IoT 서비스 생태계를 위한
IoT 보안 지침

IoT 서비스 생태계를 위한 IoT 보안 지침

버전 2.0

2017 년 10 월 31 일

본 문서는 구속력이 없는 GSMA 의 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

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 지침서 중 그 첫 번째 부분입니다. 이 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 지침서 세트의 구성은 다음과 같습니다. 본 CLP.11 IoT 보안 지침 개요서 [1]는 부속 문서를 읽기 전에 일종의 예비서로 읽는 것이 좋습니다.

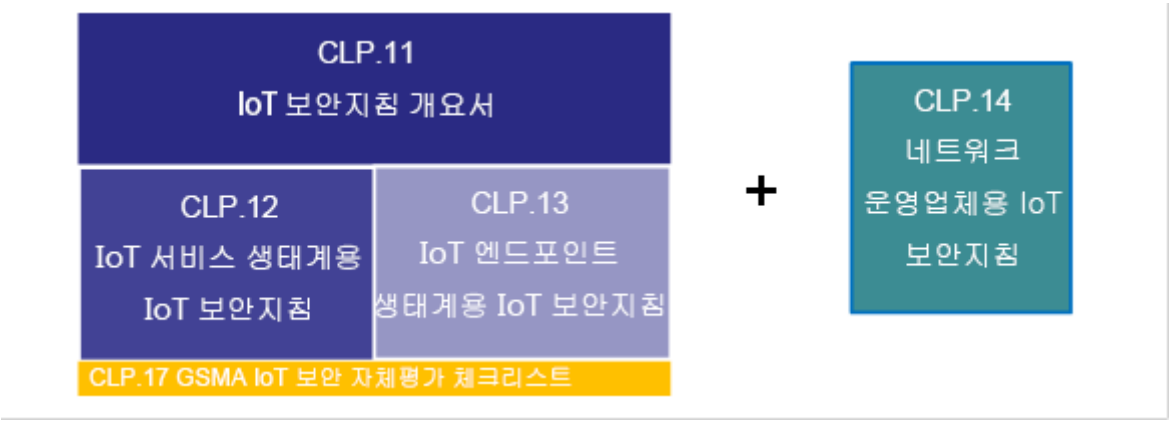

그림 1 - GSMA IoT 보안 지침서 구조

네트워크 운영사, IoT 서비스 업체와 IoT 생태계의 기타 협력사들은 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 [13]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[13]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 지침서는 서비스 생태계의 관점에서 IoT 제품이나 서비스의 컴포넌트 전체를 평가하는 목적으로 사용해야 합니다. 서비스 생태계는 IoT 인프라의 핵심을 이루는 컴포넌트를 모두 포괄합니다. 이 생태계의 컴포넌트로는 예컨대 서비스와 서버, 데이터베이스 클러스터, 네트워크 요소와 그 외 제품이나 서비스의 내부 컴포넌트를 구동하는 기술 등이 있습니다.

본 문서의 범위는 IoT 서비스, 네트워크 요소의 설계와 구현에 관한 제언으로 한정됩니다.

본 문서에서는 IoT 사양이나 표준의 개발을 제안하지 않으며 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 엔드포인트 디바이스 제조업체 - IoT 엔드포인트 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 제어 목록	컴퓨팅 객체에 첨부되는 허가 목록
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.
클라우드	애플리케이션과 그 데이터를 호스팅하고 저장, 관리, 처리하는 인터넷상의 원격 서버 네트워크.
컨테이너	복수의 격리된 시스템, 즉 컨테이너를 한 호스트에서 실행할 수 있게 만드는 기술.
임베디드 UICC (eUICC)	네트워크의 원격 프로비저닝 또는 인증 받은 서비스 구독을 GSMA 에서 지정한 대로 지원하는 UICC.
최종 고객	IoT 서비스 업체에서 제공하는 IoT 서비스의 고객. 최종 고객과 IoT 서비스 업체가 같을 수도 있습니다. 전기회사가 그 예입니다.
엔드포인트 생태계	현실 세계와 디지털 세계를 색다르게 연결하는 저 복잡도 디바이스와 리치 디바이스, 게이트웨이의 구성. 자세한 내용은 CLP.11 [1]를 참고하십시오.
순방향 보안	보안 통신 프로토콜의 속성: 롱텀 키가 손상되어도 과거 세션 키가 손상되지 않는다면 보안 통신 프로토콜에 순방향 보안이 있다고 합니다.
사물 인터넷	복수의 기계와 디바이스, 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 엔드포인트	복잡한 IoT 엔드포인트 장치 또는 IoT 게이트웨이 장치를 이르는 일반적인 용어.
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.

용어	설명
IoT 서비스 업체	새롭고 혁신적인 커넥티드 제품과 서비스를 개발하고자 하는 기업이나 조직.
네트워크 사업자	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.
보안 그룹	하나 이상 가상 서버 인스턴스의 트래픽을 제어하는 가상 방화벽 역할을 합니다.
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 주고 받은 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.
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.
가상 사설 네트워크	특정 고객 서비스만 사용할 수 있도록 네트워크에서 안전하게 논리적으로 분리된 부분. VPN 이 네트워크 나머지와 격리돼 그 자체로 가상화 네트워크처럼 작동한다 하여 이런 이름이 붙었습니다

1.5 약어

용어	설명
3GPP	3 세대 프로젝트 파트너십
ACL	액세스 제어 목록
API	응용 프로그램 인터페이스
APN	액세스 포인트 이름
CERTS	컴퓨터 비상 대응팀
CLP	GSMA 커넥티드 리빙 프로그램

용어	설명
DDoS	분산 서비스 거부 공격
GSMA	GSM 협회
HSM	하드웨어 보안 모듈
IoT	사물 인터넷
IP	인터넷 프로토콜
SQL	구조화된 쿼리 언어
TCB	신뢰 컴퓨팅 기반
VM	가상 머신
VPN	가상 사설 네트워크
WAF	웹 애플리케이션 방화벽

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]	해당 없음	OWASP 보안 애플리케이션 설계 프로젝트(OWASP Secure Application Design Project) https://www.owasp.org
[6]	해당 없음	TCG 신뢰 플랫폼 모듈(TCG Trusted Platform Module) http://www.trustedcomputinggroup.org
[7]	해당 없음	TCG IoT 보안 지침(TCG Guidance for Securing IoT) http://www.trustedcomputinggroup.org
[8]	해당 없음	OAuth 2.0 http://oauth.net/2/
[9]		OpenID 파운데이션 http://openid.net/foundation/

참고	문서 번호	제목
[10]	해당 없음	GSMA 모바일 커넥트(GSMA Mobile Connect) https://mobileconnect.io/
[11]	GPC_SPE_034	글로벌플랫폼 카드 규격(GlobalPlatform Card Specification) www.globalplatform.org/specificationscard.asp
[12]	GPD_SPE_010	글로벌플랫폼 TEE 내부 코어 API 규격(GlobalPlatform TEE Internal Core API Specification) www.globalplatform.org/specificationsdevice.asp
[13]	CLP.17	GSMA IoT 보안 평가 체크리스트 https://www.gsma.com/iot/iot-security-assessment/
[14]	해당 없음	ETSI TC SmartM2M 규격(ETSI TC SmartM2M specifications) www.etsi.org
[15]	해당 없음	oneM2M 사양(oneM2M Specifications) www.onem2m.org
[16]	3GPP TS 33.220	일반 인증 아키텍처(Generic Authentication Architecture, GAA); 범용 부트스트래핑 아키텍처(Generic Bootstrapping Architecture, GBA) www.3gpp.org

2 서비스 모델

요즘 IoT 제품과 서비스는 서비스 생태계가 있어야 엔드포인트와 파트너, 사용자에게 의미와 기능, 가치를 제공할 수 있습니다. IoT 제품과 서비스가 제공하는 애플리케이션이 복잡하면 인프라도 커지고 여러 가지 이질적인 서비스와 서비스 액세스 포인트로 구성되기도 합니다. 반대로, 애플리케이션이 단순 명료하면 인프라도 기분적 수준이 될 수 있습니다.

서비스 생태계는 그 형식이 어떻든 전체 IoT 기술의 핵심 측면 각각에 대해 기능과 통신의 결합체 역할을 합니다. 그 외 생태계는 모두 서비스 생태계를 통해 IoT 의 일상적 운영에 없어서는 안 될 계층 인증과 사용자 연결, 가용성, 관리 등을 수행합니다. 서비스 생태계는 이 작업의 완수를 위해서 인프라의 목표 달성에 필요한 수만큼 계층(tier)을 갖고 있습니다. 데이터베이스 클러스터, 애플리케이션 서버, 애플리케이션 프록시 서버 등이 이 같은 계층의 예로서 많은 배포에서 발견됩니다. 아래 다이어그램에서 짐작할 수 있듯 네트워크와 엔드포인트 생태계는 서비스 생태계의 핵심 기능에 의존하고 있습니다.

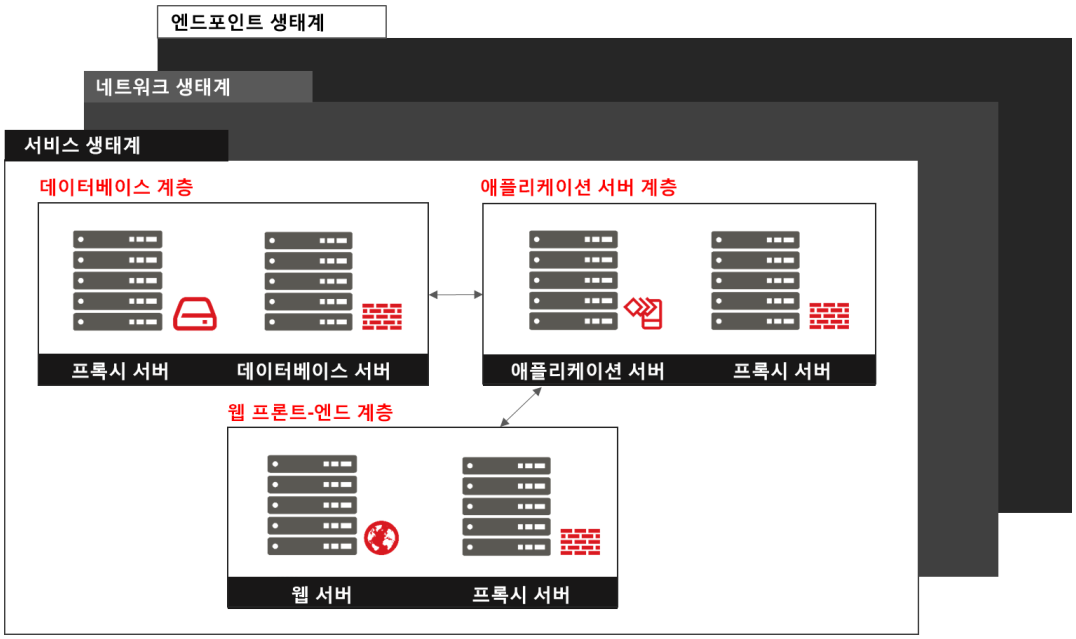

그림 2 - 서비스 생태계에 연동된 종속요소

요즘 나오는 서비스 생태계의 예를 들면 다음과 같습니다.

- 클라우드 인프라 기반 솔루션
- 컨테이너 기반 애플리케이션 배포
- 전통적인 데이터센터 서버 환경
- 데이터베이스 클러스터
- 웹 애플리케이션 프레임워크 서비스 클러스터

이 같은 환경은 그 설계와 토폴로지, 구현에 큰 편차가 있는 듯 보일지도 모르지만 정보가 애플리케이션에 들어오고 나가는 원리는 같습니다.

요즘 나오는 컴퓨팅 시스템은 모두 애플리케이션의 인프라에 진입하는 지점, 즉 서비스 액세스 포인트가 필요합니다. 그 시스템에 컨텐츠와 컨텍스트를 만드는 내부 서브시스템은 안전하고 믿을 수 있는 환경과 네트워크 안에서 데이터를 처리할 수 있어야 합니다. 데이터는 어딘가에 저장해 두었다가 같은 생태계 또는 다른 생태계와 그 관련 네트워크 안에 있는 여러 컴포넌트로 인가 받은 명령어를 회신하거나 전송하는 서비스 레이어로 다시 돌려보내야 합니다.

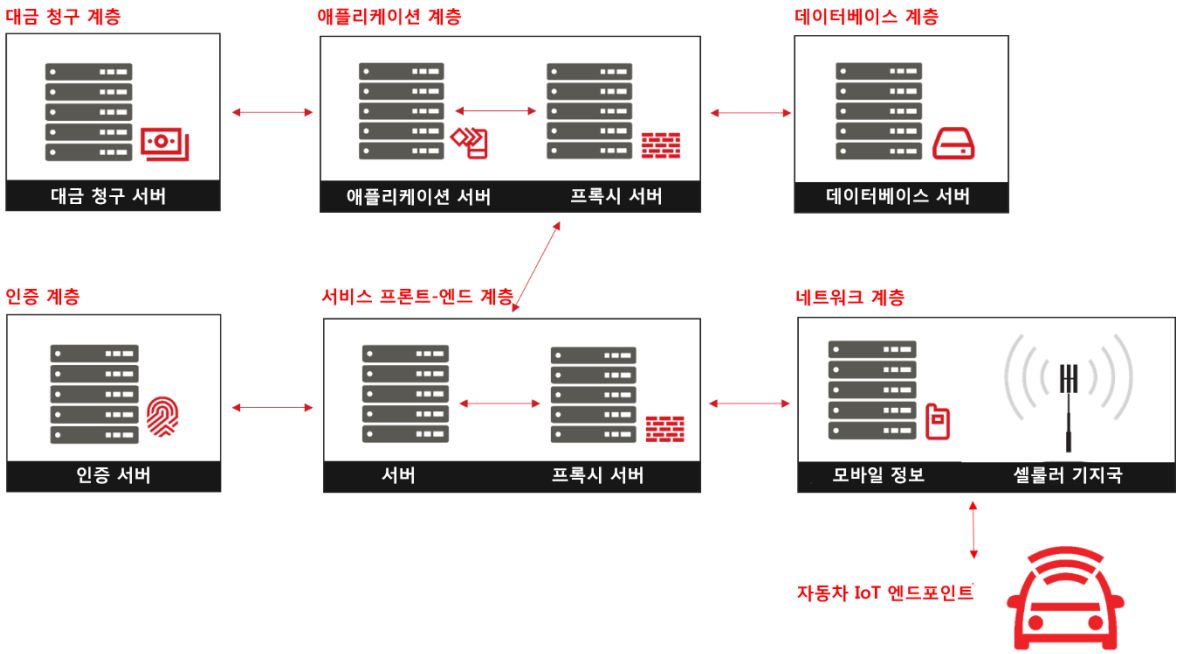

그림 3 - 서비스 생태계의 예

이 표준 프레임워크를 구현하기 위해 어떤 기술을 사용하든 정보는 검증된 프로토콜과 기술을 통해 처리되고 서비스되고 인증됩니다. 처리 환경의 토폴로지와 추상화는 요즘의 속도, 연산력, 저장 요건에 맞춰 조금씩 변했지만 이 같은 혁신을 구현하는 기술은 기본적인 측면에서 보면 동일합니다. 가령, 각 계층에는 특정 유형의 서버군과의 연결을 관장하는 프록시 또는 방화벽 시스템이 존재합니다. 대금 청구 서비스는 대금 청구 계층에 위치합니다. 애플리케이션 서버는 해당 애플리케이션만의 계층에 위치합니다. 데이터베이스 서비스는 데이터베이스 계층 안에서 관리해야 합니다. 이들 시스템은 프록시 서버 수준에서 적용되는 진입/진출 규칙에 따라 함께 작동합니다.

따라서, 서비스 생태계의 보안 모델은 일단의 컴포넌트로 쉽게 나눌 수 있습니다. 본 문서에서는 이 컴포넌트들을 살펴보겠습니다.

3 보안 모델

서비스 엔드포인트 모델에서 보안은 애플리케이션 아키텍처의 구축에 사용된 토폴로지나 혁신과 상관 없이 동일한 인프라와 전략, 정책을 이용해 설계할 수 있습니다. 서비스 생태계의 각 측면은 몇 가지 컴포넌트로 나눌 수 있습니다. 이들 컴포넌트는 따로, 하지만 유사한 방법으로 보안 확보가 가능합니다.

예컨대, 엔드포인트, 파트너, 사용자에게 질의를 받고 회신할 수 있는 간단한 서비스를 구축하고자 할 때 흔히 필요한 컴포넌트를 생각해 봅시다. 이 모델에는 다음과 같은 계층이 들어갈 것입니다.

- 웹 서비스 계층
- 애플리케이션 서버 계층
- 데이터베이스 계층
- 인증 계층
- 네트워크 계층
- 3 자 인증 계층 (대금 청구 계층 등)

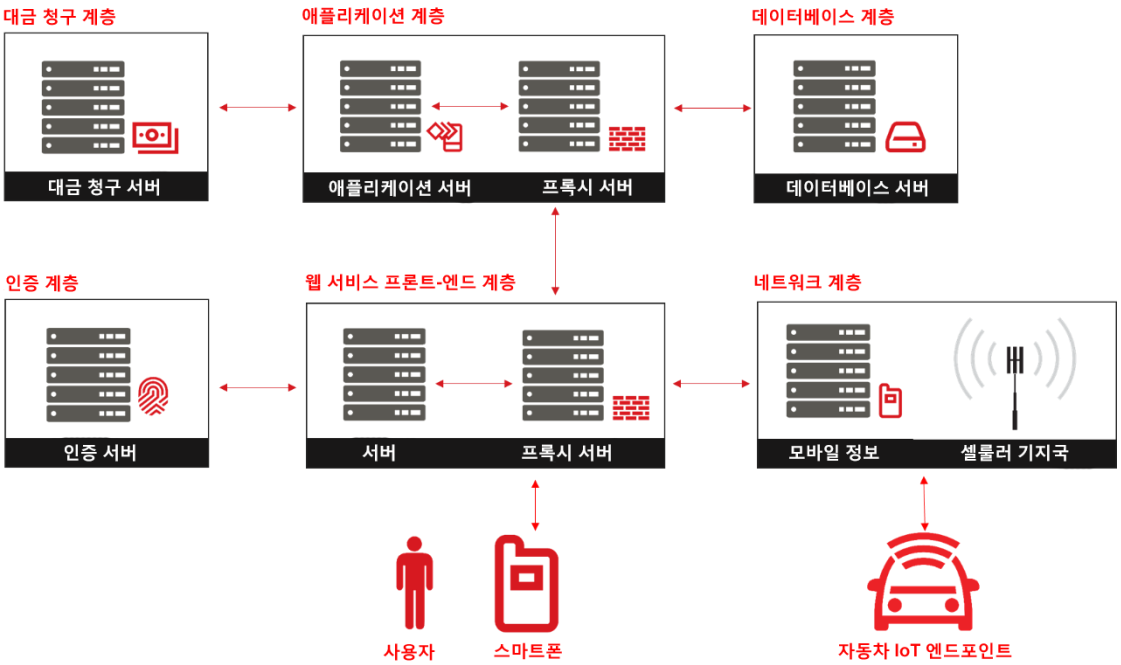

그림 4 - 별도 계층이 있는 서비스 생태계의 예

각 계층에 서버가 하나 밖에 없더라도 각 논리 개념을 자체 계층으로 분리하면 아키텍처의 효과가 높아집니다. 이렇게 되면 침해가 발생하거나 더 많은 요청에 대응하기 위해 시스템을 확장해야 할 때 한 기술 레이어를 다른 레이어와 격리하기 좋습니다.

어떤 유형의 시스템을 *계층의 유형*이라는 관점에서 생각하면 보안 확보와 주문형 확대, 퇴역, 일몰이 더 쉬워질 수도 있습니다. 계층의 수명이 다할 때까지 증강하거나 조정할 수 있을 만큼 다재다능한 API 만 있으면 됩니다. 이 같은 API 를 정의하는 일은 본 문서의 범위를 벗어납니다. 그러나 조직이 선택하거나 정의할 API 의 고급 보안 속성에 관한 권고사항은 여기서 논의할 것입니다.

