집의 위험이 있습니다. 보험사에서 기술을 대상으로 최종 사용자 프라이버시 요건의 의무화하를 추진하고 있어 회사가 자사 장치에서 생성되는 데이터에 책임을 지지 않는다면 위험을 자초할 수도 있습니다.

9 낮은 우선순위 권고사항

낮은 순위 권고사항은 대처하기에 지극히 큰 비용이 드는 위험이나 엔드포인트 설계에 영향을 줄 가능성이 낮은 위험에 적용되는 권고사항을 말합니다. 이 권고사항은 가치가 있고 그 안에 든 내용도 중요하지만 소개되는 완화 또는 시정 전략은 회사에 따라 연관성이 낮을 수도 있습니다. 각 권고사항을 평가하여 소개된 위험이 회사 및 고객과 관련이 있거나 중요한 것인지 판단하기 바랍니다. 고객이 위험의 해소를 요구한다면 권고사항을 적용해야 합니다.

9.1 서비스의 의도적, 비의도적 거부

무선 통신은 끊임 없이 *재밍*의 위협에 놓여 있습니다. 재밍이란 적법한 신호를 훼방하고자 의도적으로 잡음이나 패턴을 송출하는 행위를 말합니다. 무선 신호는 특정 패턴으로 공간을 날아가는 전자의 집합이므로 통신 데이터를 형성하는 패턴을 방해하거나 훼손하는 신호를 만들기는 꽤 간단합니다.

대개 이 같은 공격의 목표는 적법한 사용자에게 서비스가 도달하지 못하게 차단하는 것입니다. 때로는 목적을 가지고 남용이 일어나기도 합니다. 예컨대, 인증 메커니즘이 없는 통신 프로토콜은 *스푸핑*을 당할 수도 있습니다. 이를 위해서는 실제 신호를 *재밍*하여 공격자의 스푸핑 신호가 목표에 닿을 가능성을 높여야 합니다.

GPS(Global Positioning Systems) 스푸핑이 그 예입니다. 민간 GPS 신호는 기본적으로 누구나 받을 수 있는 평문 송출 신호이기 때문에 엄호화와 인증이 약합니다. 또한 비교적 약한 무선 신호이고 TV, 전자렌지의 UHF 전치 증폭기와 같은 환경 이상을 만나면 쉽게 약해집니다.

위치 정보를 받아야만 제기능을 하는 장치가 재밍을 당한 GPS 신호를 받으면 신뢰도에 위험이 생기며 이것은 다시 정보 보안 위험으로 이어질 수도 있습니다. 뒤이어 스푸핑이 적용될 때 특히 더 그렇습니다.

재밍과 기타 형태의 DoS 공격에 대처하기 위해서는 서비스 차질의 파장을 최소화하는 방법에 중점을 둔 견고한 통신 프로토콜을 개발해야 합니다. 네트워크는 장치가 갑자기 또는 비정상적으로 네트워크에서 사라졌는지 탐지해야 합니다. 각 엔드포인트는 네트워크를 떠나기 전에 "작별 인사"를 해야 합니다. 그러지 않는다면 통계 분석에 이상이라고 표시해야 합니다.

또한 장치가 네트워크에 합류할 때마다 통신 보안 키를 재협상해야 합니다. 같은 통신 키를 다시 사용하면 안 됩니다. 같은 비대칭 크립토그래픽 키로 *부트스트랩*해야 하지만 키 협상에서 도출된 비대칭 키는 어느 것이나 통신 세션마다 새것이어야 합니다.

무선 통신에서는 여러 가지 이유로 의도하지 않은 재밍이 발생합니다. 신호의 전파를 방해하는 환경 조건, 장비 오작동, 같은 주파수에서 작동하는 인접 장비 등이 그것입니다. 이유가 무엇이든, 무선 통신을 이용하는 엔지니어는 신호의 약화 또는 두절을 유발하는 일시적 상황이 있을 것임을 예상하고 있습니다. 이 같은 두절은 애플리케이션*과* 네트워크 통신 프로토콜의 설계를 통해 보상을 해야 합니다.

개발자들은 GSMA 의 연결 효율 가이드라인[9](Connection Efficiency Guidelines)을 참고하기 바랍니다. 여기에는 의도하지 않은 DoS 공격에 대처하는 방법과 DHIR(Device Host Identity Reporting)에 관한 지침이 수록돼 있습니다.

9.1.1 위험

의도적인 않은 DoS 의 위험에 대처하지 않으면 비정상이거나 보안이 결여된 엔드포인트의 거동이 나타납니다. 엔드포인트에서 항상 동일한 세션 키가 사용된다면 공격자가 그것을 빌미로 네트워크 아키텍처를 남용해 통신 보안에 사용되는 대칭 키에 관한 정보를 수집할 수도 있습니다. 세션이 분리될 때마다 보안 세션을 구축하는 것이 엔드포인트 통신의 보안을 위해 중요합니다.

9.2 안전 위주의 분석

IoT 제품은 대부분 디지털 기술로 현실 세계의 일부 측면을 수용합니다. 그러므로 인간이 IoT 엔드포인트에서 제공 받은 정보를 토대로 현실 세계에서 의사결정을 내릴 가능성이 높습니다. 아니면, IoT 엔드포인트가 디지털 세계에서 입수한 정보를 가지고 현실 세계에 영향을 주는 결정을 내릴 수도 있습니다.