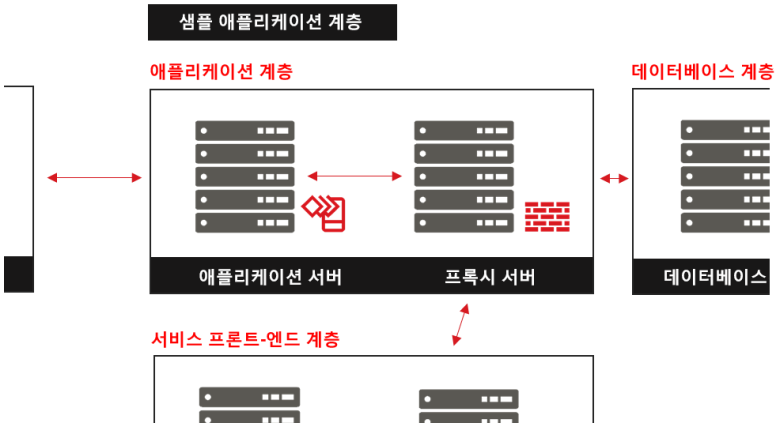

그림 5 - 방화벽 기술의 보호를 받는 애플리케이션 계층

위 예에서는 조금 더 완전한 계층을 설명했습니다. 이 계층을 묘사하는 데 필요한 증강으로는 프록시 서버가 유일합니다. 이 프록시 서버는 그 계층 안에 도입될 실제 보한 기술을 나타내는 디스크립터일 뿐입니다. 실제 제어장치가 하드웨어 방화벽이든, 소프트웨어 방화벽이든, 보안 그룹이든, 액세스 제어 목록(ACL), 그 외 다른 것이든, 계층을 대신해 진입과 진출 제어를 의무화하는 컴포넌트가 존재하게 됩니다.

조직에서 API 를 선택하거나 정의할 때에는 엔지니어링팀의 우려를 해소해 줄지도 모르는 기존 스펙을 검토해야 합니다. 특히 다음 스펙을 검토해야 합니다.

- ETSI SmartM2M TS 102 690, ETSI SmartM2M TS 102 921 [14]
- oneM2M TS-0001, oneM2M TS-0003 [15]
- 3GPP TS 33.220 [16]

서비스 프론트-엔드 계층처럼 공개적으로 접속 가능한 컴포넌트의 경우, 모델에게 필요한 증강은 다음을 위한 추가 보안 컴포넌트뿐입니다.

- DDos(Distributed Denial of Service) 차단
- 로드 밸런싱
- 리던던시
- 옵션 WAF(Web Application Firewall) 능력

위 기술들은 어떤 서비스든 제대로 작동하려면 구현해해야 합니다. 또 리소스가 제한된 환경에서도 보호 대상 서비스에 대해 가용 상태를 유지하기 위해서도 필요합니다. 이들 컴포넌트를 정의하는 것은 본 문서의 범위 밖이지만, 다음 엔터티들을 참조하면 더 자세히 알 수 있습니다.

- 클라우드 보안 얼라이언스
- NIST 클라우드 컴퓨팅 표준
- FedRAMP
- 시스코 네트워크 관리 가이드라인

계층이 안전하게 기능하는 데 필요한 그 외 속성으로는 서버 자체의 정의가 있습니다. 이것은 엔지니어링 팀이 선택한 플랫폼에 내재하는 관리, 응용, 운영 체제 제어장치에서 정의합니다.

전부는 아니지만, 이 플랫폼 환경에 내재하는 이슈는 다음과 같습니다.

- 중앙집중식 로그 서비스에 대한 로깅
- 관리 인증과 허가
- 통신 보안 단속
- 데이터 백업, 복원, 복제
- 애플리케이션 임무의 분리
- 시스템 모니터링과 통합

3.1 네트워킹 인프라에 대한 공격

네트워크의 관점에서 서비스 엔드포인트를 무력화하려고 하는 측에서는 엔터티들이 통신하는 방식에 약점이 있고 서비스 액세스 포인트를 통해 노출되는 서비스에 취약점이 있다고 가정합니다. 이 같은 공격에서는 네트워크상의 특별한 위치가 통신 채널 상에서 힘 있는 위치와 동일시 된다고 가정합니다.

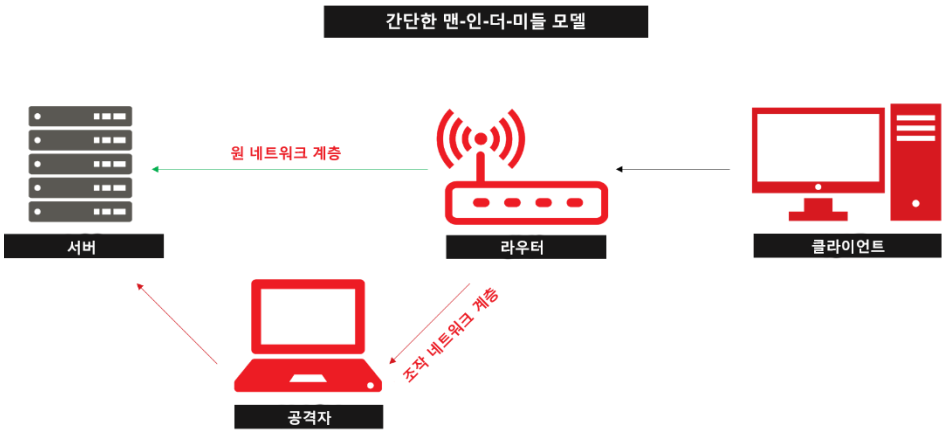

그림 6 - "중간자" 공격 모델의 예

이 모델에서 가장 흔한 형태의 공격은 중간자(Man-In-The-Middle, MITM) 공격입니다. 이 공격은 통신 채널에 피어 인증이나 일방향 피어 인증, 상호 인증 상실이 없다고 가정합니다. 공격자의 목표는 대화의 일방으로 위장하여 피어로 하여금 공격자를 대신해 행동을 하게 만드는 것입니다. 이 공격은 상호 인증으로 무력화할 수 있는데, 여기에는 잘 정의된 조직적 신뢰 기반과 신뢰 컴퓨팅 기반(TCB), 그리고 통신 모델이 하나 필요합니다.

다른 공격들은 예컨대 순방향 보안에 대한 공격, 암호화 통신 분석, 사이드 채널 공격 등입니다. 이것들은 적절한 크립토그래피 프로토콜과 알고리즘, 표준으로 무력화해야 합니다.

이들 공격은 어려우며 조직 내부에서, 조직과 그 파트너 또는 엔드포인트 생태계 간 핵심 인터넷 인프라 내부에서, 또는 엔드포인트와 가까운 인프라에서 네트워크 인프라에 접속해야만 가능합니다. 가장 단순하면서도 흔한 공격은 Wi-Fi 나 이더넷, 이동전화 네트워크 같은 엔드포인트의 네트워크 인프라를 조작해 서비스와 그 피어 간에 특권의 위치를 점하려고 시도하는 것입니다.

한 엔드포인트의 인프라에 대한 공격은 그 엔드포인트 또는 그 물리적 위치에서 이용 가능한 일단의 엔드포인트에 국한됩니다. 핵심 인터넷 인프라를 대상으로 한 공격은 일반적으로 BGP(Border Gateway Protocol) 하이재킹이나 핵심 라우터 공격, DNS 인프라 남용의 형태를 띱니다. 이들 공격은 특정 목표와 더 많이 분리된 특권의 위치를 제공해 접속권을 지닌 공격자가 한 번에 많은 시스템을 겨냥할 수 있게 합니다. 내부 네트워킹 인프라에 대한 공격은

내부 네트워크에 액세스할 수 있어야 가능합니다. 이것은 내부자 공격 또는 기업 환경 안에 특권의 존재를 암시하는 것으로 이미 시스템이 상당히 무력화되었다는 의미일 수도 있습니다.

어떤 공격 방식이 동원되든, 이 모델은 상호 인증과 순방향 보안, 적절한 크립토그래픽 프로토콜과 알고리즘으로 쉽게 차단할 수 있습니다. 이들 방법을 적용하면 공격자가 이 인프라를 남용하는 능력이 약화되거나 공격 비용이 크게 증가해 웬만한 공격자라면 실행을 단념하게 됩니다.

3.2 클라우드 또는 컨테이너 인프라 공격

이 공격은 클라우드 또는 컨테이너 인프라 환경에서 특권이 있다고 가정합니다. 예컨대, 어떤 공격자가 클라우드 서비스 네트워크를 침범할 수 있다면 게스트 가상 머신(VM) 시스템을 실행 중인 호스트에도 액세스할 수 있을지 모릅니다. 그렇게 되면 공격자가 VM 시스템의 실행을 검사하고 변경할 수 있게 됩니다. 이 공격자는 구체적인 목표를 갖고 있을 수도 있고 그저 값 나가는 데이터가 들어 있는 여러 시스템에 액세스할 목적으로 운 좋게 클라우드 서비스 업체를 공격했을 수도 있습니다.

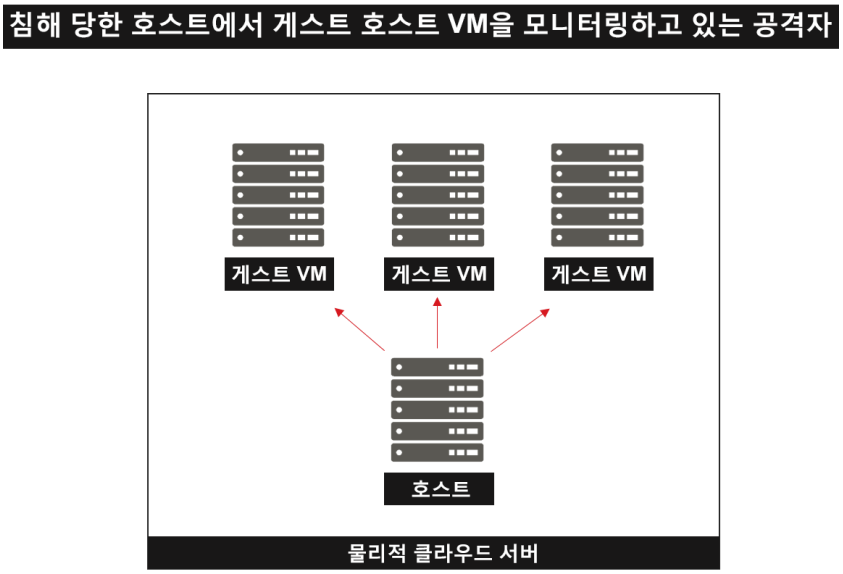

그림 7 - VM 공격 모델의 예

또 다른 클라우드 또는 컨테이너 인프라 공격은 공격자가 목표 VM 과 같은 물리적 서버에서 그 VM 을 지배할 수 있다고 가정합니다. 그러면 공격자는 몇 가지 방법을 동원해 물리적 서버의 다른 VM 들을 침범할지도 모릅니다. 가능성은 다음과 같습니다.

- VM 인프라의 취약점을 틈타 게스트에서 호스트 시스템으로 침입
- 측면 채널 공격을 통해 다른 게스트 VM 에서 보안 키를 유추
- 물리적 서버에서 과도한 리소스를 소비해 목표 VM 이 공격자가 장악하고 있는 물리적 서버로 옮겨가도록 유도

어떤 공격 모델이 사용되든 기업이 이 위험을 차단하기 위해 할 수 있는 것은 거의 없습니다. 대신, 클라우드 서비스 업체가 적절한 기능을 구현하여 공격자가 클라우드 또는 컨테이너 인프라를 와해시킬 여지를 없애야 합니다.

이 같은 위험을 낮추는 한 가지 방법은 각 컨테이너를 특정 사용자 한 명과 고유한 크립토그래픽(cryptographic) 아이덴티티 하나로 한정하는 것입니다. 이렇게 하면 리소스가 많이 소요되고 추가 비용이 발생할 수도 있지만 공격자가 VM 인프라를 남용해 한 번에 복수의 사용자 또는 복수의 서비스에 액세스하는 능력이 저하됩니다.

클라우드 또는 컨테이너 환경에서 특권이라는 위치는 게스트 VM 에서 실행되는 애플리케이션에게는 심각한 위협이지만 그 위치에 액세스하기까지는 고도의 기술과 많은 시간, 리소스가 필요합니다. 일단 액세스를 했더라도 공격자는 관심 있는 VM 이 어느 시스템에 들어 있는지 확인하기 위해 오랫동안 액세스 상태를 유지해야 합니다. 또한 클라우드 서비스 업체의 사고 서브시스템에게 발각되지 않고 해당 VM 을 모니터링하거나 변경할 수 있어야 합니다. 이것은 매우 어려워 침해의 가능성을 낮추기에 충분합니다.

그러나 이런 유형의 침해는 게스트 VM 또는 그 위에서 실행되고 있는 애플리케이션에게 상당 부분 탐지되지 않습니다. 따라서, 메트릭스를 모아 특정 클라우드 VM 또는 컨테이너의 거동에 이상한 점을 드러낼 수도 있지만 침해가 실제로 발생했는지를 확인하기는 매우 어렵습니다. VM 인프라의 호스트 레이어에서 충분한 특권을 지니고 있는 공격자라면 게스트를 조작해 조작을 탐지하기 어렵게 만들 수 있을 것이기 때문입니다.

게스트에서 게스트로 향하는 공격은 클라우드 서비스 업체라고 해도 탐지하기가 지극히 어렵습니다. 그러나 이런 공격은 대체로 이론에 불과합니다. 사이드 채널 공격은 가능하지만 실제로 가능한지를 두고는 의견이 분분합니다. 이런 공격에는 기초 실행 플랫폼에서 일정 수준의 일관성이 필요한데, 실제 환경에서는 보장이 안 되기 때문입니다. 더구나, VM, 컨테이너, 하이퍼바이저 환경에서 호스트를 향한 게스트의 에스컬레이션 공격은 찾아보기 어렵고 성공은 더더욱 어렵습니다. 이로 인해 취약성이 게스트 또는 특정 목표에 대한 대량 공격으로 이어질 가능성은 더욱 낮아집니다.

따라서, 이것이 공격자에게는 중요한 특권 위치이긴 하나, 난이도와 비용, 기회의 비현실성으로 인해 공격이 성공할 가능성은 낮습니다.

3.3 애플리케이션 서비스 공격

애플리케이션 실행 아키텍처에 대한 논의는 본 문서의 범위를 벗어나는 것이긴 하지만 이 레이어가 공격의 위험에 가장 크게 노출돼 있다는 사실은 기억해야 합니다. 서비스 생태계가 본 가이드에 명시된 바와 같이 올바르게 구성되었다면 공격자들은 네트워크 인프라에 대한 공격보다 애플리케이션 자체에 대한 공격으로 옮겨갈 것입니다.

애플리케이션은 어떤 제품이나 서비스에서도 가장 복잡한 레이어이고, 공격자가 여러 가지 기술을 통해 특권을 확대할 가능성이 상존하는 곳입니다. 그러므로, 본 문서의 목표는 네트워크 인프라를 공격 대상에서 배제하는 데 있지만, 그렇게 되면 성공 가능성이 훨씬 더 높은 *한 곳으로* 공격 대상이 이동하는 결과를 낳게 됩니다.

공격의 여지를 없애기 위해서는 애플리케이션 보안에 관한 양질의 문서(예컨대 OWASP Secure Application Design Project [5])를 다양하게 검토하여 애플리케이션 실행 아키텍처를 최대한 안전하게 구현해야 합니다.

3.4 프라이버시

파트너 시스템이 데이터/메트릭스나 사용자 지향 컴포넌트를 소비하여 전체 시스템에 부가가치를 제공하도록 설계돼 있더라도 그 파트너가 구현하는 보안의 수준에 대해서는 아무런 보장이 없습니다. 단순히 정보를 3 자에게 제공하기보다는 어떤 종류의 데이터를 넘겨줄지, 유형의 수익은 무엇이어야 하는지, 그 정보를 어떻게 보호할지 평가해야 합니다.

법적 책임은 계약과 보험으로 낮출 수 있지만 3 자의 잘못으로 고객이 떠날 수도 있습니다. 조직이라면 이와 같은 위험을 감수하기보다는 3 자 엔지니어링팀을 평가하여 인프라와 애플리케이션, API 의 보안 수준을 판단해야 합니다. 보안 수준이 충분하지 않다면 다른 파트너를 찾아보기를 권장합니다.

3.5 악성 객체

3 자 시스템은 정보나 멀티미디어를 소비자에게 제공하도록 설계돼 있습니다. 그리고 이것을 달성하는 한 가지 방법으로 광고가 있습니다. 그 안에는 여러 가지 파일이 복잡하게 얽혀 있는데 소프트웨어가 이것을 바르게 분석하기는 어렵습니다. 광고 네트워크는 맬웨어를 분산하기에 좋은 채널입니다. CDN(Content Distribution Network)도 맬웨어 분산에 가능성이

있는 채널입니다. 복잡한 멀티미디어 타입이나 코드 번들로 역동적인 정보를 렌더링하는 시스템은 어느 것이든 맬웨어가 몰릴 수 있습니다.

따라서, 기업에서는 특정 채널을 통과하는 기술의 유형을 평가해야 합니다. 그리고 어느 것을 허용할지, 고객에게 전달하기에 과한 것은 어느 것인지 결정해야 합니다. 예컨대, 광고 회사가 IoT 기업이 파트너에게 제공하는 프록시 서비스 애플리케이션을 통해 클라이언트 시스템으로 Java 코드를 보내고자 한다면, 일단 특정 환경에서 작동하는 클라이언트 시스템이 Java 기술을 통한 공격에 더 취약한지 아닌지 판단해야 합니다. 더 취약하다고 판명되면 Java 는 불허하고 HTML 같은 기술은 통과시키는 것입니다.

맬웨어는 다형 파일 타입부터 Adobe Flash, Java, 멀티미디어 익스플로잇(exploit)까지 형태가 다양하기 때문에 한 가지 방법으로 최종 사용자의 안전을 보장하지는 못합니다. 간단한 대책으로는 엔지니어링 팀이 채널에 사용*될* 기술과 사용자가 받는 영향을 규정한 정책을 집행하는 것입니다. 서브시스템 모니터링과 샌드박스를 도입해 클라이언트 시스템에 렌더링되는 객체가 덜 남용되도록 하는 방법도 있습니다.

3.6 인증과 허가

파트너는 특정 사용자 집단만을 대상으로 하는 서비스를 제공하기도 합니다. 사용자가 선택에 따라 돈을 내고 이용하는 서비스가 그 예가 될 수 있습니다. 사용자가 네트워크 서비스 업체, 소셜 네트워크 인프라, 기존 M2M, IoT 관리 엔터티에서 제공하는 기존 인증 API 와 같이 잘 알려진 별도의 기술과 공유하는 자격증명을 이용해 시스템에 인증하는 방법도 있습니다.

이렇게 하면 플랫폼 간에 기술을 공유하기는 좋지만, 엔지니어링 팀에서는 기술이 우발적으로 자격증명을 소비해 3 자 서비스에 공여되지 않은 허가를 남용하지 않도록 해야 합니다. 예컨대, 어떤 플랫폼 API 는 사용자가 수락하거나 거부하는 클래스로 허가를 한정하게 할 수 있습니다. 그러면 사용자는 자신의 프라이버시 니즈에 적합한 것으로 환경을 조정할 수 있습니다. 플랫폼이 세분해서 보안 허가를 제공할 수 없다면 액세스를 *아주* 원하는 기술을 명시해야 합니다.

엔지니어링 팀은 파트너들에게 허가를 세분화해 한 서비스가 폐지되더라도 존속하는 데이터가 우발적으로 노출될 여지를 차단하라고 요청할 필요가 있습니다.

3.7 오탐과 미탐

서비스의 모니터링과 로깅은 기존 보안 인프라를 강화하기에 아주 좋은 방법이지만 오탐과 미탐의 여부를 면밀하게 평가해야 합니다. 이들 시스템은 IoT 제품 또는 서비스 내 여러 생태계에서 나온 데이터를 해석할 뿐, 내부 엔지니어링 팀에서 개발한 것이 아니고 한 이벤트에 대해 인위적인 정보만 제공하기 때문입니다. 그러면서 공격 이벤트가 실제로 일어나고 있는지 정확히 분간하지 못할 수도 있습니다.

그러므로, IT 및 엔지니어링 팀을 통해 의심스런 이벤트가 실제로 악성 거동 때문인지 판단하는 것이 중요합니다. 이렇게 하면 모니터링 팀이 합법적 사용자의 시스템 액세스를 불허할 가능성을 불식할 수 있습니다. 이 프로세스가 자동화되었는데 부정확하다면 클라이언트 애플리케이션이나 인프라의 변칙으로 인한 오탐 탓에 많은 사용자가 정당한 서비스를 받지 못할 수도 있습니다. 의심스러운 이벤트가 일어나고 있다면 IT 와 엔지니어링 팀에서는 데이터를 보고 그것이 실제로 공격인지 평가해야 합니다.

엔지니어링 팀에서는 또 아날로그 채널을 통해 획득한 정보를 모델링할 때에도 주의해야 합니다. 그렇게 획득한 데이터를 완전히 신뢰할 수 없을 때 애플리케이션이 안전한 처리 방법을 제대로 평가하지 못한다면, 특히 데이터를 매우 높은 속도로 처리해야 하는 생태계에서 오탐과 미탐은 심각한 결과를 초래할 수도 있습니다. 시간과 기술, 전문성만 충분하다면 어떤 아날로그 데이터든 디지털 시스템으로 위장할 수 있습니다.

4 자주 하는 보안 질문

본 문서에서 서비스 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 섹션에서는 엔드포인트의 관점에서 공통의 목표를 기술하고 그 목표를 달성하기 위한 권고사항을 제시합니다.

4.1 클로닝에는 어떻게 대처해야 합니까?

IoT 서비스 업체에서 만든 유효한 장치와 복제품이거나 "립 오프"(클론)인 장치를 구별하기는 쉽지 않습니다. 허가 받지 않은 엔드포인트에 서비스를 제공하고 싶은 IoT 서비스 업체는 없습니다. 서비스 업체도 CPU 시간과 대역폭, 디스크 스토리지, 기타 리소스에 대해 대가를 지불해야 하기 때문입니다. 서비스 업체는 장치가 IoT 서비스 업체에서 만든 것이든 그렇지 않든 값을 치러야 합니다.

서비스 업체는 또 엔드포인트 아키텍처가 와해되고 있는지 판단하는 능력도 갖춰야 합니다. 그래야 여러 인스턴스로 *클로닝된(cloned)* 장치에 대응 조치를 할 수 있습니다. 이 같은 클로닝은 부도덕한 제조업체나 특정 사용자를 사칭하려고 하는 공격자가 즐겨 사용합니다.

다음 권고사항을 서비스를 이용한 클로닝 근절에 참고하기 바랍니다.

- 조직의 신뢰 기반(RoT) 정의
- 네트워크 인증 서비스 이용
- 서비스 생태계를 통해 인증 의무화
- 애플리케이션 레이어 인증 및 허가 정의

4.2 사용자를 엔드포인트를 통해 인증하는 방법은?

IoT 에서 중요한 개념 가운데 하나는 엔드포인트 인증을 사용자 인증과 분리하는 것입니다. 엔드포인트는 신뢰 컴퓨팅 기반으로 인증할 수 있지만 *사용자*를 인증하는 방법은 엔드포인트의 통신 보안 TCB 를 기반으로 하는 별도의 과정입니다. 이 같은 추상화에서 가장 중요한 것은 통신 채널이 사용자 인증에 얼마나 믿을 만한지 평가하는 것입니다.

예컨대, 엔드포인트 TCB 가 없거나 엔드포인트 TCB 구현이 약해 엔드포인트의 신뢰도가 낮다면, 실행할 엔드포인트 소프트웨어/펌웨어에 의존하는 사용자 인증 메커니즘은 신뢰할 수 없습니다. 이것은 엔트포인트를 통한 사용자 인증을 인증으로 간주할 수 없다는 의미입니다.

다른 관점에서 아키텍처가 좋은 엔드포인트 TCB 도 인증 체계를 쉽게 우회할 수 있다면 최종 사용자 인증이 허술해질 수 있습니다. 따라서, 서비스 생태계는 엔드포인트의 신뢰도뿐만 아니라 인증 메커니즘의 구현도 중시하여 서비스 생태계가 올바른 사용자의 시스템 로그인을 보장할 수 있게 해야 합니다.

이 복잡한 문제에 관해서는 다음 권고사항을 참고하기 바랍니다.

- 서비스 신뢰 컴퓨팅 기반 구현
- 조직의 신뢰 기반(RoT) 정의
- 명확한 인증 모델 정의
- 네트워크 인증 서비스 이용
- 서비스 생태계를 통해 인증 의무화
- 강력한 비밀번호 정책 적용

- 애플리케이션 레이어 인증 및 허가 정의

4.3 어떻게 하면 서비스가 엔드포인트의 변칙 거동을 찾아낼 수 있습니까?

분산 IoT 네트워크에서 엔드포인트를 관리할 때 어려운 점 가운데 하나는 엔드포인트가 변칙적으로 거동하는지 아닌지 판단하는 것입니다. 이것은 보안 측면뿐만 아니라 안정성 측면에서도 중요합니다. 변칙 거동은 펌웨어나 하드웨어에 문제가 있다는 신호일 수도 있고 조직이 예상치 못한 문제를 해결할 준비를 해야 한다는 경고일 수도 있습니다. 그러나 거동이 네트워크의 일부분에 국한돼 있어 IoT 서비스 업체가 분석하지 못할 경우 이들 메트릭스는 사라지고 조직은 불리한 상황에 놓이게 됩니다.

이 문제를 해결하려면 엔드포인트와 네트워크 레이어, 서비스 생태계에서 거동을 검사하는 능력이 필요합니다. 그러나 이들 데이터 포인트를 한 데 모을 수 있는 인프라와 서비스, 파트너십이 제대로 갖춰져 있지 않다면 문제가 있는지 없는지, 문제가 보안과 관련 있는지 안정성과 관련 있는지 판단하는 데 필요한 정보를 얻지 못합니다.

서비스 생태계의 관점에서 다음 권고사항을 검토해 보기 바랍니다.

- 공용 인터넷에 노출된 시스템의 보안 인프라 정의
- 시스템 로깅 및 모니터링 방식 정의
- 통신 모델 정의
- 네트워크 인증 서비스 이용
- 입력 검증 실시
- 출력 필터링 실시
- 파트너 강화 모니터링 서비스 이용
- 무선 연결에 사설 APN 이용
- 오탐/미탐 평가 모델 정의

4.4 서비스가 엔드포인트의 변칙 거동을 어떻게 제한합니까?

어떤 엔드포인트가 변칙 거동을 한다고 확인되면 서비스는 어떤 리소스를 한정하거나 제한할지 결정을 해야 합니다. 이 문제는 서비스 인프라의 모든 레이어와 관련이 있습니다.

예컨대, 번잡한 루프 안에 위치한 모바일 네트워크와 끊임 없이 연결되고 분리되는 셀룰러 기반 엔드포인트라면 불규칙한 거동이 해결될 때까지 강제로 꺼둬야 합니다. 공격자가 백엔드

서비스 공격에 이용 중인 침해된 엔드포인트도 또 다른 예가 될 수 있습니다. 이 시나리오에서 백엔드 서비스는 남용되고 있는 엔드포인트가 서비스에 절대로 닿지 않게 해야 합니다.

이 시나리오를 처리하는 방법은 IoT 서비스 업체마다 다르며 사업 목표, 사고 처리 방식에도 영향을 받습니다. 위와 같은 가이드라인의 개발과 관련해서는 다음 권고사항을 참고하기 바랍니다.

- 조직의 신뢰 기반(RoT) 정의
- 공용 인터넷에 노출된 시스템의 보안 인프라 정의
- 사고 대응 모델 정의
- 복구 모델 정의
- 퇴역 모델 정의
- 통신 모델 정의
- 노출된 데이터를 대상으로 위반 정책 정의
- 서비스 생태계를 통해 인증 의무화
- 무선 연결에 사설 APN 이용
- 오탐/미탐 평가 모델 정의

4.5 어떤 서버 또는 서비스가 해킹당했는지 어떻게 압니까?

엔드포인트 거동은 이해하기가 어렵고 공격 대부분을 잡아내려면 방대한 거동 분석이 필요하지만 서비스 생태계는 조금 더 단순명료합니다. 서비스와 서버는 IoT 서비스 업체 또는 클라우드나 서버 인프라를 관리하는 파트너에게 철저히 제어되는 환경 안에서 배포됩니다. 그러므로 조직과 파트너는 쉽게 이용할 수 있는 모니터링/진단 시스템을 통해 문제의 불씨를 찾아내 봉쇄합니다.

다음 권고사항을 참고하기 바랍니다.

- 관리 모델 정의
- 시스템 로깅 및 모니터링 방식 정의
- 사고 대응 모델 정의
- 입력 검증 실시
- 출력 필터링 실시

4.6 서버가 해킹당하면 어떻게 해야 합니까?

서버가 침해를 당했다고 확인되면, 관리팀에서는 문제를 최대한 신속하고 효율적으로 해결해야 합니다. 이때 어떤 리소스와 정보, 계정이 위험에 처했는지 특정하는 과정에서 문제가 복잡해지기도 합니다. 아키텍처가 허술한 환경에서는 침해의 파장을 계량화하지 못할 수도 있습니다. 그러므로 조직에서는 보안 취약성을 해결하고 *동시에* 현장에서 위험에 처한 자산을 보호하는 계획을 *반드시* 실행에 옮겨야 합니다. 생태계와 취약성에 대해 안전을 확보했다면 피해를 입은 기술을 재건하는 계획을 수립해도 됩니다.

자세한 사항은 다음 권고사항을 참고하기 바랍니다.

- 사고 대응 모델 정의
- 복구 모델 정의
- 퇴역 모델 정의
- 보안 등급 정의
- 각 데이터 형식에 등급 정의

4.7 관리자는 서버와 서비스를 어떻게 상대해야 합니까?

서비스 생태계를 위험에 빠뜨리지 않는 관리 모델을 개발하는 일은 IoT 서비스의 아키텍처에서 중요한 부분입니다. 관리에는 여러 레이어가 있으며 엔지니어링 및 보안 팀에서 각 레이어를 검토해야 합니다. 예컨대, 서버를 관장하는 관리자는 (가상, 마이크로 서비스, 유니 커넬 아키텍처의 사용 여부를 불문하고) 안정적이고 안전한 통신 채널을 통해 라이브 서버를 상대할 수 있어야 합니다. 웹 애플리케이션을 관장하는 관리자는 같은 웹 통신 레이어를 통해 애플리케이션을 상대하기도 하지만 코드에 임베드된 전문 애플리케이션을 통하기도 합니다.

관리 수요가 무엇이든 인터페이스는 액세스를 제한하여 공격자가 기술을 상대하거나 남용할 여지를 주지 말아야 합니다. 다음 리소스를 참고하기 바랍니다.

- 공용 인터넷에 노출된 시스템의 보안 인프라 정의
- 관리 모델 정의
- 명확한 인증 모델 정의
- 통신 모델 정의
- 무선 연결에 사설 APN 이용

4.8 어떻게 해야 서버 아키텍처가 침해의 영향을 제한할 수 있습니까?

IoT 네트워크의 장점 가운데 하나는 서비스를 특정 소비자와 연동할 수 있다는 것입니다. 웹 서비스에서는 사용자 각자가 어떤 유형의 장치에서든 또한 잠재적으로는 세계 어디서든 서비스를 이용할 수 있어야 합니다. IoT 기술에서는 그렇지 않습니다. IoT 기술에서는 일반적으로 특정 엔드포인드 장치가 있어야 IoT 서비스를 이용할 수 있습니다. 이 같은 차이 때문에 서버 에코시스템 아키텍트들은 엔드포인트와 소비자의 1:1 관계를 이용해 엔드포인트의 백엔드 액세스를 제한합니다.

엔드포인트가 센서 메트릭스를 백엔드 서비스로 푸시하는 상황을 가정해 보겠습니다. 마이크로 서비스 아키텍처라면 서비스 생태계가 특정 마이크로 서비스나 유니 커넬을 투입해 특정 소비자를 상대할 수도 있습니다. 엔지니어는 이 아키텍처를 이용해 *개별 소비자에게 특정한* 데이터와 서비스를 제공하는 데 필요한 리소스와 액세스 능력에 *한정하여* 마이크로 서비스에 공급되게 할 수 있습니다.

이는 어떤 서비스가 침해를 받았는데 엔드포인트만 그 서비스와 통신할 수 있다면 그 서비스를 침해하는 데서 오는 이익이 없다는 뜻입니다. 침해로 얻은 액세스가 엔드포인트에 *이미* 제공되고 있는 리소스에 한정되기 때문입니다. 요컨대, 공격에서 얻는 실익이 없습니다.

다음 권고사항을 참고하기 바랍니다.

- 서비스 신뢰 컴퓨팅 기반 구현
- 부츠트랩 방법 정의
- 공용 인터넷에 노출된 시스템의 보안 인프라 정의
- 영구저장 모델 정의
- 관리 모델 정의
- 퇴역 모델 정의
- 명확한 인증 모델 정의
- 가능할 경우 서버 프로비저닝
- 애플리케이션 실행 환경 정의
- 가상 머신 침해

4.9 어떻게 해야 서버 아키텍처가 침해 시 데이터 손실을 줄일 수 있습니까?

IoT 아키텍처의 또 하나 좋은 점은 데이터 손실의 저감입니다. 이것은 서비스를 특정 사용자에게 한정하는 방식과 유사합니다. 데이터도 사용자가 인증을 마치면 그 사용자에게 한정할 수 있습니다. 그러나 데이터 저장은 사용자별로 구현하기가 쉽지 않습니다. 데이터베이스와 저장 인프라의 비용 때문입니다.

대신, 저장 인프라 안에서 특정 사용자 대신 작동하는 서비스에 고유한 토큰을 제공해야 합니다. 이렇게 하면, 공격자가 데이터 저장 환경에 액세스해 서비스에 연결할 수 있을지는 몰라도 침해 당한 사용자 외에는 사용자 데이터를 상대하거나 불러오거나 변경할 수 없습니다.

네트워크 레이어의 관점에서는 트래픽의 흐름을 서버 생태계에서 인터넷으로 줄이는 것도 필요합니다. 진출 제어를 하면 공격자는 지적재산이나 고객 데이터를 특정 채널을 통해 보내야 합니다. 이렇게 되면 다량의 데이터를 옮기기 어렵게 되거나 사고 중 통신을 탐지해 차단할 수 있는 통신 레이어를 통과해야 합니다.

자세한 사항은 다음 권고사항을 참고하기 바랍니다.

- 부츠트랩 방법 정의
- 공용 인터넷에 노출된 시스템의 보안 인프라 정의
- 영구저장 모델 정의
- 보안 등급 정의
- 각 데이터 형식에 등급 정의
- 가능할 경우 서버 프로비저닝
- 애플리케이션 실행 환경 정의
- 기본 오픈 또는 페일 오픈 방화벽 규칙

4.10 서비스 아키텍처는 어떻게 무단 사용자의 연결을 제한합니까?

일반적인 IoT 아키텍처의 장점 가운데 하나는 무단 인터넷 사용자가 백엔드 서비스에 직접 연결하기가 어렵다는 것입니다. 웹 애플리케이션 대부분은 이런 고급 기능이 없으며 공용으로 제공해야 합니다. 그러나 IoT 에서는 엔드포인트가 특정 서비스에 연결되어야 하는 엔터티이므로 가상 사설 네트워크(VPN)을 이용해 백엔드 서비스에 대한 접속을 제한할 수 있습니다. 이것은 표준 인터넷 프로토콜을 통해 구현하거나 사설 APN 같은 모바일 서비스를 이용해 구현할 수 있습니다. 자세한 사항은 다음 권고사항을 참고하기 바랍니다.

- 공용 인터넷에 노출된 시스템의 보안 인프라 정의
- 무선 연결에 사설 APN 이용

4.11 어떻게 하면 원격 익스플로잇의 가능성을 낮출 수 있습니까?

웹 애플리케이션과 서비스의 원격 익스플로잇은 인프라 관리자들이 항상 우려하는 것입니다. 매일 같이 공격자가 내부 네트워크나 중요한 리소스에 접근하지 못하게 하는 싸움이 벌어지고 있습니다. 공격자가 서비스 생태계를 침해할 여지를 줄이는 방법은 쉽고 빠르게 유지보수가 가능한 적정 수의 서비스로 잠재적 목표물의 수를 줄이는 것뿐입니다. 아키텍처에게 두 번째로 중요한 보강 요소는 기초 아키텍처의 설계입니다. 실행 아키텍처, 운영체제 구성, 배포 툴체인, 프로그래밍 언어 보안, 그 외 애플리케이션의 실행 보안을 규정하는 옵션이 그것입니다. 이들 옵션이 애플리케이션의 충돌과 인프라의 침해를 가르는 요소가 될 수도 있습니다.

원격 익스플로잇의 여지를 줄이는 방법은 다음을 참고하기 바랍니다.

- 공용 인터넷에 노출된 시스템의 보안 인프라 정의
- 업데이트 모델 정의
- 입력 검증 실시
- 출력 필터링 실시
- 기본 오픈 또는 페일 오픈 방화벽 규칙
- 애플리케이션 실행 환경 정의
- 로우해머 및 유사 공격
- 가상 머신 침해

4.12 서비스는 사용자 프라이버시를 어떻게 관리할 수 있습니까?

IoT 서비스 업체가 성장하면 소비자 데이터를 기발하게 이용하는 여러 조직과 다방면으로 제휴 관계를 맺게 됩니다. 그러나 거기에는 소비자의 프라이버시라는 대가가 따릅니다. 소비자는 어떤 데이터를 제휴업체와 공유할지 또 어떻게 사용될지 결정할 권리를 지녀야 합니다. 또한 제휴업체는 데이터를 특정한 방식으로 이용해야 합니다. 승인 모델이 여기에 도움이 되겠지만, 프라이버시와 법적 영향, 기업 보험 등 논의의 범위는 훨씬 더 방대합니다.

조직 내에서 논의를 시작하고자 한다면 다음 권고사항을 참고하기 바랍니다.

- 보안 등급 정의

- 각 데이터 형식에 등급 정의
- 명확한 인증 모델 정의
- 노출된 데이터를 대상으로 위반 정책 정의
- 통신 프라이버시 모델을 평가한다
- 3 자 데이터 배포 정책을 수립한다
- 3 자 데이터 필터를 구축한다
- 사용자를 위한 API 를 구축해 프라이버시 속성을 관리한다

4.13 서비스가 어떻게 가용성을 높일 수 있습니까?

Dos 또는 DDos 공격은 요즘 인터넷에서 워낙 흔하므로 기업마다 이 같은 부류의 대규모 공격에는 대비를 해둬야 합니다. 또한 공격이 길어지는 상황에서도 온라인 상태를 유지할 수 있어야 합니다. 이 같은 공격이 흔한 이유는 실행하는 데 별다른 기술이 필요하지 않고 공격 실행에 필요한 도구를 온라인에서 쉽게 구할 수 있기 때문입니다. 실제로, 돈만 지불하면 공격자가 특정 목표를 대상으로 DDos 공격을 실행해주는 온라인 서비스도 있습니다.