따라서, IoT 서비스 업체는 안전의 관점에서 자사 제품을 평가하여 기술로 인명이 영향을 받을 수도 있는지, 그렇다면 어떻게 언제 그런지 파악해야 합니다. 인체에 위해가 되는 기술의 남용을 막는 안전조치를 제대로 강구하지 않으면 고객이 위험에 처할 수도 있습니다.

안전 문제에 대처하기 위해서는 IoT 서비스 업체의 실무, 법무, 보험팀과 협의를 해야 합니다. 이들 팀이 제품이나 서비스에 사용되는 기술이 능력과 한계를 알고 있는지 확인해야 합니다. 이들 기술이 회사의 니즈에 부합하고 원하는 용도에 필요한 만큼 고객을 안전하게 보호할 수 있는지 판단해야 합니다.

9.2.1 위험

충분한 시간을 들여 제품이나 서비스가 고객의 안전에 미치는 영향을 평가하지 않으면 수익의 손실이나 예상치 못한 사고, 심하면 사망에 이를 수도 있습니다.

9.3 보이지 않는 컴포넌트와 신뢰하지 않는 브리지 차단

실물 회로에 사용되는 구성요소는 서로 또는 중앙처리장치와 통신할 때 대체로 기밀유지와 무결성 대책을 강구하지 않습니다. 그러므로 공격자 누구나 이들 버스에서 전송되는 데이터를 읽고 쓸 수 있습니다. 통신 보안에서 이와 같은 허점의 결과는 공격자가 실물 회로에서 합법적 장치를 사칭하는 것입니다. 공격자는 마음만 먹으면 NVRAM 이나 RAM, 심지어 트러스트 앵커와 같은 핵심 구성요소를 사칭할 수 있습니다.

이와 같은 공격의 목표는 버스 위 두 구성요소 사이에 적용된 보안을 우회하는 것입니다. 이 시나리오의 전형적인 예는 이 약점을 이용하여 NVRAM 에 저장된 애플리케이션 이미지를 분석하는 무결성 검증 프로세스를 우회하는 것입니다. CPU 가 NVRAM 에 저장된 메모리를 불러올 때 공격자는 패스스루 시스템을 이용해 CPU 에 실제 메모리 콘텐츠를 공급할 수 있습니다. CPU 에서 실행되고 있는 애플리케이션이 애플리케이션 이미지의 무결성을 검증하면, 공격자는 실물 버스에서 통신을 설정하여 공격자에게 유리한 NVRAM 콘텐츠로 바꿔치기 할 수도 있습니다. 다시 말하면, CPU 가 어떤 애플리케이션 이미지(원 이미지)를 검증한 후에 공격자의 이미지를 RAM 에 불러와서 실행하는 것입니다.

이 공격에 대처하는 방법은 다음과 같습니다.

- NVRAM 콘텐츠를 RAM 으로 로드
- RAM 에 로드된 애플리케이션 이미지 검증
- RAM 에서 코드 직접 실행 또는 RAM 에서 콘텐츠 캐싱

이때 공격자가 RAM 을 무력화해 이 프로세스를 약화시킬 수도 있습니다. 그러나 RAM 을 대상으로 '중간자' 공격을 실시하는 것은 NVRAM 에 대한 공격보다 훨씬 더 복잡하고 비쌉니다. NVRAM 에 비해 버스가 훨씬 더 빠르고 액세스(주로 블록으로 액세스) 패턴이 훨씬 더 불규칙하기 때문입니다.

아니면, 공격자가 검증 받은 NVRAM 콘텐츠 중에서 작은 영역에 대해 체크섬을 만들어 NVRAM 에서 나오는 신호를 주기적으로 체크할 수도 있습니다. 체크섬이 다르다면 콘텐츠가 조작을 당하고 있는 것입니다. 이것은 성공할 수도 있지만 그 가능성은 낮습니다. 공격자가 실행 중인 애플리케이션이 무작위로 체크하지 않는 소량의 데이터만 조작할 수 있기 때문입니다.

이와 같은 공격에 대처하는 길은 NVRAM 의 콘텐츠를 검증한 후 실행 가능한 RAM 으로 불러오는 것이 최선이지만, 이 문제에 완벽한 해결책은 없습니다. 실물 부품에 보안을 확보하는 비용은 워낙 높기 때문에 고객이 원하지 않는 한 이 공격을 더 완벽하게 차단하려고 하는 것은 현실적이지 않습니다.

이 공격은 I2C 처럼 더 기본적인 물리적 통신 프로토콜이 사용되면 더욱 간단합니다. I2C 와 같은 버스는 기본적으로 물리적인 송출 시스템입니다. 그러므로 I2C 버스 위에 앉아 있는 구성요소는 다른 구성요소인 척 할 수 있습니다. 그러면 공격자가 통신 채널에 대해 기밀유지와

무결성을 의무화하지 않는 버스에서 다른 장치를 사칭할 수 있습니다. 이것이 우려된다면 물리적인 버스 프로토콜 위에서 사용되는 애플리케이션 프로토콜에 대해 기밀유지와 무결성을 의무화해야 합니다.