이에 따라, 이 같은 위협에 맞서 서비스의 가용성을 확보해주는 완전히 새로운 모델들이 개발되었습니다. 그런 서비스 생태계를 구축할 때에는 다음 권고사항을 참고하기 바랍니다.

- 공용 인터넷에 노출된 시스템의 보안 인프라 정의
- 시스템 로깅 및 모니터링 방식 정의
- 사고 대응 모델 정의
- 복구 모델 정의
- 통신 모델 정의
- 기본 오픈 또는 페일 오픈 방화벽 규칙

5 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.

5.1 서비스 신뢰 컴퓨팅 기반 구현

신뢰 컴퓨팅 기반(TCB)은 하드웨어와 소프트웨어, 프로토콜, 그리고 정책의 집합체입니다. 어떤 컴퓨팅 플랫폼이든 TCB 가 근간이 되어야 하며 TCB 를 통해 애플리케이션이 안정적으로 안전하게, 높은 품질로 실행될 수 있는 환경을 정의해야 합니다.

TCB 는 어느 급의 시스템에도 구축하고 배포할 수 있습니다. 모바일 장비(스마트폰), IoT 엔드포인트는 물론 서비스 생태계에 존재하는 서버에도 가능합니다. TCB 를 이루는 기술은 모두 비슷합니다. 하지만 시스템의 급에 따라 기술의 특징이 크게 달라지기도 합니다. 예컨대, 클라우드 서버에서 TCB 의 부트스트래핑은 엔드포인트의 부트스트래핑과는 모습이 크게 다릅니다.

서비스 엔드포인트에서 TCB 를 구축한다는 것은 애플리케이션 이미지의 확산 방식을 정의하는 것과 같습니다. 이 맥락에서 이미지는 실행 가능한 애플리케이션과 구성 파일, 메타데이터로 구성된 원시 바이너리 데이터를 나타냅니다. 이런 것들을 통칭하여 애플리케이션 이미지, 줄여서 이미지라고 합니다. 최신 서비스 생태계에서는 시스템이 수요에 따라 복제되거나, 가동하거나 스핀다운 돼 컴퓨팅 환경의 변화에 맞춰 확대/축소를 거듭합니다. 이는 시스템이 보안 모델을 꾸준히 유지하면서도 효과적으로 확대/축소할 수 있도록 TCB 가 그 방식을 정의해야 한다는 의미이기도 합니다.

이를 위한 팀의 역할은 다음과 같습니다.

- 컴퓨팅 플랫폼 표준화:
  - 일단의 물리적 서버 모델 선택
  - 일단의 클라우드 플랫폼 또는 가상 머신(VM) 이미지 선택
- 컴퓨팅 플랫폼에서 실행할 애플리케이션과 라이브러리, 구성 파일 정의:
  - 컨테이너 환경 정의 (해당할 경우)
- 위에서 정의한 요소들로 구성된 애플리케이션 이미지 생성

- 계층 TCB 서명 키를 이용해 이미지 아카이브에 크립토그래픽 서명
- 아카이브와 서명 안전하게 저장

위와 같은 역할을 완수한다면 특정 계층에서 배포 가능한, 승인 받은 애플리케이션 이미지가 탄생하게 됩니다. 계층마다 가장 적합한 하드웨어와 애플리케이션 모델이 달라집니다. 예컨대, 데이터베이스 하드웨어는 성능 및 저장 요건이 애플리케이션 계층과는 크게 다릅니다. 저장 계층은 데이터베이스 계층과 하드웨어 저장 요건은 유사하지만 성능 요건을 다릅니다. 각 계층의 정의를 표준화하고 나면 각 하드웨어 플랫폼에서 배포와 검증이 가능한 이미지가 탄생합니다.

TCB 배포 시 난제는 다음과 같습니다.

- 이미지의 크립토그래픽 서명을 관리하는 조직적 신뢰기반의 구축
- 각 이미지에 서명하는 절차의 수립
- 각 이미지를 검증하는 절차의 수립
- 이미지를 자동으로, 이미지 검증과 함께 확산하는 절차의 수립

본 권장사항과 더불어 다음 기관에서 발간한 자료를 참고하기 바랍니다.

- GlobalPlatform Card Specification [11]
- Trusted Computing Group's TPM Specification [6]
- GlobalPlatform TEE Internal Core API Specification [12]

5.1.1 위험

잘 정의된 신뢰 컴퓨팅 기반이 없다면 컴퓨팅 플랫폼은 자기가 엔지니어링 팀에서 승인한 구성에서 실행되고 있는지 확인할 길이 없습니다. 이것이 중요한 이유는 애플리케이션 서브시스템이 공격자에게 침해를 당했는지 판단할 수 있어야 하기 때문입니다. TCB 를 이용하면 이 같은 위험을 해소할 수 있을 뿐만 아니라 모든 네트워크 통신이 보안 레이어를 갖출 수도 있습니다.

5.2 조직의 신뢰 기반(RoT) 정의

조직의 신뢰 기반이란 조직 내에서 컴퓨팅 플랫폼 엔터티를 인증하기 위한 인증서 또는 공용키 기반의 시스템을 말합니다. 서비스 생태계의 각 컴퓨팅 플랫폼은 네트워크 통신 중에 암호화된 방식(cryptographically) 인증을 받아야 합니다. 이렇게 되면 내부자 또는 네트워크에서 특권 위치에 있는 누군가가 권한을 사칭하거나 시스템의 신뢰를 남용할 가능성이 줄어듭니다.

조직의 신뢰 기반을 구축하기 위해서는 다음과 같은 조치가 필요합니다.

- 예를 들면 하드웨어 보안 모듈(HSM)을 구축하거나 도입해 조직의 기반이 되는 비밀을 저정한다
- 기반이 되는 비밀 및/또는 인증서를 만든다
- 비밀의 사적 측면을 안전하게 보관한다
- 서명 키를 하나 이상 만들어 계층 TCB 서명 키에 사용한다
- 조직의 기반으로 서명 키의 공적 측면에 서명한다
- 서명 키가 사업 및 엔지니어링 책임자의 인증과 허가 없이 사용되지 않게 한다

새 계층 시스템이 정의되면 그 때마다 서명 키로 시스템의 고유 크립토그래픽 키 또는 인증서에 서명할 수 있습니다. 이 새 시스템에 다른 시스템이 연결될 경우, 시스템은 조직의 기반에서 정의한 신뢰의 체인을 검증해 시스템의 정체를 확인할 수 있습니다.

메시지가 시스템을 나타내는 공용 키의 서명을 받았는지 크립토그래픽 방식으로 검증하는 것입니다. 그 후, 시스템은 서명 키가 생성한, 그 시스템의 고유한 공용 키의 서명을 검증합니다. 이어, 클라이언트는 서명 키가 정말로 조직의 기반에서 인증한 서명 키가 맞는지 검증해야 합니다.

조직에서 각각의 인증서 또는 비밀에 접근할 수 있는 사람은 줄어들고 수립된 정책과 절차는 그런 비밀을 이용할 수 있는 사람을 제한해야 하므로, 각 신뢰 수준은 클라이언트가 기반 체인(root chain)을 따라 내려갈수록 높아져야 합니다.

서비스 생태계 안에서 허가 받은 피어에게 인증 기능을 제공하는 서비스가 필요합니다. 예컨대, 인증서 또는 비밀을 이용한 인증은 그 자체가 보안을 보장하지는 못합니다. 인증서가 취소되었는지, 현재 유효한지 검증하는 서비스가 있어야 합니다. 또한 기초 인프라의 요건에 따라 단기간에 서버 또는 서비스의 신원을 인증하는 서비스도 필요할 수 있습니다.

신뢰 기반을 정의할 때에는 다음 사항을 고려해야 합니다.

- 각 비밀은 남용되지 않게 보호해야 한다
- 각 비밀의 내부 이용은 검증 가능하도록 추적하고 모니터링 해야 한다
- 비밀 이용 승인을 받은 개인은 비밀에 액세스할 때 다단계 인증을 이용해야 한다
- 일관성 있고 안전한 이용을 담보하는 정책과 절차를 수립하기는 쉽지 않다
- 인증서를 퇴역시키거나 취소하는 절차를 만들기도 쉽지 않다

- 어떤 키가 남용되었는지 확인하기도 쉽지 않다
- 올바른 크립토그래픽 알고리즘을 찾는 일은 직관과는 거리가 멀다

신뢰 기반의 개념에 관해서는 다음의 정보 출처를 참고하기 바랍니다.

- 신뢰 컴퓨팅 그룹(Trusted Computing Group)
  - TPM 규격(TPM Specification) [6]
  - TCG IoT 보안 지침(TCG Guidance for Securing IoT) [7]
  - ISO 11889
- PKI 규격
  - RFC 2510
  - RFC 3647

5.2.1 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.

5.3 부츠트랩 방법 정의

애플리케이션이 바르게 작동하려면 안정적이고 안전한 고품질의 플랫폼에서 일관성 있게 로드되고 실행되어야 합니다. TCB 는 이 플랫폼의 구성 방식을 정의하지만, 부트스트랩 모델은 그 위에서 애플리케이션의 실행 방식을 정의합니다.

부트스트랩 모델을 효과적으로 정의하기 위해서는 다음 사항을 고려해야 합니다.

- 애플리케이션이 크립토그래픽 방식으로 자기를 피어에게 나타낼 수 있도록 API 를 정의한다
  - 신뢰 받는 업계 리더가 정의한 기존 API 의 활용을 고려한다
- 애플리케이션이 엔드포인트와 서비스 피어, 파트너를 인증하는 방식을 정의한다
- 애플리케이션의 구성이 어떤 모습이어야 하는지 정의한다
- 각 애플리케이션, 특히 서로 다른 계층에서 실행 중인 애플리케이션들이 고유한 정체성(identity)을 갖도록 한다

애플리케이션이 자기를 피어에게 크립토그래픽 방식으로 나타내는 것이 직관적으로 보일 수도 있고 그렇게 하기 위해 API 가 필요 없을 수도 있지만, 제작 중인 과정은 그렇게

직관적이지만은 않습니다. 부트스트랩 모델에서는 크립토그래픽 ID 가 애플리케이션 *에* 프로비저닝되는 *방식을* 고려해야 하기 때문입니다. 애플리케이션은 ID 를 어떻게 획득하는가? ID 는 안전하게 확보되는가? 비밀이 업데이트되거나 변경될 때 ID 가 비밀을 취소하는 과정은 어떻게 되는가?

런타임 시 애플리케이션은 효과적으로 실행할 수 있는 자원이 필요합니다. 애플리케이션은 이 과정에 연루된 외부 서비스, 엔드포인트, 파트너와 통신을 하고 상호 인증을 할 수 있어야 합니다.

때로는 애플리케이션의 구성이 생산 중 자기의 보안 수준을 결정하기도 합니다. 구성을 적용한 후 읽기 전용으로 애플리케이션에게 제공해야 합니다. 애플리케이션 또는 그 애플리케이션 인프라를 남용하는 누군가가 애플리케이션의 구성을 변경하는 일이 있어서는 안 됩니다.

조직의 신뢰 기반을 이용해 전 생태계에 배포된 계층 별로 신뢰 모델을 정의해야 한다면 애플리케이션마다 고유한 크립토그래픽 ID 를 확보할 수 있게 됩니다. 이렇게 되면 예컨대 피어가 데이터베이스 서비스와 애플리케이션 서비스를 구별할 수 있습니다.

5.3.1 위험

잘 정의된 부트스트랩 모델이 없다면 시스템이 운영에 필요한 각 레이어를 검증할 방법이 없게 됩니다. 요컨대, 전체 기술의 각 측면에 신뢰의 레이어가 사라집니다. 이 같은 신뢰 레이어의 부재는 복잡성을 유발하고 이는 다시 공격자가 파고 들만한 허점으로 귀결될 수 있습니다.

5.4 공용 인터넷에 노출된 시스템의 보안 인프라 정의

누구나 액세스할 수 있는 서비스의 경우, 서비스의 가용성과 기밀성, 무결성 유지를 위해 다음과 같은 몇 가지 보안 및 신뢰 기술이 필요합니다.

- DDoS 에 강한 인프라
- 로드 밸런싱 인프라
- 리던던시 시스템
- 웹 애플리케이션 방화벽 (옵션)
- 전통식 방화벽

이와 같은 기술을 애플리케이션의 계층 앞에 배치해 공격자가 조작하지 못하게 해야 합니다. 통신 보안 모델은 익명의 3 자가 시스템에 접근할 가능성을 일소하거나 낮추는 반면, 위 기술들은 공격자가 시스템을 무용지물로 만드는 능력을 약화시킵니다.

서비스가 구현하는 *모든* 프로토콜에는 프론트-엔드 보안을 적용해야 합니다. 예를 들어, 서비스가 IPv4 와 IPv6 를 통해 제공된다면, 두 프로토콜을 통해 동일한 보안 제약요소를 서비스에 적용해야 합니다. 어떤 서비스가 TCP 뿐만 아니라 SCTP(Stream Control Transmission Protocol)를 통해서도 액세스할 수 있다면 두 프로토콜 모두에 대해 보안 제약요소를 적용해야 합니다. IoT 제품이나 서비스에 연동된 공용 서비스를 제공하지 않는 포트는 액세스를 허용하면 안 됩니다.

가능하면 진입*과* 진출 필터링을 모두 관리해야 합니다. 진입 필터링으로 공격이 어느 정도 차단되기는 하지만 공용 서비스에 대한 공격으로도 시스템 생태계는 침해를 당할 수 있습니다. 이 시점에서 진출 필터링이 중요합니다. 생태계 안에서 좌우로 움직이는 공격자가 침해된 컴포넌트를 이용하지 못하게 하기 위함입니다. 또한, 진출 필터링은 공격자가 중요한 데이터를 생태계에서 본인이 제어하는 서버로 갖고 나오는 과정을 복잡하게 만드는 역할도 합니다. 관리자가 공격자를 찾아내 격리하도록 시간을 벌어주는 것입니다.

몇몇 회사에서 기존 기술과 결합이 가능한 간단한 API 모델을 통해 이 서비스를 제공하고 있습니다. 덕분에 힘 안 들이고 기술을 이용할 수 있습니다. 별다른 엔지니어링 차원의 수고 없이 가입하고 서비스 제공업체의 시스템 안에서 애플리케이션을 구성하기만 하면 됩니다. 서비스 업체와 상의하여 업체의 보안 기술을 회사에 구현하는 최선의 방법을 찾아보기 바랍니다.

본 권장사항과 더불어 다음 기관에서 발간한 자료를 참고하기 바랍니다.

- 아마존의 DDoS 회복력 강화 모범 사례:
  - https://d0.awsstatic.com/whitepapers/DDoS_White_Paper_June2015.pdf
- 아버 네트웍스 DDoS 대응 모범 사례:
  - https://www.arbornetworks.com/images/documents/Arbor%20Insights/AI_DDoSMitigation_EN2013.pdf
- 시스코 DDoS 방어 가이드:
  - http://www.cisco.com/web/about/security/intelligence/guide_ddos_defense.html

5.4.1 위험

공용 서비스와 애플리케이션은 인터넷의 보안성 때문에 보안 인프라가 중요합니다. 무작위 DDoS 공격도 별다른 이유 없이 자주 일어납니다. "지하 시장"에서 몇 백 달러만 주면 DDoS 서비스를 살 수 있을 정도입니다. 그러므로, 침입자를 기업의 적이나 고객으로 한정할 수만도

없습니다. 무작위 공격이 일어나는 이유는 시스템을 다운시킬 수 있는지를 보기 위함입니다. 중요한 IoT 서비스가 예상치 않게 무력화되지 않도록 공격에 대비해두는 것이 좋습니다. 가용성은 IoT 제품이나 서비스의 생명과도 같습니다.

5.5 영구저장 모델 정의

요즘 컴퓨팅 환경은 컨테이너 기반 시스템이나 클라우드 환경처럼 수명이 짧은 경우가 많습니다. 그 결과 이들 시스템에 할당된 저장장치는 애플리케이션이 이들 기술을 항구적인 저장장치로 이용할 수 있을 만큼 크지도 않고 오랫동안 쓰도록 설계돼 있지도 않습니다. 또한 이들 시스템은 온디맨드 엔터티로 정의되기도 하며 중앙집중화의 모습은 찾아보기 어렵습니다. 다시 말하면, *어떤 시스템*에 항구적으로 사용 가능한 저장장치가 충분히 남아 있는지 다른 시스템이 알 길이 없는 것입니다.

이 같은 이유로 중앙의 저장시스템이 중요하며 안전하게 보호해야 하는 것입니다. 저장 시스템은 이런 환경에 존재하는 임시 시스템이라면 어느 것이든 액세스할 수 있어야 하므로, 수명이 짧은 서버나 서비스가 침해된 후 다른 서버나 서비스가 많이 사용하는 항구적인 저장 엔터티(또는 계층)에 액세스하는 상황이 벌어지게 됩니다. 이것은 공격자가 특정 네트워크를 횡적으로(또는 잠재적으로 수직으로) 침해하는 데 자주 이용하는 경로입니다.

이것을 차단하려면 서버나 서비스마다 항구적인 저장장치에 액세스를 하되 각자가 대표하는 애플리케이션, 무엇보다 애플리케이션이 대행하는 *고유한 엔드포인트나 파트너, 사용자*에 맞춰 정보를 저장해야 합니다. 이 항목의 마지막 부분이 가장 기본이 되는 항목입니다. 특정 *엔터티*를 대신해 항구적인 저장장치 액세스를 의무화하면 수명이 짧은 서버나 서비스가 데이터에 액세스하는 것을 제한할 수 있기 때문입니다.

다시 말하면, *수명이 짧은 시스템*을 침해한 공격자는 *수명이 짧은 그 시스템*과 연계된 ID 를 대신해 저장된 데이터에만 영향을 미칠 수 있습니다. 그 시스템이 단일 ID 의 데이터에만 액세스할 수 있다면 공격자는 이 시스템의 침해를 이용해 다른 계정으로 수평 이동하지 못하고, 그 단일 ID 에 대한 정보에만 액세스할 수 있습니다. 이렇게 되면 공격자가 취약점을 이용해 시스템을 크게 익스플로잇하는 능력이 크게 제한됩니다.

5.5.1 위험

안전한 항구적 저장 모델이 마련되지 않으면 고유한 사용자별 속성을 다른 자산에서 안전하게 분리하는 아키텍처는 존재하지 않게 됩니다. 그 결과, 공격자에게 어떤 저장 장치에 대한

액세스를 부여한 토큰이 침해를 당하면 여러 사용자의 데이터가 침해를 당할 수도 있습니다. 그러나 항구적인 저장 모델을 적용하면 침해를 한 사용자 또는 암호화된 데이터가 포함된 한 가지 저장 기술로 국한시킬 수 있습니다. 어느 경우든, 침해의 범위는 크게 줄어들어, 조직이 사용자와 회사에 대한 위협에 모두 대처할 시간을 더 많이 벌 수 있습니다.

5.6 관리 모델 정의

각 시스템은 관리자가 문제 해결과 애플리케이션 결함의 진단을 위해 액세스할 수 있어야 합니다. 이것은 서비스나 서버의 수명이 짧고 관리 모델이 충분하게 설계되지 않은 환경에서는 어려운 일일 수도 있습니다.

이것을 실현하기 위해서는 관리팀이 각 계층의 각 시스템과 어떻게 통신하는지 파악해야 합니다. 이종 시스템을 서로 분리하는, VPN 과 같은 인증 경계가 있어야 합니다. 관리팀은 각 계층을 통해 인증을 해야 합니다.

또한 관리자가 시스템과 어떻게 소통하는지도 찾아내야 합니다. 시스템의 모습을 VM 처럼 순간적으로 포착할 수 있는가? 단말기가 사용되는가? 원격 보안 쉘(SSH)이 시스템과의 소통에 사용되는가? CPU 활용도, 디스크 활용도, 네트워크 활용도 등 시스템 메트릭스의 모니터링과 분석을 위한 API 가 있는가? 그것을 문제해결이나 ID 이상에 적용할 수 있는가?

어떤 모델이든 다음과 같이 *반드시* 정의해야 하는 사항이 있습니다.

- 관리자는 어떻게 환경에 맞춰 인증을 할 것인가?
- 인증을 하는 관리자는 어떻게 물리적 ID 에 귀속될 수 있을까?
    - 2 요소 인증(2FA) 이용
- 시스템의 현재 모습을 어떻게 포착할 것인가?
- 변경은 어떻게 할 수 있고 어떻게 추적해야 하는가?

5.6.1 위험

관리를 위한 액세스의 경로를 제대로 갖추지 못한 환경은 대개 임시 방편을 이용해 생산 중인 시스템에 액세스하고 맙니다. 이렇게 되면 관리 포트가 공공의 접속, 또는 진단 서비스에 무방비 상태가 되면서도 3 자의 사용을 차단하지 못하게 되기 십상입니다. 명확한 관리 모델을 정립하면 애비뉴 공격자가 핵심 IoT 자원에 특권을 가지고 액세스할 가능성이 줄어듭니다.

5.7 시스템 로깅 및 모니터링 방식 정의

각 시스템은 모니터링을 통해 관리자와 정보기술(IT) 담당자가 이상을 탐지하고 진단해야 합니다. 모니터링은 다각도로 실시해야 한다. 예컨대, 인프라 수준에서 네트워크를 모니터링하면 네트워크 구성요소를 대상으로 한 애플리케이션 공격이나 DDoS 공격을 탐지할 수 있습니다. 계층 모니터링은 특정 애플리케이션이나 인프라가 침해를 당했는지 확인합니다. 반면 시스템 수준의 모니터링은 개별 애플리케이션 또는 애플리케이션 플랫폼이 공격을 받고 있거나 침해를 당했는지 판정합니다.

이렇게 하면 몇 단계의 모니터링을 하게 되며 정보를 한 리소스에 담아 감독팀에게 전달할 수 있습니다. 이 기술을 제공하고 메트릭스를 IT 전문가와 시스템 엔지니어가 사용할 수 있도록 시각적 시스템으로 변환하는 애플리케이션이 몇 가지 나와 있습니다.

공격 행동을 나타내는 이상 징후는 대표적으로 다음과 같습니다.

- 네트워크 트래팩 증가
- *이상한* 방향(특히 진출)으로 네트워크 트래픽 증가
- 진출이 필요 없는 리소스에서 진출 네트워크 트래픽 발생
- CPU 활용 이상
- 시각적 인터페이스가 없고 GPU 를 CPU 의 일부로 보유하고 있는 시스템에서 GPU 활용
- 디스크 또는 네트워크 저장장치의 활용
- 특정 호스트의 시스템 시각이 비정상적으로 변경

이상 상태를 포착하는 모니터링 시스템은 쉽게 구할 수 있지만, 그 맥락은 조직에서 이용 중인 애플리케이션이나 인프라마다 다를 수도 있습니다. 모니터링 시스템을 공급하는 업체와 협의하여 해당 구현자산에 대해 가장 효과적으로 메트릭스를 포착하고 해석하는 방법을 정하기 바랍니다.

계층마다 공격 또는 침해를 나타내는 이상징후는 다를 수도 있습니다. 계층별로 이상징후를 찾아내는 작업도 필요합니다.

본 권장사항과 더불어 다음 기관에서 발간한 자료를 참고하기 바랍니다

- Amazon EC2 Monitoring Documentation
  - http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/monitoring_ec2.html

- Google Cloud Monitoring
  - https://cloud.google.com/monitoring/
- Microsoft Azure Monitoring
  - https://azure.microsoft.com/en-us/documentation/articles/best-practices-monitoring/
- DigitalOcean Monitoring Tutorials (General)
  - https://www.digitalocean.com/community/tags/monitoring?type=tutorials

5.7.1 위험

기술을 모니터링하는 시스템은 IoT 보안 모델의 핵심 속성 가운데 하나입니다. 모니터링이 없다면 중요한 서비스 컴포넌트에서 취약성이 발견되었는지 확인할 길이 없습니다. 관리자는 모니터링을 통해 서비스와 인프라의 문제를 빠르게 진단하고 보안 사고와 소프트웨어 버그를 구별할 수 있습니다.

5.8 사고 대응 모델 정의

침해 또는 지속적 공격의 가능성을 탐지하는 것만으로는 부족합니다. 조직에서는 공격에 대응하고 맞서 싸워야 합니다. 시스템이 침해를 당하면 그 시스템을 치료하거나 꺼버리는 것만으로는 부족합니다. 조직에서는 침해의 원천을 찾아 시스템을 패치하고 기존 인프라 전반에 그 패치를 배포할 수 있어야 합니다.

이때 컨테이너 기반의 환경이 사용된다면 이 작업이 어려울 수도 있습니다. 복제된 애플리케이션이 취약한 구성을 간직한 채 실행되기 때문입니다. 애플리케이션 시스템은 "리부트" 또는 "업데이트" 이벤트를 탐지할 수 있어야 합니다. 이것은 애플리케이션 연결이 클라우드 내 다른 시스템으로 전파되거나 사용자가 강제로 로그아웃 돼 업데이트가 일어나게 하는 이벤트를 말합니다.

하지만 실행 모델이 무엇이든 엔지니어링 팀에서는 포렌식(forensic) 분석이 가능하도록 메트릭스를 포착할 수 있어야 합니다. 이 같은 정책과 절차는 항구적이어야 하며 법무팀(가능하다면 보험팀까지)에게 승인을 받아 해당 정보가 사법공무원(LEO)에게 적합한 방식으로 봉쇄되었는지 검증해야 합니다. 준법팀에서는 회사가 지방 및 연방법을 준수하도록 지원하는 데서 그치지 않고 법정에서 사용 가능한 침해의 예를 제공해야 합니다.

예가 입수되면 로그와 메트릭스, 그 외 문제의 이벤트를 입증할 수 있는 데이터를 중심으로 전체 시스템의 각 측면을 평가해야 합니다. 이때 데이터는 모두 안전한 시스템에서 포착하고 저장해 법률적 검토를 받아야 합니다.

본 권장사항과 더불어 다음 기관에서 발간한 자료를 참고하기 바랍니다.

- CERT Recommendations for Creating a CSIRT
  - http://www.cert.org/incident-management/products-services/creating-a-csirt.cfmhttp://www.cert.org/incident-management/products-services/creating-a-csirt.cfm

5.8.1 위험

사고 대응 모델이 없는 조직은 자원을 동원하고 침해된 시스템을 찾아내 치료하고 시스템을 검토해 정보를 확보하기까지 훨씬 더 오래 걸립니다. 또 패치를 만들고 시스템을 복구하는 속도도 크게 떨어집니다. 이 같은 무대책은 공격자에게 일부의 침해를 환경 전체로 확산시키기에 더할 나위 없이 좋은 무대를 제공합니다. 그 결과 늦은 대응 탓에 침해가 걷잡을 수 없이 커질 수도 있습니다. 조직에서는 사고에 즉각 대응할 수 있는 대비태세를 갖춰 공격자가 서비스의 핵심부를 장악할 수 있는 시간을 줄여야 합니다.

5.9 복구 모델 정의

보안 침해 또는 하드웨어 장애의 피해자가 사용자든, 애플리케이션이든, 복구는 필요합니다. 애플리케이션 계층 내에서 정보와 기능을 복구하는 절차를 마련해야 합니다. 그리고 이 절차는 각 애플리케이션과 계층의 환경에 따라 달라져야 합니다.

예컨대, 어떤 애플리케이션이 특정 작업의 출력에 관한 정보를 엔드포인트에서 수집했는데, 저장 오류가 발생해 애플리케이션이 해당 데이터의 출력을 항구적 저장장치로 취합하지 못하게 됐다면 그 애플리케이션은 다음과 같이 할 수 있습니다.

- 성공할 때까지 저장을 계속 (어쩌면 끝없이) 시도한다
- 성공 또는 실패 임계값에 도달할 때까지 한시적으로 저장을 시도한다
- 즉시 "fail" 상태가 된다. 메트릭스가 사라질 위험이 있다
- 엔드포인트에 같은 데이터를 다시 요청한다 (데이터가 없을 수도 있음)

이 중에서 애플리케이션과 비즈니스 애플리케이션에게 가장 적합한 방법을 선택해야 합니다. 이것 역시 애플리케이션의 상황에 따라 달라지며 특정 시스템 밖에서 모델링하기가 쉽지 않을 수도 있습니다.

엔지니어링과 회사 수뇌부가 참여해 고장난 또는 침해된 애플리케이션을 어떻게, 특히 사용자가 활동 중인 상황에서 어떻게 복구할지 결정해야 합니다.

공격자의 침해를 받았다고 확인된 시스템에 대해서는 복구 전에 해당 애플리케이션이나 시스템에 패치 작업이 충분히 실시되었음을 검증하는 모델이 반드시 있어야 합니다. 이 정책과 절차가 마련돼 있지 않으면 취약한 시스템이 다시 서비스 생태계에 배포돼 추가로 침해를 받을 위험이 있습니다.

5.9.1 위험

복구 모델은 정보와 애플리케이션, 구성이 바르게 복구되게 하는 역할을 합니다. 복원 모델이 없다면 팀에서 의도치 않게 취약한 서브시스템을 서버나 인프라에 다시 재배포할 수도 있습니다. 또한 데이터베이스나 저장 환경에서 공격자의 조종을 받은 하자 데이터가 여러 시스템으로 복제돼 맬웨어나 변경된 데이터를 전파할 수도 있습니다. 복구 프로세스는 공격자가 사고 복구 중 약점을 남용할 능력을 약화시킵니다. 복구 중에 사고가 발생하면 비용은 더욱 커지게 됩니다.

5.10 퇴역 모델 정의

조직에서 배포하는 시스템과 사용하는 계층은 어느 것이나 수명이 정해져 있습니다. 몇 십년 동안 같은 제품이나 서비스를 배포했더라도 그 제품이나 서비스를 구동하는 기술은 변하기 마련입니다. 그러므로 제품이나 서비스를 설계하고 구현하는 계획뿐만 아니라 해당 제품이나 서비스를 *퇴역*시키는 계획도 있어야 합니다.

이 프로세스를 통해 기술의 퇴역 과정에서 공격자가 특정 기술의 ID 를 입수하거나 시설을 이용하지 못하게 할 수 있습니다. 예컨대, 모회사가 어떤 회사를 인수한 후 특정 제품의 도메인을 그대로 둘 수도 있습니다. 해당 제품이 이름을 바꾸고 도메인을 모회사로 옮겨가면 공격자가 지금은 쓸모 없게 된 도메인을 장악할 수도 있습니다. 공격자가 그 도메인에 대해 크립토그래픽 인증서를 발급하고 그 도메인 아래서 배포된 기술과 계속 교류할 수 있다면, 그 제품이나 서비스의 퇴역 과정에서 절차의 결여로 보안상 큰 허점이 발생하게 됩니다.

어떤 제품이나 서비스의 아키텍처와 구현, 관리에 사용되는 각 기술은 목록화해 사용 가능성을 평가해야 합니다. 기술이 사용 가치를 잃게 되면 그 모델에 따라 퇴역시켜도 됩니다. 그러면 엔지니어링 팀과 회사 수뇌부에서 기초 플랫폼의 공백 없이 옛 기술을 버리고 더 적합한 기술을

도입해 나아갈 수 있습니다. 또한 파트너와 사용자에게 더 이상 제공되지 않을 기술을 사업 종료 후 공격자가 남용할 여지를 주지 않고 퇴역시킬 수도 있습니다.

5.10.1 위험

퇴역 절차가 없다면 경쟁사 또는 공격자에게 엔드포인트와 서비스 모두를 침해 당할 위험이 있습니다. 만일 어떤 조직에서 도메인이나 전화번호, 기타 재활용 가능한 서비스 등과 같은 객체에 대한 액세스를 풀어주고 공격자 또는 경쟁사에게 그 객체를 인수를 권리가 있다면 법적으로, 윤리적으로 문제가 없을 수도 있습니다. 이것은 장치나 서비스가 부도적한 남용, 심지어 악성 행동을 부추기는 결과로 이어질 수도 있습니다.

5.11 보안 등급 정의

파트너 조직과의 상호작용을 제대로, 효과적으로 관리하기 위해서는 보안 등급을 정의해야 합니다. 이것은 데이터 보안에 관한 조직 내부의 정책에 지침이 될 뿐만 아니라 파트너 조직이 사업 데이터와 자사 데이터, 고객 데이터에 적용하는 보안 수준을 정의하는 데 기준이 됩니다.

이 절차는 조사를 통해 조직에게 맞춰야 하겠지만, 데이터 보안 등급 정책은 대부분 다음 분류로 시작합니다.

- 공공 - 누구나 액세스 가능
- 대외비 - 사용자가 배포를 승인해야 함
- 비밀 - 특정 사용자에게 국한된 데이터
- 일급비밀 - 조직에게 국한된 데이터. 절대 유출 금지

기본 등급을 분류한 후에는 데이터 분류별로 보안 등급을 매겨야 합니다. 다시 말하면 이론이 아니라 실무에서 분류를 어떻게 사용할 것인지 평기해야 합니다. 회사*와*/엔지니어링의 관점에서 어떤 정책과 절차를 수립해야 하는지 정해야 한다는 것입니다.

그러면 조직에서는 기술적 정책을 수립할 수 있을 뿐만 아니라 기술적 요건을 지원하는 사업 정책을 집행할 수도 있습니다. 이 경우, 엔지니어링 팀이 고의든 아니든 정책의 위반을 꾀할 파트너와 내부 조직에게 기술적 요건을 전파하기도 더 쉬워집니다.

보안 등급을 표준화하고 하면 회사와 사용자의 프라이버시 요건이 보안 등급 모델에 어떻게 영향을 줄 수 있는지 평가해야 합니다. 조직은 시간을 들여서 프라이버스 모델을 보안 등급에 적용하고 사용자의 데이터에 의미를 부여해야 합니다. 또한 파트너가 사용자를 노출 위험에

빠뜨릴 수도 있는 어떤 자원에 액세스하기를 원할 때 사용자의 프라이버시를 보호해야 합니다. 파트너는 보안 등급에 프라이버시를 적용함으로써 민감한 데이터에 접근하고자 할 때 회사 수뇌부*와* 사용자의 동의를 구해야 합니다. 사용자는 본인의 민감한 데이터를 보호할 수 있는 선택권을 지녀*야 하*고 본인의 데이터가 3 자에게 노출되는 범위를 제한할 수 있어야 합니다.

5.11.1 위험

보안 등급 모델은 보안을 효과적으로 이용하는 솔루션의 구축에 중요합니다. 정보를 보호하려면 정보를 계량화해 상응하는 정책과 절차에 따라 적절한 통제 수단을 강구할 수 있어야 합니다. 그와 같은 모델이 없을 때 엔지니어링 팀에서는 위험에 대한 인식에 따라 때로는 너무 엄격하고 때로는너무 느슨한 보안 대책을 적용하게 됩니다. 엔니지어어와 회사 수뇌부를 포함해 팀 전체가 업무상 데이터의 중요도를 파악하고 예산상 적절한 통제 범위 안에서 어떻게 보호해야 할지 결정해야 합니다.

5.12 각 데이터 형식에 등급 정의

보안 등급을 수립한 후, 조직에서는 전체 IoT 제품이나 서비스에서 사용할 데이터의 유형을 정의해야 합니다. 그러면 조직에서 어떤 유형의 정보를 획득하고 생성해 IoT 시스템의 피어에게 배포할지, 데이터를 어떻게 취급해야 하는지 명확하게 규정할 수 있습니다. 이렇게 규정된 데이터는 IoT 환경에서 사용되는 전체 구성요소에 맥락과 가치를 부여하게 됩니다.

본 문서에서는 어떤 조직과 관련된 각종 데이터를 모두 모델링하지는 않습니다. 다만 몇 가지 유형을 예로 들면 다음과 같습니다.

- 사용자
- 조치
- 이미지
- 수정 가능한 문서
- 개인 식별 정보
- 보호대상 건강 정보

정보는 한 가지 이상의 *유형*에 속할 수도 있습니다. 그러나 데이터 자체는 한 가지 *보안 등급*에만 속해야 합니다. *유형*이 데이터의 속성과 처리 방식을 규정한다면, *보안 등급*은 해당 정보를 *어떻게, 어디서, 언제* 사용할 수 있고 *누구와* 공유할 수 있는지를 나타냅니다.

데이터의 유형을 정의하고 등급을 할당하는 작업은 결코 간단하지 않습니다. 그 과정에서 회사의 기준이 정해지며, 엔지니어링 팀에서는 데이터와 그 분류에 대해 기술적인 통제장치를 적용할 수 있습니다. 그러면 엔지니어링 팀과 회사 수뇌부가 나중에 파트너와 데이터 공유 및 처리에 방식에 관해 협상할 때 크게 도움이 됩니다.

5.12.1 위험

보안 등급과 마찬가지로, 통제장치도 데이터의 속성, 그리고 회사와 데이터의 관계를 계량화하지 않고는 구현이 불가합니다. 보안 등급에 따라 해당 정보가 시스템 안에서 어떻게 사용돼야 하는지, 보안 확보를 위해 데이터에 어떤 보호수단을 적용해야 하는지가 정해집니다. 보안 등급이 없을 때 엔지니어는 너무 엄격하거나 너무 느슨한 보안 대책을 적용하는 경향을 보입니다. 보안 대책은 엔지니어링 팀과 회사 수뇌부의 합의로 마련하여, 통제수단이 회사에 대한 데이터의 중요도와 균형을 이루도록 해야 합니다.

6 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.

6.1 명확한 인증 모델 정의

프라이버시 모델은 사용자의 정보가 파트너에게 제공되는 방식을 다루지만, 승인 모델은 회사 또는 파트너가 사용자를 대신해 어떻게 행위 해야 하는지를 규정합니다. 승인 모델은 예컨대 파트너의 메트릭스가 가정 내 난방이나 냉방의 사용을 최적화할 수 있는 홈 오토메이션 시스템에서 편리하게 적용됩니다. 승인 모델 하에서는 파트너가 특정 메트릭스가 탐지됐을 때 해당 사용자의 집에 대한 난방 또는 난방 제어를 바꿀 수 있습니다.

이를 위해서는 세분화된 승인 기능과 그것을 파트너에게 배포하는 방법을 설명하는 유사한 GUI 가 있어야 합니다. 사용자가 요구 받은 특정 기능에 대해 액세스를 허용하거나 취소할 수 있어야 합니다. 또 취소된 기능이 즉시 효과를 발휘해 남용의 여지를 없애야 합니다.

시스템을 철저히 모니터링하여 파트너가 허가 받지 않은 조치를 취하지 못하게 해야 합니다. 승인 모델을 세분화하여 제어하면 파트너에게 어떤 기능에 대한 액세스를 어떤 빈도로 허용할지 사용자가 구성할 수 있습니다. 이 같은 속성은 사용자가 남용하는 파트너 또는 침해 받은(해킹 당한) 파트너에게서 자신의 시스템을 지키는 능력을 높여줍니다.

6.1.1 위험

승인 모델이 없다면, 3 자가 사용자의 기능에 액세스하는 것을 막을 방법이 없습니다. 그러면 불량한 3 자 또는 침해를 당한 3 자가 사용자의 기술이나 데이터에 제한 없이 액세스할 수도 있습니다. 승인 모델을 만들면 사용자가 허락하는 속성으로 액세스가 제한됩니다. 그 결과 사용자는 3 자에게 제공할 기능과 데이터에 대해 더 큰 통제력을 갖게 되며, IoT 서비스 업체는 침해의 확산 가능성을 낮출 수 있습니다.

6.2 크립토그래픽 아키텍처의 관리

IoT 환경에 배포된 기술은 기초적인 저출력 엔드포인트이든, 대규모 클라우드 서비스든 예외 없이 크립토그래피(cryptography)를 사용해야 합니다. IoT 제품이나 서비스에서 보안을 제대로

구현하려면 갈수록 진화하는 사양에 맞춰 사용되는 크립토그래피 기술을 적절히 설계하고 관리하고 조정해야 합니다.