9.3.1 위험

아무 솔루션도 구현하지 않는다면 공격자가 애플리케이션에서 무결성 체크를 우회할 수 있게 됩니다. 그러면 공격자는 더 권한이 높은 코드로 실행되는 애플리케이션(부트로더, TCB 등)을 침해할 수 있습니다.

그러나 이 공격은 부트로더에 대한 단순한 공격과 비교해 가능성이 훨씬 더 낮습니다. NVRAM 과 같은 부품이나 RAM 과 같은 고속 부품을 대상으로 '중간자' 공격을 감행하기는 까다롭고 복잡하며 아직 비쌉니다. 공격자가 이런 방식으로 임베디드 시스템을 무력화하는 것은 언제나 가능은 하겠지만 실행에 옮기기에는 비용이 너무 많이 듭니다.

그러므로 코드를 RAM 으로 불러오고 무결성을 검증하는 것만으로 공격의 대부분은 피할 수 있습니다.

또한 앞서 설명한 바로 그 이유 때문에 크립토그래픽 키를 이처럼 보안이 결여된 권리에 보관해서는 안 됩니다. 트러스트 앵커에 보관하고 TCB 사 사용해야 합니다. 사칭이나 침해의 가능성이 있는 NVRAM 과 같은 매체에는 보관하면 안 됩니다.

9.4 콜드 부팅 공격의 차단

콜드 부팅 공격[참조]은 컴퓨터에서 실물 메모리를 빼내 공격자가 통제하고 있는 2 차 시스템에 끼워넣어 비밀을 추출하는 공격 방식을 말합니다. 이 공격의 이점은 공격자가 자체 제작한 운영체제를 실행시켜 RAM 의 내용물을 항구적 저장장치로 옮길 수 있다는 것입니다. 그러면 공격자는 불러온 데이터를 면밀하게 검토하여 사용 가능한 보안 관련 토큰이 있는지 확인할 수 있습니다. 주요 사례는 다음과 같습니다.

- 크립토그래픽 비밀 또는 개인 키
- 로그인 자격증명(사용자 이름과 비밀번호)
- 개인 식별 정보(PII)
- 웹 서비스 접속 토큰

공격의 목표는 공격자가 어떤 리소스(공격이 아니면 방법으로는 이용하지 못하는 것)를 장기적으로 이용할 수 있는 비밀을 획득하는 것입니다. 예컨대, TLS 의 최신 표준에 사용되는 크립토그래픽 알고리즘은 웬만한 공격자가 깨뜨리기란 불가능합니다. 그러나 상호 인증 TLS 서비스에 사용되는 사설 클라이언트 인증서를 침해하면 공격자가 더 편리한 시스템에서 클라이언트를 시뮬레이션할 수 있습니다.

이 공격을 성공하려면 공격자가 칩에 저장된 비트를 바꾸지 않고 대상 컴퓨터 시스템에서 RAM 을 떼어낼 수 있어야 합니다. 연구 보고서에서 설명한 바와 같이 이것은 메모리 칩을 냉각하면 가능합니다. 그러나 RAM 은 쉽게 떼어낼 수 있어야 합니다. RAM 이 회로 기판에 납땜으로 부착돼 있다면 공격이 매우 복잡해집니다. 공격자는 남땜 인두로 메모리를 추출해야 하는데 이 과정에서 콘텐츠가 손상될 가능성이 있습니다.

엔드포인트의 프라이버시를 높이려면 항상 메모리를 끌 때 스크럽(scrub)하는 것이 중요하고 바람직합니다. 그러나 콜드 부팅 공격은 언제든, 심지어 시스템이 실행 중일 때에도 일어날 수 있습니다. 그러므로 메모리를 스크럽하면 좋긴 하지만 실제 공격을 차단하지 못할 수도 있습니다.

이 공격을 차단하는 방법으로 더 효과적인 것은 CPU 안에 있는 RAM 으로 보안 관련 동작을 처리하는 것입니다. CPU 와 MCU, MPU 중에는 실행 중인 애플리케이션에서 사용할 수 있는 내부 SRAM 이 소량 들어 있는 것이 많습니다. 애플리케이션이 핵심 보안 토큰(개인 키)을 이 내부 RAM 에서만 사용할 수 있게 한다면 탈착 가능한(또는 외부) RAM 의 내용물은 공격자에게 매력이 덜 할 것입니다.

9.4.1 위험

콜드 부팅 공격의 위험을 감안하지 않는다면 핵심 보안 키가 간단한 공격 모델로 탈취 당할 수도 있습니다. 보안 키가 IoT 시스템 업체의 생태계 안에 있는 모든 엔드포인트에게 동일하다면 대규모 침해가 발생할 수도 있습니다.

자세히 알아보기: https://citp.princeton.edu/research/memory/

9.5 명확하지 않은 보안 위험(시 스루 월)

통신 네트워크에서 상호 인증과 기밀유지, 무결성을 구현해 의무화하더라도 트래픽 패턴과 이벤트 사이에 직접적인 상관관계가 있을 수도 있습니다. 데이터가 어떤 물리적 이벤트에 대응하여 늘어나면 물리적 이벤트과 데이터 간에 실제로 상관관계가 나타나기도 합니다.

그러면 공격자가 신호 패턴을 모니터링하여 패넌에서 의미를 도출할 수도 있습니다. 이것은 공격자가 평문 데이터에 직접 액세스를 하지 못해도 가능합니다.

사용자가 특정 방에 있을 때 반응하는 홈 오토메이션 기술이 한 예입니다. 통신 시스템을 원격으로 모니터링할 수 있는 공격자라면 IoT 엔드포인트와 게이트웨이, 백엔드 시스템 간 통신의 패턴만 보고 어떤 집에 몇 명이 있는지, 집안 어디에 있는지, 사용자는 누구인지 알 수도 있습니다.

또 여럿이 있는 집과 혼자 있는 집을 쉽게 구별하고 집안 어디에 있는지도 알아낼 수 있을지 모릅니다. 보험사와 사법 당국에서는 이것이 집에 살고 있는 사람들에게 어떻게 더 위험한 상황을 만들지 알아야 할 것입니다.

이와 같은 위험은 대처하기가 쉽지만은 않습니다. 가장 흔하면서도 강력한 수단은 샘플을 채취할 사용자가 있든 없든 미리 정해진 속도로 샘플을 보내는 것입니다. 기밀유지와 무결성을 의무화하여 원격 공격자가 데이터의 평문을 평가하지 못하게 한다면, 외부인이 사용자의 활동이 들어 있는 샘플과 빈 샘플을 구별할 방법은 없게 됩니다.

하지만 이 모델에는 문제점이 있습니다. 스펙트럼 포화의 증가, 저전력 또는 배터리 기반 기술의 전력 소모 증가, 빈 샘플 패킷을 해독하고 검증하고 해석하는 데 필요한 프로세스 수준의 상승 등이 그것입니다.

대안으로, 샘플을 무작위 간격으로, 또 여러 가지 버스트로 보내는 것이 있습니다. 이런 유형의 패턴은 덜 비싸고 전력도 덜 소모하며 처리 전력도 덜 필요합니다. 그러나 사용자의 존재를 나타내는 미묘한 변화는 간파할 수 있을지도 모릅니다. 예컨대 진정한 엔트로피 시스템은 어느 것이나 100% 랜덤이고 예측이 불가능하지만, 사용자의 거동은 충분히 예측 가능합니다. 사용자가 방에 들어오고 방 안에 있는 센서가 반응을 해 네트워크상에 있는 피어 IoT 엔드포인트에게 데이터를 보내기 시작한다면 일관된 행동의 *시작*이 사용자의 존재를 나타낼 수도 있습니다.

이런 류의 위험이 따르는 기술을 개발하는 팀에서는 프라이버시 노출의 잠재적 영향을 조사하고 법무 팀과 협의하여 해당 기술이 회사의 법적 지위나 보험 모델에 영향을 미칠지 판단해야 합니다.

9.5.1 위험

IoT 서비스 업체가 프라이버시 노출과 보안 위험의 가능성이라는 관점에서 자사 기술을 평가하지 않는다면 아키텍처를 큰 폭으로 정비하여 해결이 필요한 위험을 보상*해야 할*지도 모릅니다. 나중에 큰 돈을 들여 아키텍처를 조정하려 하기보다 엔지니어링 단계에서 또는 최대한 일찍 위와 같은 솔루션을 제품에 삽입해야 합니다.

9.6 집속 이온빔과 X 레이에 대한 대처

FIB(Focused Ion Beam)는 반도체 평가에 자주 쓰이는 제조 기기입니다. 이 기술은 나노미터 단위로 회로를 검사하고 변경할 수 있어 제조 공정의 오류를 찾아내고 제조 공정을 바꾸기 전에 회로 패치를 테스트하기에 좋습니다.

정보 보안에서 FIB 는 내부 버스를 태핑(tapping)하여 내부 구성요소끼리 주고받는 데이터를 가로채고자 할 때 쓸 수 있습니다. 또 공격자가 보안 제약을 우회할 수 있도록 내부 회로와 내부 구성요소의 작동 방식을 바꿀 때에도 쓸 수 있습니다.

거의 모든 장치가 FIB 의 공격 대상이 될 수 있습니다. 그러나 일부 장치만이 FIB 프로세스를 통해 실행됩니다. 이것은 FIB 자체가 개당 100 만 달러에 달하는 지극히 비싼 기술이기 때문입니다. 이렇게 비싸기 때문에 툴킷에 이런 장비를 갖추고 있는 조직은 거의 없습니다. 이 장치는 또 자동이 아닙니다. 고도의 조작 기술과 반도체 분석에 관한 높은 전문성이 있어야 사용할 수 있습니다. 그러므로 FIB 의 실제 비용은 백만 달러가 훌쩍 넘으며 유틸리티와 교육, 급여, 사용자 교육까지 포함하면 수백 만 달러에 달합니다.