엔지니어링 팀에서는 다음을 파악해야 합니다.

- 크립토그래픽 알고리즘이 낡지 않았는가?
- 사용 중인 크립토그래픽 키의 비트 길이는 적절한가?
- 해시 알고리즘이 충돌 공격에 취약한가?
- 강력한 무작위 번호 생성기가 사용 중인가?
- 메시지에 무작위 데이터가 충분히 들어 있는가
- TLS 등 크립토그래픽 프로토콜이 모범 사례를 반영하고 있는가?
- 순방향 보안 같은 프라이버시 중심의 개념이 사용되고 있는가?
- 평문 비밀번호나 핀 번호가 네트워크를 통해 전송되고 있는가?
- 비표준 크립토그래픽 알고리즘이 사용되었는가?

위의 각 항목은 IoT 제품이나 서비스 안에서 고품질의 크립토그래픽 아키텍처를 유지하는 데 중요합니다. 크립토그래피 솔루션의 성공적 배포는 복원력이 높은 크립토그래픽 솔루션을 이용해 복원력이 덜한 솔루션을 이용하는 기술에 패치를 배포하는 엔지니어링 팀의 능력이 관건입니다.

예컨대, RC4 알고리즘이 최근에 심각한 보안 결함이 있음이 밝혀졌는데, RC4 를 사용하도록 구성된 클라이언트에 패치를 안전하게 배포하여 RC4 를 AES-256 으로 대체할 수 있다면 RC4 는 크게 걱정할 문제가 아닙니다. 회복력이 더 좋은 기술, 예컨대 Ephemeral Diffie Hellman 와 대칭 키 또는 UICC 보안 키를 이용해 상호 인증이 실시된다면, 그 패치는 취약한 크립토그래픽 알고리즘을 사용하지 않고도 검증이 가능합니다.

사용자나 엔드포인트에서 사용하는 비밀번호와 핀은 절대로 평문으로 네트워크르 통해 전송하면 안 됩니다. 통신 채널이 암호화를 통해 보호되더라도 마찬가지입니다. 대신 비밀번호나 핀의 크립토그래픽 해시를 이용하여 크립토그래픽 터널에 구성 오류가 있더라도 비밀번호 자체가 노출되는 일이 없게 해야 합니다. 해시는 비밀번호로써 생성해야 하며 고유한 일회용 토큰을 하나 이상 갖춰야 합니다. 이 토큰은 네트워크 세션에서 흔히 가져오는데, 엔드포인트와 서비스 인프라 모두에 저장된 롤링(rolling) 코드에서 값을 취하는 것이 더 안전합니다. 이렇게 하면 네트워크에서 특권을 지닌 공격자라 할지라도 강제 서명 공격으로 이어질 수도 있는, 유익한 값을 지닌 해시를 만들지 못합니다.

커스텀 크립토그래픽 알고리즘(자체 설계된 알고리즘)은 절대로 사용하면 안 됩니다. 크립토그래퍼가 개발하고 크립토그래픽 보안에 정통한 감독기관에서 권장한 권장 알고리즘만 써야 합니다. 그리고 설계가 허술한 알고리즘이나 낡은 알고리즘, 압축, 바이너리 알고리즘, 기타 크립토알로리즘으로 자주 오인되는 알고리즘(LZO, base64, ROT13, XOR 등)은 피해야 합니다.

이 주제에 관해 더 자세한 사항은 다음 가이드와 문서를 참고하기 바랍니다.

- ISO 18033-1:2015 – Encryption Algorithms
- ISO 18033-2:2015 – Asymmetric Ciphers
- ISO 18033-3:2015 – Block Ciphers
- www.owasp.org/index.php/Guide_to_Cryptography
- csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf
- csrc.nist.gov/groups/ST/toolkit/key_management.html

6.2.1 위험

크립토그래픽 아키텍처가 포함된 솔루션을 알맞게 배포하면 사용되는 알고리즘과 프로토콜, 비밀이 모두 현 권고사항에 부합합니다. 또한 권고사항은 시간이 지남에 따라 바뀝니다. 크립토그래픽 아키텍처가 없다면 보안에 허점을 유발하는 낡은 기술이 어느 것인지 파악하기가 더 어려울 것입니다.

6.3 통신 모델 정의

서비스 생태계의 각 시스템은 상호 인증을 할 수 있어야 합니다. 이 생태계에 속하는 컴퓨팅 플랫폼 중 어느 것도 무명의 공용 사용자에게 접근을 허용하면 안 됩니다. 엔드 포인트, 파트너, 사용자 각각이 상호 인증을 요하는 기술을 통해 서비스 생태계와 통신을 해야 합니다. 사용자 인터페이스를 구성하는 서비스는 대개 별도 환경에서 배포되고 관리되므로 대중이 액세스하는 인터페이스는 그 공간으로 한정해야 합니다. 반면 서비스 생태계에는 인증 받은 리소스에게 서비스를 배포할 때 사용되는 시스템이 모두 들어 있습니다.

여기에는 시스템이 프로비저닝하지 않은 엔드포인트도 포함됩니다. 하드웨어가 그 제작과 개인화 공정에서 기업에서 배포한 리소스로 *인증*받을 수 있을 만큼 적절하게 구성되어 *야 하기* 때문입니다.

따라서 통신 모델은 다음 기능을 갖춰 *야 합*니다.

- 상호 인증
- 기밀유지
- 무결성

위를 효과적으로 성취하기 위해 통신 모델에게는 다음과 같은 덕목이 필요합니다.

- 중앙집중식 신뢰 기반 또는 대안으로 분산된 신뢰 기반
- ID 프로비저닝과 취소
- 완전 순방향 기밀성(PFS)

통신 모델의 각 엔터티가 피어와 같은 기관에게 승인을 받도록 신뢰기반을 이용해야 합니다. 이렇게 되면 모든 엔터티가 중앙의 한 기관에게 프로비저닝과 인증을 받게 할 수 있습니다. 이와 같은 신뢰 기반을 실현하는 기술은 중앙집중식(TLS 인증서와 유사)이어도 되고 분산 방식(비트코인 블록체인 기반의 IoT 모델과 유사한 것, 예컨대 IBM/삼성의 ADEPT 프로젝트, Tilepay 등)이어도 됩니다. 단, 중앙에 있는 기업은 한 곳이 모델의 *소유자*가 되어 프로비저닝 시스템을 보호해야 합니다.

이 통신 모델에는 프로비저닝과 취소 기능을 넣어 침해 당한 비밀이나 ID 를 적은 수고로 시스템에서 제거할 수 있게 해야 합니다. Online Certificate Security Protocol(OCSP)가 같은 기술이 이 과정에서 유용하게 쓰일 수 있습니다.

통신 프로토콜은 *과거*의 통신을 침해할 여지를 최소화할 수 있는 기술을 이용해 *야 합니다*. 이것은 통신 비밀의 교환에 이용되는 일회성 비대칭 크립토그래픽 키를 만들면 가능합니다. 인증서가 침해를 당해도 일회성 비밀은 그렇지 않습니다. 이렇게 되면 암호화된 메시지를 오래 보관해도 공격자가 그것을 해독할 위험은 없습니다. 설령 인증서로 보호되는 비밀이 침해를 당하거나 노출되더라도 마찬가지입니다.

통신 보안의 난제는 기술의 구현과 수명에 있습니다. 권위 있는 기관에서 높은 신뢰도로 보유하는 암호화 알고리즘을 선택하면 장애 가능성을 낮출 수 있습니다.

엔지니어링 당국에서 설계했거나 승인한 라이브러리와 알고리즘 구현체를 이용해 *야 합니다*. 커스터마이징 방식으로 구현된 알고리즘은 사용하면 *안 됩니다*. 그런 알고리즘은 엔지니어링 팀의 업무를 방해할 뿐만 아니라 허술하게 설계됐거나 잘못 구현된 시스템 때문에 알고리즘이 크립토그래픽 차원에서 약화될 가능성을 야기합니다.

본 권장사항과 더불어 다음 기관에서 발간한 자료를 참고하기 바랍니다.

- CafeSoft Apache Mutual Authentication How-To Guide:
  - http://www.cafesoft.com/products/cams/ps/docs32/admin/ConfiguringApache2ForSSLTLSMutualAuthentication.html

6.3.1 위험

통신 보안은 IoT 의 근간입니다. 통신 보안이 없다면 내장된 장치들이 올바른 백엔드 서비스와 통신을 하도록 보장할 길이 없습니다. 이것은 텔레매틱스, 의료기기, 산업용 제어 시스템와 같이 장치에 보내는 명령을 안내하고 구성하고 전송하는 핵심 서비스에게 중요합니다. 통신 보안이 없다면 명령어가 엔드포인트에 제대로 전달되고 있는지 확인할 길이 없습니다. 원하는 피어와 메시지를 정확하게 주고 받을 수 있도록 통신 보안을 의무화해야 합니다.

6.4 네트워크 인증 서비스 이용

네트워크 운영자가 파트너 역할을 할 때에는 그 운영자만의 토큰을 이용해 사용자 인증을 실시합니다. 이들 토큰은 네트워크 운영자의 UICC 에 있으면서 네트워크 수준에서는 사용자를 인증하지만 애플리케이션 수준에서는 인증을 하지 않습니다. 다음 기술을 이용한다면 네트워크 인증이 더 간편해질 수도 있습니다.

- Generic Bootstrap Architecture (3GPP TS 33.220)
- M2M SM (ETSI TS 102 921)

인증 기술이 인증 목적으로 애플리케이션 수준에서 유의미한지 평가해야 합니다. 그 토큰을 보안 저장매체로 쓸 수 있다면, 그 장치를 물리적 엔드포인트의 인증 레이어로 이용해 그 토큰을 이용하는 TCB 를 구축할 수 있는지 판단해야 합니다.

네트워크 기반 인증을 의무화하는 운영자는 많지만 이 API 에 대한 액세스를 허용해 사용자나 엔드포인트를 인증하는 것은 신기술이나 다름 없습니다. 협업 중인 네트워크 운영자가 이 공간에서 의미 있는 환경을 만들고 있는지 평가해야 합니다. 그렇다면, 이 기술을 네트워크 수준의 인증 토큰 이상으로 사용하는 방안을 검토해야 합니다. 여러 기술보다 하나의 보안 저장 기술을 활용하는 편이 더 수월할 수도 있기 때문입니다.

6.4.1 위험

네트워크 인증 서비스에 트러스트 앵커가 포함돼 있는데도 이 서비스를 애플리케이션 보안에 활용하*지 않는다면* 애플리케이션이 안정적으로 사용자를 인증하는 능력이 제약을 받게 되고

기초 엔드포인트 플랫폼의 비용이 늘어나게 될 것입니다. 이는 다시 배포 비용의 증가와 조직이 네트워크 운영자에게 입수할 수 있는 정보의 감소로 이어질 것입니다.

6.5 가능할 경우 서버 프로비저닝

서버 프로비저닝은 서버를 생산 환경에서 정의하고 구성하고 개인화하고 배포하는 과정입니다. 서비스의 관점에서 프로비저닝 과정은 서버가 보안을 강화하고 적대적일 가능성이 있는 환경에 배포될 준비를 마치는 수단입니다.

클라우드 인프라, 전용 호스팅 업체, 회사의 개인 랙 공간 등 어디에 배포되든 서버는 내부자와 외부자 모두에게 취약합니다. 그렇다면 서비스 인프라에 배포되기 전에 공격에 대한 면역력을 키워야 합니다.

이를 위해서는 먼저 주변 환경에 제공해야 하는 서비스를 파악해야 합니다. 서버가 존재할 환경이 공용인지 사설인지, 그것이 서버 보안이라는 맥락에서 무슨 의미를 지니는지 정의해야 합니다. 서버에서 작동하는 각 서비스가 대중에게 개방되는 것인지 인증 받은 클라이언트만 서비스에 접속할 수 있는 것인지 판단해야 합니다.

서버에서 실행될 운영체제의 수명주기를 평가해야 합니다. 어떻게 소프트웨어 업데이트를 적절히 관리하여 보안 패치를 신속하게 배포하고 작동 중인 서버에 확약할지 결정해야 합니다. 업데이트가 실패하거나 생산 서비스에 예기치 않은 문제를 야기할 경우 롤백 모델을 평가해야 합니다. 일부 라이브러리나 애플리케이션 업데이트가 의도치 않게 부작용을 유발할 수도 있기 때문입니다.

마지막으로, 프로비저닝된 서버의 퇴역 모델을 평가해 가장 안전하게 시스템에서 자산을 덜어내는 방법을 찾아내야 합니다. 이상 서비스나 클라이언트의 거동을 평가할 때 필요할 수도 있는 시스템 로그도 그 중 한 가지입니다.

이 같은 권고사항이 의미하는 바는 조직에서 패치 관리 프로세스를 구현해 취약한 서비스를 찾아내고 패치를 배포하고 패치 구현의 성공 여부를 모니터링 해야 한다는 것입니다.

패치 관리에 관해서는

- http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-40r3.pdf 도 참고 바랍니다.

6.5.1 위험

서버 프로비저닝은 IoT 환경의 전체적인 보안에서 중요한 부분을 차지합니다. 서버 프로비저닝이 없다면 서버 아키텍처에 대한 조직의 통제력은 크게 약해질 것입니다. 그러면 아키텍처 사양이 결여로 보안사의 허점이 나타날 수도 있습니다. 사양이 없다면 조직은 배포된 기술이 최신 모범 사례에 부합하는지 판단할 수 없게 됩니다. 또한, 이 같은 기술을 개선하려면 배포된 시스템 각각을 조사해 배포된 자산 간의 델타를 평가해야 하는데, 중요한 보안 업데이트를 배포해야 할 경우 평가의 효율이 떨어지고 큰 우려의 대상이 됩니다. 서비스를 정의할 때 일관성과 아키텍처가 없다면 하나하나 일일이 점검하지 않고는 즉시 관심이 필요한 시스템을 쉽게 찾아낼 방도가 없습니다.

6.6 업데이트 모델 정의

실행 환경이나 애플리케이션 이미지, TCB 를 업데이트하는 일은 지난한 과정입니다. 다음과 같은 모델이라면 이 과정을 간소화할 수 있습니다.

- 실행 플랫폼의 각 레이어에 대해 새 애플리케이션 이미지의 고유 URL 등 네트워크 이미지를 정의합니다
- 레이어 별로 서명 키를 만듭니다.
- 각 레이어에서 새롭게 허가 받은 버전에 대해 그 레이어의 이미지를 만듭니다
- 레이어 이미지에 이미지를 설명하는 메타데이터(버전, 타임스탬프, 정체성 등)를 넣습니다
- 레이어 이미지에 서명 키로 서명을 합니다
- 이미지와 서명, 공용 키를 고유 네트워크 리소스 또는 업데이트 서비스를 통해 제공합니다

새 시스템은 배포 시 다음과 같은 능력을 갖춰야 합니다.

- 레어이 별로
    - 배포할 버전 불러오는 능력
    - 이미지를 크립토그래픽 방식으로 검증하는 능력
    - 시스템에 이미지 레이어를 배포하는 능력

애프리케이션에는 개인 비밀을 저장하면 안 됩니다. 비밀은 시스템이 배포될 때마다 역동적으로 프로비저닝하여 각 시스템을 개인화해야 합니다. 이 아이덴티티는 시스템이 퇴역하면 취소해야 합니다. 그 시스템의 수명에는 구애 받지 말아야 합니다.

이 권고의 의미는 패치 관리 프로세스를 동원해 인프라 내의 서비스와 기술을 관리해야 한다는 것입니다.

자세한 사항은 다음 문서를 참조하기 바랍니다.

- http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-40r3.pdf

6.6.1 위험

잘 정의된 업데이트 모델이 없다면 서비스와 애플리케이션은 업데이트 절차의 남용을 통한 침해의 위험에 노출됩니다. 공격자가 자체 애플리케이션을 업데이트 프로세스에 주입하여 클라우드 시스템과 기타 서버에 자기 소프트웨어를 이식할 수도 있습니다. 통신 보안 인프라의 보안이 허술하면 DNS 같은 네트워크 서비스를 조작하는 것만으로도 쉽게 가능한 시나리오입니다. 과거에는 BGP(Border Gateway Protocol) 공격과 같이 더 고도의 공격으로 보안이 허술한 서비스를 침해한 사례도 많았습니다.

6.7 노출된 데이터를 대상으로 위반 정책 정의

데이터에 정책과 절차를 정의하는 것만으로는 부족합니다. 파트너가 데이터를 노출했는지 여부를 탐지하는 모델도 있어야 합니다. 조직에서는 파트너가 사용자의 데이터와 프라이버시를 보호하기 위해 수립된 기술적 통제장치나 정책에 반하는 행위에 가담했는지를 평가하는 계획을 마련해 둬야 합니다.

이를 위해서는 엔지니어링 팀이 단순히 사용자 데이터가 아니라 *보안 등급*에 적용되는 모니터링 및 로깅 기술을 규정해야 합니다. 이렇게 되면 정보뿐만 아니라 정보의 *분류*에도 감사를 적용할 수 있습니다. 그러면 사용자 정보가 노출되었을 때 조직이 자기를 방어하기에도 수월합니다. 조직은 개인 정보를 보호하기 위한 보안 등급과 기술적 통제장치가 마련돼 있고 정책에 따라 관리, 저장, 배포되고 있음을 보여줄 수 있어야 합니다.

조직에서 모니터링과 로깅 기술을 적용해 파트너의 보안 등급 규칙 위반을 입증할 수 있다면 더 좋습니다. 수뇌부는 그 시점에 파트너에게 벌금을 부과할지, 관계를 청산할지 등을 결정해야 합니다.

6.7.1 위험

위반 정책이 없다면 3 자가 노출한 데이터에 대한 책임을 면할 법적 장치는 전무한 상황이 됩니다. 회사가 노출된 데이터의 원천이고 3 자가 그 데이터를 잃어버렸다고 해도 넘겨준 데이터에 대해 책임은 회사에게 돌아갑니다.

위반 정책이 있다면 파트너는 제공 받은 데이터에 대해 일정 수준의 보안을 유지할 의무를 지게 됩니다. 보안 위반이 발생하더라도 자체 보안 요건만 준수한다면 IoT 서비스 업체는 책임을 면합니다. 그렇다면 정책의 준수는 파트너의 책임이 됩니다.

정책은 법무 및 보험팀의 검토를 받아 모델이 엄격한 보안 정책 및 절차를 통해 조직의 책임을 줄이도록 돼 있는지 확인해야 합니다. 기업에 따라 제품과 서비스의 속성상 법규 적용의 면제를 받지 못할 수도 있습니다.

6.8 서비스 생태계를 통해 인증 의무화

사용자 인터페이스가 사용자를 직접 인증하면 *안 됩니다*. 시스템은 항상 중앙에서 제공하는 서비스를 통해 사용자를 인증해야 합니다. 단, 로컬 패스코드의 보호를 받는 모바일 장치에서 구동하는 애플리케이션에 한해 예외로 합니다. 이 패스코드는 로컬 애플리케이션의 접속에 사용해도 됩니다. 그러나, 원격 서비스와 리소스에 대한 액세스는 별도 인증 통큰으로 검증해야 합니다.

다만 사용자에게 이 인증 방법에 수반하는 위험을 충분히 설명한다면 엔지니어링 팀에서 사용 편의를 이유로 두 인증 제도를 하나로 결합해도 됩니다. 이 경우 인증 받은 사용자의 로컬 애플리케이션 비밀번호가 원격 서비스를 대상으로 하는 인증 토큰이 들어 있는 로컬 데이터베이스를 해독할 수 있게 됩니다. 이 다단계 인증 모델이라면 대부분의 사용자에게 충분합니다.