그러나 외주를 전문으로 하는 업체들이 있습니다. 역설계는 상당 부분 합법이므로 이들 업체는 장치의 역설계를 원하는 고객에게 반도체 공격 서비스를 제공합니다. 비용은 특정 구성요소의 공격에 필요한 커스터마이징 수준과 전문성에 따라 1 만 달러에서 100 만 달러 사이입니다. 예를 들어, 아웃소싱 회사라면 어떤 칩의 보호장치를 우회할 수 있는 *플레이북*은 있겠지만, 특이한 보안 잠금 기술이 딸린 커스텀 FPGA 솔루션은 기존 *플레이북*이 없어 훨씬 더 큰 비용이 들 것입니다. FIB 를 제대로 이용하려면 새 프로세스가 필요해 막대한 시간과 돈을 요할 것입니다.

최신 트러스트 앵커의 변종을 비롯한 일부 신기술은 FIB 탐지에 강하다고 주장합니다. 그런 주장에 어느 정도 타당한 측면도 있지만 동적이지 않은 하드웨어 보호장치(대부분이 그렇지 않음)는 어느 것이든 우회 기법 분석에 충분한 시간만 들인다면 *플레이북*이 나오기 마련입니다. 그러므로 그런 주장은 *일정 시간*까지만 유효할 가능성 높습니다.

따라서, 이와 같이 침습적이지만 거의 항상 성공하는 공격 기술에 대처하려면 엔지니어링 팀에서 트러스트 앵커에만 전적으로 의존하지 않는 보안 전략을 설계하는 것이 중요합니다. 대신, 그 기술을 기본 트러스트 앵커로 이용하지만 각 엔드포인트의 크립토그래픽 키를 개인화하는 프로토콜을 설계하여 어느 한 장치의 침해가 엔드포인트 전체 네트워크의 침해로 이어지지 않게 해야 합니다.

공격자가 *목표로 삼은 엔드포인트마다* FIB 를 이용해 크립토그래픽을 추출해야 하는 상황을 생각해 봅시다. 그러면 비용은 머지않아 크게 늘어나 어느 공격자든 큰 부담을 느끼고 말 것입니다. 이들 공격 방법은 *완전히 막아내기* 불가능하므로 *가치를 떨어뜨려* 숨김(obscurity)이 아니라 아키텍처를 통해 위험을 낮춰야 합니다.

9.6.1 위험

FIB 의 위험은 크립토그래픽 비밀과 그 외 지적재산이 구성요소, 심지어 보안이 철저한 구성요소에서 추출 당할 수 있다는 것입니다. 고객 IoT 에게 비용 대비 효과가 좋은 방법으로 FIB 를 격퇴하기는 현실적으로 쉽지 않기 때문에 조직에서는 엔드포인트 시스템의 보호 전략을 바꾸거나 엔드포인트 생태계 완전히 침해를 당할 위험을 감수해야 합니다.

9.7 공급망 보안 검토

어느 컴퓨팅 시스템이든 보안의 시작은 회로 기판을 구성하고 있는 원시 구성요소입니다. 실리콘과 크립토그래픽 토큰, 읽기 전용 메모리(ROM), 펌웨어, 그 외 임베디드 시스템의 핵심 속성 모두 그 시스템의 보안에 기여합니다. 이들 구성요소 중에서 어느 하나라도 탬퍼링 당하게 되면 전체 시스템이 보안 침해를 당할 수도 있습니다.

그러므로 보안을 중시하는 IoT 시스템 업체라면 구성요소의 출처와 조립, 조립된 기술을 출하하는 실행 프로세스를 고려해야 합니다. 기술을 생성하는 프로세스가 면밀하게 계획되지 않으면 프로세스 내 한 지점의 오류가 치명적인 보안 오류로 이어질 수도 있습니다.

다음 사항을 검토하기 바랍니다.

- 실리콘은 어디서 누가 만들었는가?
- 실리콘 설계를 믿을 만한 3 자 정보 보안팀에서 분석하였는가?
- 실리콘이 보안 시설에서 제조될 것인가?
- EEPROM 이나 NVRAM 에 부트로더 같은 실행가능 이미지를 어떻게 채울 것인가?

- 실행가능 이미지를 플래시하는 프로세스는 안전한가?
- 실행가능 이미지는 제조사에게 어떻게 전달되는가?
- 실행가능 이미지가 EEPROM 이나 NVRAM 에 플래시되고 나면 검증을 받는가?
- 크립토그래픽 비밀은 칩에 어떻게 프로비저닝되는가?
- 비밀을 제조사에서 만든다면 키 생성 시 검증된 RNG 를 이용하고 있는가?
- 보안 키는 TCB 권고사항에 따라 고유한가?
- 크립토그래픽 비밀은 IoT 서비스 업체와 어떻게 공유되는가? 보안은 철저한가?
- 고유한 칩 식별자(일련번호 등)는 어떻게 크립토그래픽 비밀과 연계되고 IoT 서비스 업체와 공유되는가?

제품을 만들고 조립하는 보안 시설을 늘리면 비용이 늘어날 수도 있지만 조직에게는 중요한 조치가 될 수도 있습니다. 이것은 제품의 용도와 도입 환경, 고객, 그 외 사람의 안전이나 군용, 주요 인프라 등과 같은 요소에 따라 달라집니다. 도출되는 기술이 사람의 생명에 영향을 미치기도 한다면 공급망에 보안의 허점은 없는지 확인해야 합니다.

9.7.1 위험

공급망 보안이 없다면 조직은 여러 가지 위험에 놓이게 됩니다. 그 중에는 다음과 같이 예상하기 어렵지만 회사에게 치명인 것도 있을 수 있습니다.

- 엔드포인트 복제(불법 제조)
- 기술의 도용(경쟁사가 서비스 업체에게서 훔쳐 서비스 업체를 상대로 이용하는 것)
- 자격증명 도용(데이터 가로채기 또는 사칭 공격)
- 임플란트(나중에 활동을 시작할 수도 있는 악성 "백도어")의 주입

9.8 적법한 가로채기

정당한 가로채기란 고객과 서비스 업체 간의 통신을 합법적으로 가로채거나 조작하는 행위를 말합니다. 이것은 둘 중 한 가지 방법으로 가능합니다. 하나는 가장 일반적인 시나리오로 사법기관에서 통신 사업자에게 적법한 요청서를 제출하여 특정 가입자의 통신에서 나온

메타데이터나 실제 데이터에 액세스하는 것입니다. 다른 하나는 사법기관에서 IoT 서비스 업체에게 특정 가입자의 데이터나 메타데이터에 대한 액세스를 요청하는 것입니다. 기관에서 사업자를 통해 액세스를 요청할 경우, IoT 서비스 업체는 법적 요청의 범위에 따라 다르겠지만 문제가 있어도 통보를 전혀 받지 못할 수도 있습니다. 따라서 서비스 업체는 해당 기관의 법적 요청을 시행하거나 따를 준비가 돼 있어야 합니다.

그러므로 서비스 업체는 법 집행 요청에서 어떤 프라이버시 문제가 나타날 수 있는지 확인하고 법적 권한 안에서 조직의 법적 모델 및 프라이버시 정책과 관련된 정보를 제공할 준비를 해둬야 합니다.

얼마 전 구글, 애플 등 대기업에서는 회사가 비밀 요청을 받으면 사용자에게 기관을 대신해 합법적으로 알려주는 *영장 카나리아*를 도입했습니다. 회사는 적법한 사법기관과 접촉하*지 않았음*을 나타내는 문구나 이미지 등을 뺄 수 있습니다. 그런 문구나 이미지를 뺀다는 것은 물론 비밀 요청이 있었다는 의미입니다.

9.8.1 위험

기업이 적법한 가로채기 요청에 대비하지 않았을 때 실제로 그런 요청을 받으면 불리한 위치에 놓일 수도 있습니다. 요청을 수락해야 하지만 법적 인프라나 프라이버시 정책이 없어 위험에 처할 가능성도 있습니다.

엔드포인트 프로토콜과 IoT 플랫폼에 대해 기밀유지와 무결성을 확보하지 않으면 회사도 모르는 사이 네트워크 밖에서 통신이 가로채기를 당하게 됩니다. 이 경우 회사는 사용자의 데이터가 유출되거나 스노우든 NSA 유출과 같은 사건에 연루돼 사용자 데이터 보안에 대한 대중의 신뢰를 크게 잃을 수도 있습니다.

10 요약

요약컨대, IoT 제품이나 서비스에 상존하는 보안 위험은 잘 정의된 아키텍처와 보안 관련 사건 전후에 위험을 찾아내는 정보력, 그리고 사건을 처리하는 정책과 절차만 있다면 거의 다 대처할 수 있습니다. IoT 서비스 업체에 어떤 높은 수준의 보안 개념이 중요한지 분석해 자주하는 질문을 참고하면 좋습니다. 이것은 엔지니어링 팀이 보안 아키텍처의 허점을 해결하고자 할 때 가장 시급한 권고사항을 찾는 데 길잡이가 될 수 있습니다.

팀이 아키텍처의 정의에서 진전을 보이면
독립적인 권고사항도 검토할 수 있게 됩니다. 보안상의 의문점과 우려사항이 구현 과정에서 점점 더 두드러지기 때문입니다.

전체적으로, 엔지니어링 팀은 거의 다 매우 비슷한 위험에 놓이게 됩니다. 조직이 다른 피어와 우려사항을 공유하여 위험과 대처 전략 모두에 대해 공통의 지식베이스를 구축하는 것이 중요합니다. 여러 조직이 함께 하면 기술과 지식을 모두 축적해 서로 도와가며 IoT 의 미래에 보안을 든든히 할 수 있습니다.

부록A 범용 부트스트랩 아키텍처의 활용 사례

다중 홉(hot) 네트워크의 전반적인 보안 수준은 체인 내 가장 약한 링크로 판가름됩니다. 그러므로 IoT 엔드포인트와 게이트웨이 간의 로컬 링크에 광역 네크워크와 대등한 수준의 보안을 적용하여 보안 수준을 맞춰야 합니다.

이것을 실현할 수 있는 기술 가운데 하나가 범용 부트스트랩 아키텍처(GBA)[17]입니다. 이것은 인증은 물론 데이터 무결성에도 사용 가능합니다. 이것은 미리 공유된 키를 기반으로 하는데, 이들 키는 이어서 인증과 암호화의 근간으로서 시간 제한 키(토큰)의 생성에도 사용됩니다.