다만, 중앙 인증 서비스는 먼저 사용자를 로컬 애플리케이션에 인증하고 인증 토큰의 사용 방법과 기간에 관한 정책과 절차를 의무화해야 합니다. 메트릭스도 수집하여 사용자가 다른 컴퓨팅 플랫폼으로 이전했는데도 같은 토큰을 사용하고 있는지도 판단해야 합니다. 사용자가 단기간에 한해 다른 곳에서 같은 토큰을 사용하고 있을 수도 있습니다. 이동의 유형과 속도에 따라 이들 메트릭스에서 토큰의 침해 가능성이 드러날 수도 있습니다. 그렇다면 해당 토큰은 무효화해야 하고 사용자는 어쩌면 다단계 인증을 통해 다시 로그인해야 합니다.

6.8.1 위험

아키텍처가 아무리 안전해도 엔드포인트 시스템에는 남용 가능성이 있으므로 백엔드 시스템의 확인 없이 사용자를 인증하는 것은 항상 미덥지가 않습니다. 이것은 사용자가 본인의 자격증명(credential)을 업데이트하지 않았거나 여러 유형의 장치에 자격증명을 분산해 둘 수도 있다는 가정 때문입니다. 이는 효율과는 거리가 멀며 침해 당한 장치가 사용자 자격증명의 옛 버전을 이용할 여지를 남겨두는 것입니다.

6.9 입력 검증 실시

엔드포인트나 사용자, 자칭 사용자에게 획득한 데이터는 모두 이상 여부를 분석해야 합니다. 공격자에게 가장 손쉬운 경로는 항상 사용자 인터페이스에서 속하는 서비스에서 웹 애플리케이션 입력을 남용하는 것입니다. 이는 이 기술이 사용자마다 다른 지역색과 인코딩, 기타 파라미터에 따라 역동적으로 정보를 제공해야 하기 때문입니다. 숙련된 사용자는 인코딩의 일부 속성을 조작해 프로세싱 서브시스템의 여러 레이어에서 공격자에게 유리한 부작용을 유발할 수도 있습니다.

예를 들면, 정교한 공격에는 무효(null) 바이트를 더 높은 수준의 언어에서 스트링으로 처리되는 메시지로 인코딩합니다. 일부 높은 수준의 언어는 무효 바이트를 구분자가 아니라 바이너리 스트링의 일부로 받아들입니다. 이 바이너리 스트링이 낮은 수준의 라이브러리로 전달되면 내장된 무효 바이트는 스트링 구분자로 해석되고 잘려 애플리케이션이 의도한 것과는 완전히 다른 무언가를 의미하게 됩니다. 과거에는 이것이 특정 사용자에게 제한된 파일 시스템 리소스에 접근하는 기발한 방법이었습니다.

악성 입력의 형태는 무궁무진하지만, 엔지니어가 건마다 테스트할 필요는 없습니다. 프로세트는 꽤 단순합니다.

- 데이터를 내부적으로 어떻게 활용할지 결정한다
- 내부 사용 모델을 준수해야 하는 인코딩과 문자에 관해 정책을 의무화한다
- 이 정책에 따라 데이터를 분석하는 API 를 설계한다
- 데이터가 모델에 어긋난다고 밝혀지면 예외를 발동한다
- 이 세션에 관하여 메타데이터로 이벤트를 내부적으로 기록해 공격적 행동을 탐지한다

시스템에 저장된 데이터는 모두 먼저 처리해 정적인 모델로 구성해야 합니다. 효과적인 방법은 데이터를 모두 기본 64 알고리즘으로 인코딩한 후 데이터베이스에 넣는 것입니다. 그러면 데이터가 데이터를 조작할 염려가 없습니다.

6.9.1 위험

입력 검증 기능이 없는 시스템은 SQL 인젝션(SQLi) 등 OWASP 톱 10 에 인용된 문제를 비롯해 각종 공격의 대상이 될 수 있습니다. 원격 코드 실행 공격도 마찬가지입니다. 남용의 범위가 워낙 넓기 때문에 위험을 여기서 제대로 계량하기는 어렵습니다. 입력 검증은 클라우드 서비스든 엔드포인트에서 실행되는 애플리케이션이든 보안 애플리케이션을 가리지 않고 중요한 속성입니다.

6.10 출력 필터링 실시

출력 필터링은 입력 검증을 보완하는 수단입니다. 이 프로세스를 통해 표현 레이어를 공격자의 조작으로부터 보호하고 시스템이 특권을 지닌 것으로 간주되는 사용자에게 정보를 제공하지 못하게 할 수 있습니다.

전자의 경우, 표현 레이어가 제공할 데이터는 모두 서비스 레이어를 떠나기 전에 평가를 받아야 합니다. 이렇게 하면 표현 레이어에 인코딩되는 데이터, 예컨대 JSON 메시지 또는 인코딩된 자바스크립트에 해당 데이터의 표현을 망가뜨리거나 무효로 만들 수도 있는 형식이 들어가지 않습니다. 이는 시스템에 저장된 문자 중에서 렌더링 됐을 때 표현 모델을 망가뜨릴 가능성이 있는 것은 걸러내거나 표현에 영향이 없도록 인코딩해야 한다는 뜻입니다.

이 문제를 해결하는 방법은 제한된 문자를 걸러내거나 모든 문자에 인코딩을 의무화하여 이들 문자의 표현이 GUI 를 바꾸지 못하게 하는 것(문자는 렌더링 엔진이 제어 코드로 인식하지 않음)입니다. 그냥 메시지를 표시하지 않는 방법도 있습니다. 이 방법 중 어느 것을 선택해도 효과는 있지만 상황에 따라 적합한 것은 달라집니다. 메시지 포럼의 예와 같이 공격자가 스크립트를 올리고 다른 사용자가 아무런 생각 없이 그것을 복사해 실행한다면 상황은 매한가지가 됩니다. 그러므로 HTML 이나 기타 스크립트를 표현 레이어에 주입하지 않는 방법으로 정보를 렌더링하기보다는 정보를 스크럽(scrub)하여 다른 사용자가 영향을 받지 않게 해야 합니다.

데이터를 사용자에게 다시 보여주면 안 되는 경우라면, 이것은 공격자가 저장하고 렌더링한 데이터와는 관련이 없습니다. 이 문제는 오히려 대중이 소비하기에는 적합하지 않은 데이터의 구현과 관련이 있으며 관리자와 엔지니어 전용으로 놔둬야 합니다. 예를 들어, 정보 처리 과정에서 내부 에러가 생성된다면 그 에러를 디버그 데이터와 함께 사용자에게 렌더링하면 안 됩니다. 사용자가 버그를 찾아 수정하여 애플리케이션의 약점을 익스플로잇 할 수도 있기 때문입니다. 이 정보는 내부에서 로그해야 하며, 버그를 남용할 수 있을 만큼 컨텍스트를 많이 주지 않는 사용자에게 일반 오류를 제기해야 합니다. 사용자가 버그를 복제할 수 있더라도 익스플로잇 방법의 개선이 뚜렷한 애플리케이션에서 나온 출력의 차이를 평가할 수 있으면 안 됩니다.

6.10.1 위험

출력 검증은 IoT 보안에서 중요한 특성입니다. 출력 검증을 하지 않는 시스템에는 중요한 사용자 데이터와 프라이버시 관련 데이터, 진단 데이터의 노출, 장황한 에러 메시지 등과 같은 위험이 따릅니다. 이들 메시지는 사용자 정보를 노출하거나 네트워크 서비스를 대상으로 익스플로잇을 유발할 수도 있습니다.

6.11 강력한 비밀번호 정책 적용

인증 시스템은 강력한 비밀번호를 의무화해 사용자 인증에 비밀번호를 사용하게 해야 합니다. 비밀번호의 복잡도는 정보 보안 연구자와 엔지니어, 회사 수뇌부 집단 간에 끊임 없는 충돌의 대상이었습니다. 회사 수뇌부는 사용자가 비밀번호를 쉽게 기억하기를 바랄 때가 많습니다. 엔지니어들은 인터페이스의 복잡도를 낮추고 싶어 합니다. 표현 레이어의 설계자들이 특히 그렇습니다. 정보보안 전문가들은 공격자의 능력을 과대평가해 특정 기술에 쓸 데 없이 복잡함을 강요하기도 합니다.

하지만 정답은 각 집단이 요구하는 것의 중간 어디쯤입니다. 비밀번호는 의무적으로 길어야 하지만 복잡해서는 안 됩니다. 여덟 자리 비밀번호가 기본일 때가 있었고, 본 문서의 작성 시점에는 여섯 자리를 허용하는 시스템도 일부 존재하며 비밀번호 길에는 최신 모범 사례 표준을 따라야 함에도 비밀번호는 여덟 자리보다 훨씬 더 길 가능성이 높습니다. 비밀번호가 길어지면 복잡도 요건은 그 만큼 간소해집니다. 사용자로서는 여러 가지 문자를 복잡하게 조합하지 않고 그냥 문장 하나를 기억해도 됩니다. 공백과 대/소문자, 숫자, 구두점을 이용할 수

있기 때문에 공격자가 무작위 대입공격(brute-forcing)을 적용할 수 없을 정도로 복잡도가 커지게 됩니다.

여기서 잊지 말아야 할 사실. 공격자가 비밀번호를 침해하는 방법은 일반적으로 다음 네 가지입니다.

- 비밀번호 데이터베이스를 털어 개별 비밀번호 깨기
- 애플리케이션 인증 서비스에 대한 무작위 대입공격
- 맬웨어 설치
- 하드코드 또는 기본 비밀번호

긴 비밀번호를 의무화하면 첫 번째 위험이 줄어듭니다. 하지만 서비스 생태계 레이어의 보안이 훨씬 더 이익입니다. 공격자가 일차적으로 비밀번호 데이터베이스를 불러올 수 없어야 하는데, 이는 두 번째 항목과도 연관됩니다.

애플리케이션 비밀번호의 무작위 대입공격은 공격자가 가장 효과적으로 비밀번호를 남용하는 방법입니다. 이 가능성은 잘 설계된 인증 서비스로 크게 낮출 수 있습니다. 틀린 비밀번호가 하나 추측되었다면 시스템은 자동으로 추측 사이 지연의 크기를 늘리기 시작해야 합니다. 그 다음 시도의 총 회수를 제한하는 임계값을 정의해야 합니다. 공격자가 이 임계값에 도달하면, 해당 계정은 잠기고 2 요소 인증이나 기타 방법을 이용해 사용자가 본인 계정을 해제하고 검증을 받아야 합니다. 이 같은 보안 유형은 네트워크 기반의 공격에서 오는 효용을 크게 줄입니다. 이는 마지막 항목과 연관이 있습니다.

클라이언트 시스템의 맬웨어는 컴퓨팅 플랫폼이나 대응 기술을 설치하는 사용자가 처리해야 하는 대상입니다. 일반적으로 애플리케이션 그 자체가 보안을 확보해야 하는 것은 아닙니다. 애플리케이션 수준에서는 2FA 를 의무화하는 것이 외에 이 위험에 맞서기 위해 할 수 있는 것이 없다시피 하므로, 이것이 공격자의 *유일한 공격 방식*이라면 애플리케이션 엔지니어는 인증 시스템에 대한 비밀번호 공격의 위협을 크게 낮추게 됩니다.

그러나 이 권고사항의 구현에 대한 *보상은 높지 않습니다*. 비밀번호 인증에 대한 공격 가능성을 낮추기 위해 어떤 기술을 이용하든 비밀번호는 기본적으로 무형의 자원이기 때문입니다. 비밀번호는 한 사람만 가질 수 있는 실물 토큰이 아닙니다. 그보다는 컴퓨터 시스템에서 시각적 관찰을 통해 얼마든지 복사 가능한 추상적 물체에 가깝습니다. 따라서 어떤 식으로도 특정 사용자를 적절히 표시하지 못하는, 대단히 취약한 인증 수단입니다. 그러므로 비밀번호 자체가 약점이며, 비밀번호를 사용하는 기술은 비밀번호에 내재하는 위험에 노출돼 있습니다.

비밀번호는 시스템에서 절대로 하드코드화하면 안 됩니다. 엔드포인트에 대해서는 고유한 크립토그래픽 키를 생성해야 합니다. 엔드포인트 프로비저닝에 관해서는 엔드포인트 문서를 참고 바랍니다. 서비스와 사용자 인터페이스에 대해서는 사용자가 등록할 때 비밀번호를 지정해야 합니다. 그 당시 비밀번호는 강력한 비밀번호 보안 요건을 준수해야 합니다. 사용자가 기본으로 할당되거나 약하거나 허술한 비밀번호를 쓰게 해서는 안 됩니다.

사용자는 항상 언제라도 비밀번호를 바꿀 수 있어야 합니다. 그러기 위해서는 비밀번호 인증 요건과 통신 보안을 의무화해야 합니다. 가능하다면 2 요소 인증(2FA)을 적용해 비밀번호 변경을 허용하기 전에 사용자의 신원을 검증해야 합니다. 언제나 시스템에 새 비밀번호를 제출할 때에는 사용자가 원 비밀번호를 다시 입력하게 해야 합니다. 이렇게 하면 다른 사용자가 잠기지 않은 노트북에서 열려 있는 웹 애플리케이션을 빼앗았거나 웹 애플리케이션 세션 토큰을 훔쳤더라도 비밀번호 변경은 불가능합니다.

6.11.1 위험

비밀번호 관리를 제대로 의무화하지 않는 시스템은 공격자가 시스템 사용자의 비밀번호를 쉽게 추측할 여지를 주게 됩니다.

6.12 애플리케이션 레이어 인증 및 허가 정의

조직의 신뢰 기반과 그 시스템이 네트워크 통신 레이어를 지켜주는 인증 기술을 정의하긴 하지만, 사용자와 관리, 파트너 승인 기술은 반드시 따로 구성해야 합니다. 이들 엔터티의 통신 채널은 조직의 신뢰 기반으로 보호를 받지만 그 각각의 *행위*와 *신원*은 별도 시스템을 통해 인증을 받아야 합니다.

일반적으로 이와 같은 애플리케이션 수준의 인증은 동일한 서비스의 지원을 받게 됩니다. 그러나 정보는 별도의 리소스에서 얻게 됩니다. 예컨대, 사용자와 관리 인증 데이터는 서로 다른 데이터베이스에 둬야 합니다. 그러면 공격자가 애플리케이션 레이어를 통해(가령 SQL 인젝션을 이용해) 데이터베이스를 조작할 수 있다고 해도 사용자 데이터베이스를 통해 횡으로만 움직일 수 있습니다. 데이터베이스 그 자체를 침해하지 않고는 수직으로 움직이며 관리자 급으로 권한을 격상시킬 수 없습니다. 이것은 조직 보안에서 크나큰 개선입니다.

가능하다면 다음 대상에 대해 별도의 저장 시스템을 지정해야 합니다.

- 엔드포인트 ID
- 사용자
- 관리자 자격증명
- 파트너

그러면 조직의 신뢰기반 서비스에서 관리하는 같은 인증 API 안에서 애플리케이션과 인프라의 임무가 논리적으로 분리됩니다.

본 권장사항과 더불어 다음 기관에서 발간한 자료를 참고하기 바랍니다.

- OAuth 2.0 [8]
- OpenID Foundation [9]
- GSMA Mobile Connect [10]

6.12.1 위험

애플리케이션 레이어 인증과 승인을 의무화하는 방법이 없다면 시스템이 어떤 사용자의 행위가 실제로 그 사용자의 승인을 받은 것인지 확인할 길이 없습니다. 본 권고사항을 구현한다면 각 행위가 인증 받은 사용자와 승인에서 나왔는지 확인할 수 있습니다. 이들 메트릭스는 저장했다가 추후 침해가 의심될 때 검토할 수 있습니다. 이와 같은 단계가 없다면 남용의 위험을 최소화할 안전장치는 없습니다.

6.13 기본 오픈 또는 페일 오픈 방화벽 규칙과 시스템 하드닝

서비스 인프라 환경 중에는 진입 및 진출 보호 메커니즘이 기본으로 구성되지 않은 것도 있습니다. 이 경우 엔지니어링 팀에서 병화벽 또는 네트워크 트래픽 규칙을 자체적으로 마련해야 합니다. 이 규칙은 대중에게 서비스를 제공하기 전에 인프라 안에 확립해야 합니다.

그러나 이런 기술만으로는 서비스 인프라를 충분히 보호하지 못할 때도 있습니다. 방화벽과 기타 네트워크 트래픽 보호장치도 때로는 페일(fail), 장애를 일으킵니다. 이들 시스템이 페일할 때, 열린 상태로 페일(fail open)하는 상황이 종종 발생합니다. 그 이유는 시스템이 페일해도 트래픽은 계속 작동해야 하는 데 있습니다. 다른 컴퓨팅 환경의 트래픽이 IoT 서비스 업체의 트래픽을 따라 해당 인프라를 통과하도록 라우팅될 것이기 때문입니다. 이렇듯 트래픽은 갑자기 멈춰 세울 수가 없습니다. 따라서 시스템은 종종 열린 상태로 페일해 최대한 많은 서비스가 계속 작동하게 하는 것입니다.

엔지니어링 팀에서는 운영체제 하드닝(hardening)을 도입해 인프라 페일로 인해 재앙에 가까운 보안 사고가 일어나지 않게 해야 합니다. 그런데 이것은 기존 서비스 인프라에 연결을 더 많이 할 수 있다는 의미이기도 합니다.

예컨대, 숨어 있는 서비스를 방화벽과 같은 기술 뒤에 배치하면 안 됩니다. 대신 가상가설네트워크(VPN)나 그 외 보안 수준이 높은 보호장치로써 서비스를 공격으로부터 보호할 수 있습니다.

여기서, 소프트웨어 방화벽에는 위험도 따른다는 점을 기억해야 합니다. 노련한 공격자에게 조작을 당할 수 있기 때문이니다. 소프트웨어 방화벽을 이용할 경우, 하드닝이 제대로 안 된 서버 인프라는 공격자에게 조작을 당할 수 있습니다. 다시 말하면, 어떤 서버에서 실행되는 공공 서비스에 불필요한 특권(예컨대 슈퍼 사용자 특권)이 있을 때 서버가 침해를 당하면 공격자가 소프트웨어 방화벽을 무력화할 가능성이 높아진다는 것입니다. 그러므로, 엔지니어링 팀에서는 선택한 아키텍처에 비해 소프트웨어 방화벽의 위험도가 지나치게 높지 않은지 평가해야 합니다.

6.13.1 위험

네트워크 트래픽 보안 시스템의 장애를 보상하는 전략이 없다면 표준 서비스 하드닝 전략으로 쉽게 막을 수도 있었던 공격에 환경이 불필요하게 노출됩니다.

6.14 통신 프라이버시 모델을 평가한다

통신 프라이버시는 (위에서 설명한) 애플리케이션 프라이버시나 통신 정보 보안과는 약간 다른 문제입니다. 프라이버시가 대체로 데이터를 *효과적으로 읽거나 차단*하는 3 자의 능력으로 평가 받는 데 반해 기밀유지와 무결성은 통신 프라이버시 전반을 대변하지는 않기 때문입니다.

그 외에 통신 프라이버시에 영향을 주는 요소는 다음과 같습니다.

- 각 메시지의 크립토그래픽 고유성
- 전송 패턴
- 평문 메타데이터
- 하드웨어 주소 또는 할당 가능한 일련 번호

각 메시지는 보안을 유지하고 검증 가능한 무결성을 지녀야 하지만 크립토그래픽 차원에서도 고유해야 합니다. 사건에 대응하여 공격자가 예측 가능한 어떤 메시지가 발송될 경우, 크립토그래픽 차원에서 고유하지 않은 응답은 공격자가 리플레이할 수도 있습니다. 각 메시지는 공격자가 포착해 이로운 메시지로 리플레이하지 못하도록 고유해야 합니다.