인증은 누군가 또는 어떤 것이 정말로 자기가 주장하는 그 사람 또는 그것이 맞는지 판단하는 과정입니다. 수십 억 엔드포인트가 활동하는 IoT 공간에서 어떤 통신 거동이 진짜이고 참인지 가려내는 것은 매우 중요합니다. 이와 같은 신뢰 관계를 만들기 위해 구축된 메커니즘은 확장성과 유지능력이라는 요건을 충족해야 합니다. 또한 IoT 서비스는 다양하므로 인증 메커니즘은 다른 서비스까지 수용하면서도 공통의 인프라를 유지할 수 있는 유연성을 갖춰야 합니다. 오랜 시간을 두고 검증된 메커니즘 가운데 하나가 SIM 기반의 네트워크 인증입니다. 이 인증 인프라는 인증이 가능할 뿐만 아니라 미리 공유된 비밀을 이용해 암호화도 가능합니다. 엔드포인트의 수가 급증하고 IoT 서비스가 전 세계로 확대되면서 SIM 의 활용이 제한되고 있습니다. 이유는 네트워크 로밍과 아무도 지켜보지 않는 엔드포인트에서 실물 SIM 을 빼낼 수 있다는 보안상의 취약점 때문입니다. 임베디드 SIM 과 같은 기술의 등장으로 사전 공유 비밀을 기반으로 한 인증이 사용될 기반이 생겼고 그로 인해 현행 SIM 기반 네트워크 인증이 확장되고 있습니다. 또한 모세 네트워크(본 문서 앞 섹션의 구성 사례 2, 3, 4)의 형태로 IoT 의 성장이 일어날 가능성이 매우 높아졌습니다. 여기서 모세 네트워크는 게이트웨이에 연결된 무수한 엔드포인트를 말합니다. 이들 엔드포인트 장치 대부분은 경량 엔드포인트 장치(SIM 이나 셀룰러 연결이 없는 것)가 될 것입니다. 그렇지만 이 경량 엔드포인트들에게도 인증과 암호화 능력이 필요합니다. 모세 네트워크에서 인증의 1 차 책임은 게이트웨이에게 있습니다. 네트워크에 복잡한 SIM 기반 엔드포인트 장치의 수를 줄이기 위함입니다. 이 인증과 보안은 게이트웨이에서 엔드포인트 장치까지 이어져 특정 엔드포인트 장치부터 IoT 서비스 플랫폼까지 안전한 채널을 만들어야 합니다.

SIM 기반 인증은 용도가 한 가지뿐입니다. 즉 네트워크에 어떤 엔드포인트가 하나뿐임을 인증하는 것입니다. 엔드포인트 장치는 여러 가지 서비스를 담당하게 되는데, 각 서비스가 인증 요건이 다릅니다. 네트워크 인증을 복수의 서비스로 확대하는 프레임워크가 필요합니다. 이

목적을 위해 설계된 프레임워크 가운데 하나가 GBA(Generic Bootstrapping Architecture)입니다. GBA 는 SIM 기반 인프라를 이용해 장치와 NAF(Network Application Functions) 사이에 시간 중심의 공유 키를 만듭니다. GBA 는 3GPP 가 3GPP 규격 TS 33.220[17]에서 표준화한 인증 방법입니다. 이 방법으로 3GPP 에 가입된 장치를 서비스에 인증할 수 있습니다. 가입의 자격증명은 장치 안에, 대개 UICC 같은 SIM 에 저장됩니다. 원격으로 관리되는 자격증명은 예컨대 GSMA 에서 정한 임베디드 SIM(eUICC)[5] 등에 저장돼 관리됩니다.

이 프레임워크의 장점은 다음과 같습니다.

- 장치와 네트워크 응용 기능(NAF) 간의 PSK 를 이용하거나 인증 기반 NAF 인증(TS 33.222)[18]이 포함된 공유 키 기반 UE 인증을 이용한 상호 인증.
- 자격 증명을 트러스트 환경에 안전하게 보관 가능
- eUICC 사용 시 자격증명을 OTA 로 바꿀 수 있습니다.
- 확장성. 유지보수의 복잡성과 비용은 장치의 수에 비례해 증가합니다. 프레임워크 안에 인증이 "내장" 되기 때문입니다.
- 데이터 무결성 인증 시에 시간 기반으로 생성된 키는 TLS-PSK 터널 구축에 쓸 수 있습니다. 이 연결을 만들어지면 아주 강력한 데이터 무결성과 기밀유지가 가능합니다.

부록 B IoT 서비스 내 UICC 카드 사용법

ETSI TS 102 221 으로 표준화된 UICC 는 스마트카드 플랫폼으로서 UICC 호스트 장치에게 상호운용성을 갖춘 보안 파일 시스템 인터페이스와 보안 애플리케이션 프레임워크를 제공합니다. ETSI TS 102 221 은 UICC 호스팅 장치가 UICC 에서 관련 애플리케이션을 발견하는 프레임워크 역할을 합니다. 각 UICC 애플리케이션은 알고 있는 프로비저닝과 구성 정보뿐만 아니라
니즈에 따라 호스팅 장치가 지원할 수 있는 운영 절차(인증 또는 키 도출)에도 부합합니다.

IoT 환경에서 UICC 는 ETSI TS 102 671 에 기술된 바와 같이 여러 가지 폼팩터와 환경 운영 범위에서 이용 가능합니다. 가장 단순한 형태에서는 UICC 가 네트워크 사업자에게 소속돼 네트워크 액세스 애플리케이션(3GPP TS 51.011 에 따른 SIM 애플리케이션, 3GPP TS 31.102 에 따른 USIM, 3GPP2, WiMAX SIM 에 따른 CDMA CSIM 등)을 하나만 호스팅합니다. 이 경우 UICC 는 모바일 장치에서 보안 프로비저닝과 구성 정보, 크립토그래픽 절차를 호스팅하여 네트워크 액세스를 실현할 수 있는 표준화된 홀더를 제공합니다. 이때 UICC 의 콘텐츠를 ETSI TS 102 225 / TS 102 226 를 이용해 원격으로 관리할 수 있는 메커니즘이 추가로 존재합니다. 모바일 네트워크 생태계에는 네트워크 사업자의 통제를 받는 UICC 를 안전하게 개인화하고 배포하기 위한 절차가 마련돼 있습니다. 그 결과 UICC 호스팅 장치와 인프라 간에 개별 공유 대칭 키가 만들어집니다.

UICC 플랫폼의 중요한 기능 한 가지는 복잡한 생태계에 존재하는 복수의 이해관계자가 UICC 에서 자기 영역을 배정 받고 다른 이해관계자 모르게 자기 콘텐츠를 관리할 수 있는 격리된 보안 도메인을 지원하는 것입니다. 이 기능은 GlobalPlatform Card Specification [15] Amendment A 에서 ETSI TS 102 226 를 통해 물려 받은 것입니다. 그러므로 IoT 환경에는 UICC 하나를 통해 복수의 이해관계자가 남과 독립적으로 자기의 자격증명을 저장하고 관리할 수 있는 것입니다.

일반적으로 UICC 하나가 몇 가지 네트워크 액세스 애플리케이션을 수용할 수 있으며(활동은 한 번에 하나씩만 가능함), 그 외에 잠재적으로 IMS 액세스를 위한 ISIM 애플리케이션(3GPP TS 31.103 에 명시)이나 oneM2M TS-0003 의 부록 D 에 명시된 1M2M SM 애플리케이션(IoT 서비스일 경우)처럼 더 정교한 서비스에도 액세스를 제공하는 다른 애플리케이션도 수용할 수 있습니다. 1M2MSM 애플리케이션은 전용 IoT 서비스/애플리케이션 자격증명의 직접 프로비저닝을 지원하며 UICC 에서 3GP 가 지정한 GBA 메커니즘을 이용해 기존 네트워크

액세스의 자격증명을 도출하는 것도 지원합니다. 또한 IoT 서비스 업체가 필요(예: 특정 서비스 인증 메커니즘 지원)에 따라 크립토그래픽 절차를 커스터마이징하는 것도 지원합니다.

UICC 도 복수의 1M2MSM 애플리케이션을 수용해 IoT 서비스 업체별로 전용 대칭 키가 안전하게 배포되게 합니다. UICC 소유자(IoT 환경에서 일반적으로 네트워크 사업자 또는 OEM 제조사)는 요청하는 IoT 서비스 업체와 UICC 의 공간을 공유하여 네트워크 액세스 자격증명의 안전한 배포가 가능한 인정 UICC 개인화 체인과 인프라를 IoT 서비스 업체가 자기의 자격증명을 배포하는 데 이용하게 할 수도 있습니다.

IoT 애플리케이션의 보안이 비대칭 크립토그래피에 의존하고 있을 경우, 커스텀 UICC 애플리케이션을 유사한 방식으로 이용해 특정 IoT 서비스의 니즈에 맞게 공용/사설 키 쌍의 배포를 촉진할 수도 있습니다. 그와 같은 UICC 애플리케이션은 IoT 애플리케이션 별로 호스팅 장치에서 지정하고 지원해야 합니다.

부록 C 문서 관리

C.1 문서 이력

버전	날짜	변경 내용	승인권자	편집자 / 회사
1.0	2016-02-08	New PRD CLP.13	PSMC	Ian Smith GSMA & Don A. Bailey Lab Mouse Security
1.1	2016-11-07	GSMA IoT 보안 평가 제도에 대한 언급 추가됨. 사소한 편집 교정	PSMC	Ian Smith GSMA
2.0	2017-09-29	GSMA LPWA 네트워크 인용 추가, 사소한 내용 수정	IoT Security Group	Rob Childs GSMA

C.2 기타 정보

유형	설명
문서 담당자	GSMA IoT Programme
연락처	Rob Childs - GSMA

당 기관은 문서 품질을 중시합니다. 오류 또는 누락 발견 시 의견과 함께 연락 바랍니다.

prd@gsma.com 으로도 연락할 수 있습니다

의견, 제언, 질문은 언제든지 환영합니다.