전송 중인 패턴으로 공격자가 특정 사용자를 식별하거나 행동을 일정한 원인적 조치와 동일시할 수 있습니다. 예컨대, 사용자가 어떤 물리적 구역에 들어왔을 때 메시지를 발령하는 기술은 그 메시지를 전송 중에 가로챌 능력을 지닌 "스니퍼"에게 핑거프린트될 수도 있습니다. 이것은, 언뜻 와닿지 않을지도 모르지만, 공격자가 물리적 공간 안에서 누가 어디에 있는지 찾아낼 수 있다면 문제가 될 수도 있습니다. 네트워크 패턴을 평가하여 공격자가 전송 패턴을 간단하게 조치 가능한 데이터로 바꾸는 방법이 있는지 판단해야 합니다.

메타데이터는 정보기관에서 영장을 청구하거나 합법적으로 암호화된 데이터에 접근하지 않고 메시징 시스템의 컨텍스트를 평가하는 용도로 오랫동안 사용됐습니다. 가끔은 메타데이터만으로 조직에서 조치 가능한 정보를 만들기도 했습니다. 그러나 지금은 범죄조직과 일반 사용자까지도 메타데이터를 추적과 기타 부정한 용도에 이용할 수 있습니다. 그 결과, 3 자가 이용 가능한 메타데이터의 양을 줄이는 것이 그 어느 때보다도 중요해졌습니다. 가능하다면 메타데이터의 양을 통신 피어가 메시지의 수신자를 확인하는 데 필요한 정보로 한정해야 합니다.

같은 맥락에서 통신 모듈의 하드웨어 주소와 고유 일련번호는 가급적 보호하거나 무작위로 해야 합니다. 일례로, 애플은 Wi-Fi 접속점을 조사하는 iOS 모델을 변경했습니다. 고정된 하드웨어 주소 대신 무작위 하드웨어 주소를 이용하는 기술을 채용한 것입니다. 이것은 누군가

Wi-Fi 활성 스캔을 토대로 사용자의 위치를 추적할 위험이 줄어드는 효과가 있습니다. IoT 기술도 원리는 비슷하지만, 이 문제의 영향을 받는 통신 기술이 더 많습니다. 일부 기술은 무작위 하드웨어 주소 생성 능력이 없습니다. 셀룰러가 대표적입니다. 그러나 그 외 802.15.4, Wi-Fi, Bluetooth 등은 펌웨어 기능성에 따라 가능할 수도 있습니다.

6.14.1 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.

7 중간 순위 권고사항

중간 순위 권고사항은 엔드포인트 기술의 설계에 따라 관련 여부가 결정되는 것들입니다. 예컨대, 운영체제 수준의 보안 강화는 엔드포인트에서 실행되는 운영체제가 있을 때에 한해 유효합니다. 엔드포인트에 모놀리식 커널 애플리케이션만 있거나 임베디드 실시간 운영체제(RTOS)만 있다면 권고사항은 적용되지 않을 수도 있습니다. 권고사항이 엔드포인트 설계에 적용될 경우 구현해야 합니다.

7.1 애플리케이션 실행 환경 정의

애플리케이션 실행 환경에 대해서는 몇 가지 유의해야 할 점이 있습니다.

- 사용하는 프로그래밍 언어가 보안과 직접 관련이 있을 수도 있습니다.
  - PHP 와 Ruby 같은 언어는 보안 문제를 노출할 수도 있습니다
  - GoLang 과 Erlang 같은 언어는 위험을 낮추기도 합니다
- 3 자 라이브러리는 다음과 같은 위험에 대비해 모니터링하고 관리하고 감사를 해야 합니다.
  - 유지보수 불량
  - 보안 허점에 대한 감사 부재
  - 보안 허점이 발견된 구식 종속관계 요구
- 애플리케이션은 항상 비특권 사용자로 실행해야 합니다.
  - 애플리케이션이 특권 리소스를 요구할 경우, 래퍼(wrapper)를 이용해 해당 리소스를 프로비저닝한 후 특권을 제외하고 전체 애플리케이션을 실행해야 합니다
- 잘 정의된 TCB 와 부트스트랩 모델을 이용합니다.
  - 환경을 잘 정의한 애플리케이션은 *신뢰도*가 높고 더 *안전*합니다

본 권장사항과 더불어 다음 기관에서 발간한 자료를 참고하기 바랍니다.

- OWASP [5]

7.1.1 위험

보안 아키텍처와 함께 배포되는 애플리케이션도 출처의 역추적이 쉽지 않은 침해의 제물이 될 수 있습니다. 서비스와 애플리케이션을 침해하기 위한 도구와 기법은 지난 10 년 사이 발전을

거듭했습니다. 메타스플로잇(Metasploit)과 같은 일부 오픈 소스 기술은 커스텀 익스플로잇을 공격 플랫폼으로 전환해 공격의 은밀성을 높이는 기술을 제공할 수 있습니다.

보안 애플리케이션 실행 환경(Application Execution Environment)이라면 애플리케이션이 실행되고 서로 상대하는 방식과 런타임 중 사용되는 기술의 보안을 강화해 이 같은 위험에 대처할 수 있습니다. 이 같은 속성은 침해의 발생 가능성을 낮출 뿐만 아니라 남용되고 있는 취약성을 추적해 진단하기 위한 추적 능력과 로깅 능력을 보강하는 역할도 합니다.

7.2 파트너 강화 모니터링 서비스 이용

사용 중인 파트너가 모바일 네트워크 운영사라면 모니터링 서비스를 제공할 수 있는 곳인지 파악해야 합니다. 네트워크 운영사 중에는 자사 네트워크를 통해 통신하는 엔드포인트의 거동을 분석할 수 있는 곳도 있습니다. 이와 같은 역량을 지닌 운영사는 이상 거동과 공격적 거동을 나타내는 지표를 평가해본 경험이 있습니다.

이 경우 IoT 업체는 특정 사용자나 엔드포인트가 위협인지, 공격자에게 침해를 당한 적이 있는지 신속하게 가려낼 수 있습니다. 그 결과, 기업에서는 인프라의 다른 영역에서 공격에 선제 대응할 수도 있습니다.

이 서비스가 복잡한 이유는 네트워크 운영사가 제때에 정보를 제공해야 하기 때문입니다. 네트워크 운영사가 IoT 기업에 대한 공격이 있고 난 후에만 정보를 제공할 수 있다면, IoT 기업의 인프라에 마련된 모니터링과 로깅 시스템이 그 거동을 탐지할 수 있어야 합니다. 반면, 네트워크 운영사가 공격적 거동을 네트워크 수준에서 업체에 통보하고 어떤 구독자에게서 이상 네트워크 트래픽이 나오는지 파악할 수 있다면 업체는 해당 사용자의 트래픽을 격리하여 IoT 생태계의 노출을 제한할 수도 있습니다.

7.2.1 위험

IoT 서비스 업체가 모니터링하지 못하지만 의존하게 될 기술이 몇 가지 있습니다. 그 중 하나는 엔드포인트와 서비스 및 네트워크 생태계를 이어주는 통신 네트워크입니다. 모니터링 서비스가 없다면 IoT 서비스 업체가 네트워크 안에서 발생하는 사건을 들여다 볼 방법이 없습니다. 그러므로, 애플리케이션 수준의 아이덴터티 A 가 어떤 서비스를 침해하려고 해도 조직으로서는 엔드포인트 B 가 통신 네트워크에 실제로 연결된 장치인지 확인할 수 없습니다.

이와 같은 정보의 허점은 치명적입니다. 조직이 공격의 탓을 침해 당한 엔드포인트 B 가 아니라 아이덴터티 A 에게 돌릴 수도 있기 때문입니다.

7.3 셀룰러 연결에 사설 APN 이용

액세스 포인트 이름(APN)은 무선 네트워크를 인터넷과 연결하는 셀룰러 통신의 한 요소입니다. 이 포인트는 기본적으로 셀룰러 엔드포인트와 그 엔트포인트가 반드시 상대해야 하는 서비스 인프라 사이에서 가상 사설 네트워크(VPN) 역할을 합니다. 사설 APN(가끔 보안 APN 이라고도 함)은 다음과 같은 몇 가지 제어를 실시하기 위해 보안이 강화된 APN 의 일종입니다.

- 인증 받은 클라이언트에 한정된 액세스
- 방화벽
- 엔드포인트 간 통신 강제 차단
- 이상 탐지를 위한 서비스 모니터링
- 옵션 보안 또는 모니터링 서비스

조직은 APN 에 대한 접속을 제한하여 인증 받은 엔드포인트만 APN 을 통해 제공되는 서비스 인프라에 연결하게 할 수 있습니다. 그러면 불량 또는 무작위 무선 클라이언트가 APN 과 접속 제한 서비스에 연결할 여지가 줄어듭니다. 또한 조직에서 수상한 거동을 찾아내 그 원인이 되는 하드웨어나 사용자를 특정할 수도 있습니다.

방화벽은 클라이언트 측과 서비스 측에서 APN 에 연결된 엔터티가 미승인 채널을 통해 통신하지 못하는 하는 기능을 합니다. 또한 엔드포인트가 APN 을 공개 인터넷으로 가는 채널로 남용하지 못하게 하고 승인 받은 서비스를 향하는 트래픽을 격리하는 역할도 합니다.

엔드포인트 통신을 제한하면 불량 엔드포인트가 APN 을 광역 네트워크로 이용해 다른 엔드포인트를 공격하지 못합니다. 대신 모든 통신이 조직에서 승인한 서비스를 통해 피벗해야 합니다. 조직에서는 원한다면 포인트 간 통신을 완전히 불허할 수도 있습니다.

모니터링 서비스는 보안을 개선하여 조직에서 기존 클라우드 또는 서비스 인프라의 모니터링을 돕는 효과가 있습니다. 조직은 기존의 모니터링 서비스를 네트워크 운영사에게 제공하는 APN/네트워크 모니터링 기술과 결합해 이상 거동의 근원을 더 쉽게 추적할 수 있습니다. 그러면 엔드포인트나 서비스 인프라를 상대로 발생하는 사건을 더 심도 있게 검사할 수 있게 됩니다. 예컨대, 애플리케이션 레이어에서 사용자 A 가 침해를 당했을 수도 있다는

표시가 나오고 사용자 B 의 장비가 APN 에 인증 연결돼 있다면, 조직에서는 APN 모니터링 서비스를 이용해 사용자 B 가 사용자 A 를 침해했을 가능성이 있는지, 아니면 공격자가 사용자 A 와 B 를 모두 침해했는지 밝혀낼 수 있습니다.

네트워크 운영사에게는 위에서 설명한 서비스 위에 구축할 수 있는 서비스가 더 있습니다. 이들 서비스를 통해 네트워크 속 불순 분자를 가려내고 특정 사용자 또는 사용자 집단을 모니터링할 수 있으며 이상 증상을 보이는 트래픽을 다시 라우팅할 수도 있습니다. 그 외 옵션이 더 있을 수도 있습니다. 네트워크 운영사를 통해 어떤 서비스가 조직에게 맞는 것인지 판단해야 합니다.

이들 서비스 전체를 조화롭게 사용하기가 어렵게 보일 수도 있지만 네트워크 운영사의 도움을 받는다면 프로세스가 간소화되고 서비스를 회사의 기존 인프라에 통합하기도 더 간편해집니다. 까다로운 부분은 데이터를 효과적으로 이용하는 것입니다. 이때에는 데이터를 분별 있게 처리하고 관리할 수 있는 엔지니어링 팀이 필요합니다. 서비스에 따라 추가 비용이 들 수도 있으므로, 조직에게 맞는 가격과 서비스를 결정해야 합니다.

7.3.1 위험

프라이빗 APN 이 없다면 엔드포인트 장치가 아무 서비스나 기술과도 연결이 가능합니다. 특히 APN 의 다른 엔드포인트나 인터넷상의 불특정 서비스와 직접 연결도 가능합니다. 이렇게 되면 침해를 당한 엔드포인트가 인터넷상의 거의 모든 서비스를 상대할 수 있고 엔드포인트가 더 안전한 네트워크나 서비스를 공격하기 위한 프록시로 변할 수도 있으므로, 이 권고사항을 의무화하여 엔드포인트가 무작위로 무단 연결을 하지 못하게 해야 합니다. 엔드포인트가 승인받은 서비스에만 연결되게 하면 회사와 전체 IoT 시스템의 보안에 훨씬 더 값어치가 있습니다.

7.4 3 자 데이터 배포 정책을 수립한다

보안 등급을 지정하고 데이터 유형에 등급을 부여하고 위반 정책을 마련했다면 데이터 분산 정책을 수립해야 합니다. 데이터 분산 정책에서는 기술적 통제수단을 통해 정보를 처리하고 접속 허가를 받은 서비스 애플리케이션에 정보를 전달하는 방식을 규정합니다. 이 허가 모델은 데이터 분산 정책의 일부분으로 세분화된 데이터 허가를 만들어내는 사용자의 능력과 짝을 이룹니다.

데이터 분산 정책은 자칫 장황해지기도 하는데, 좋은 정책에는 다음을 비롯해 몇 가지 기본 요소가 있습니다.

- 이 데이터를 통과시키려면 어떤 수준의 상호 인증이 필요한가?
- 데이터에 대해 어떤 기밀유지와 무결성이 필요한가
- 회사가 데이터를 보관하기 위해 갖춰야 하는 역량은 무엇인가?
- 파트너가 데이터를 보관하기 위해 갖춰야 하는 역량은 무엇인가?
- 보관이 허용된다면 데이터의 보관 기간은 어느 정도 될 것인가?
- 데이터에 어떤 수준의 저장 보안을 적용해야 하는가?
- 데이터에 어떤 접속 보안 분류를 적용해야 하는가?

7.4.1 위험

데이터 분산 정책은 내부적으로 IoT 서비스 업체와 동일한 수준의 보안을 적용할 수 없는 파트너에 대해 보안 요건을 규정합니다. IoT 서비스 업체가 파트너 내부 시스템과 네트워크에 도입된 보안을 통제할 수는 없으므로 파트너에게 배포된 데이터가 안전하게 이동하도록 하는 방법 외에는 없습니다. 이것을 정의해 놓지 않으면 데이터가 여전히 IoT 서비스 업체의 통제 하에 있는 상황에서 파트너가 안전하지 않은 구성을 적용해 사용자 데이터를 공격자에게 노출시킬 수도 있습니다. IoT 서비스 업체는 통신에 엄격한 보안 통제장치를 적용하여 데이터가 자사의 통제를 벗어날 때까지 보안 확보를 위해 최선을 다하고 있음을 입증합니다.

7.5 3 자 데이터 필터를 구축한다

파트너에게 광고처럼 역동적으로 생성된 데이터를 받으려면 해당 데이터의 품질과 보안에 관하여 일정 수준의 추정이 필요합니다. 엔지니어링 팀에서는 추정을 하고 데이터를 표현 레이어에 적용할 것이 아니라 서비스 애플리케이션에서 파트너에게 또는 그 반대로 배포되는 데이터가 잘 형성돼 악성 콘텐츠를 담고 있지 않도록 대책을 강구해야 합니다.

이를 위해 엔지니어링 팀은 다음 모델을 고려해야 합니다.

- 파트너가 데이터 모델에 대해 구상한 포맷과 데이터가 잘 맞는가?
- 데이터의 형식이 좋은가?
- 데이터가 클라이언트의 오해를 유발할 수도 있는 다형태 객체를 나타내는가?
- 클라이언트가 표현 레이어를 렌더링하는 방식에 데이터가 영향을 미칠 것인가?
- 클라이언트가 표현 레이어를 *해석*하는 방식에 데이터가 영향을 미칠 것인가?
- 사용자가 보안을 저해하는 행동을 하도록 데이터가 유도하거나 요청하는가?

- 데이터가 클라이언트 GUI 의 컴포넌트(비밀번호 입력 필드)를 스푸핑하거나 사칭하는가?

승인 받은 모델에 부합하지 않는 데이터는 거절해야 합니다. 그런 데이터가 탐지되면 즉시 관리팀에게 보고하고 데이터의 출처와 형식에 관한 메트릭스를 최대한 많이 넣습니다. 가능하다면 보안 데이터베이스에 샘플을 기록합니다.

7.5.1 위험

3 자에게서 받은 역동적으로 생성된 데이터에는 의도했든 의도하지 않았든 맬웨어나 부적절한 콘텐츠, 기타 바람직하지 않은 데이터가 들어 있을 수도 있습니다. 3 자 서비스의 정의에 맞춰 마련된 진입 필터가 없다면 조직은 우발적으로 맬웨어나 기타 악성 콘텐츠를 최종 사용자에게 전달할 위험에 놓이게 됩니다. 그러면 그런 데이터로 인해 시스템을 침해 당하거나 고객을 잃어버릴 수도 있습니다.

8 낮은 순위 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.

8.1 로우해머 및 유사 공격

DRAM(Dynamic Random Access Memory)이나 SRAM(Static Random Access Memory) 같은 최신 RAM 기술은 일부 메모리 액세스 시퀀스로 유도할 수 있는 에러에 취약합니다. 이런 유형의 에러를 남용하면 예측 가능한 메모리 영역에서 특정 비트를 변경할 수 있습니다. 이 조건을 잘 익스플로잇하면 메모리에서 소프트웨어가 지정하는 특권을 나타내는 비트를 변경할 수 있습니다.

다시 말하면, 공격자가 제대로 익스플로잇하면 DRAM 이나 SRAM 의 최신 구현체에서 하드웨어 결함을 조작해 본인의 특권을 한 사용자에서 다른 사용자로 격상시킬 수 있다는 것입니다. 이 같은 취약성을 통해 DRAM 과 SRAM 의 최신 구현체를 익스플로잇할 수 있음이 다수 입증되었습니다. 그러나, 이것이 가능하려면 로컬 시스템에서 코드를 실행해 이 버그를 촉발할 수 있는 메모리 액세스 시퀀스를 만들어야 합니다.

샌드박스 GoLang, Python, Erlang 와 같은 런타임 언어를 통해 원격으로 이런 거동을 촉발할 수 있을지도 모르지만, 그런 형태의 공격의 정밀도에 대해서는 아직 발표된 문헌이 없으며 익스플로잇으로서 효과를 발휘할 가능성도 상당히 희박합니다.

이런 공격은 하드웨어 차원에서 해결해야 합니다. 그러나 엔지니어링 팀이 클라이언트가 특정 서비스에서 코드를 실행하지 못하게 하여 남용의 위험을 낮추는 방법도 있습니다. 가상 머신이나 런타임을 통해서도 가능합니다. 엔지니어링 팀에서는 이렇게 제한을 가하여 공격자가 이 공격에 필요한 메모리 액세스 시퀀스를 하지 못하게 할 수 있습니다.

8.1.1 위험

이런 유형의 공격에 대비해 방어태세를 충분히 갖추지 않으면 공격자가 원격으로 특권을 승격하거나 목표 호스트를 대상으로 임의의 코드를 실행할 수도 있습니다. 그러나 성공하려면

하드웨어와 운영체제, 공격 벡터 등에 상당히 해박한 지식을 갖춰야 합니다. 그러므로 이런 류의 공격은 가능성도 낮고 드뭅니다.

8.2 가상 머신 침해

최신 서비스 인프라는 가상 머신을 이용해 주문형 서비스를 제공하기도 합니다. 이 모델은 매우 편리하고 구축하기도 쉽다고 입증되었지만 인프라 전체의 보안에 문제가 있습니다. 엔지니어링 팀에서 이것저것 잘 생각해 아키텍처를 만들어 놓아도 가상 인프라를 관리하고 배포하는 조직에서 기대에 부응하지 못할 수도 있습니다.

가상 서버 환경에서 배포할 때 커다란 걱정거리 한 가지는 호스트가 침해를 당할 위험, 또는 서버(가상 게스트)가 같은 인프라에서 실행 중인 다른 게스트의 데이터를 가로챌 가능성입니다.

그와 같은 공격은 충분히 일리가 있고 IoT 서비스 업체에서 평가해야 하는 문제지만 공격을 완벽하게 실행하려면 상당한 기술과 시간이 필요합니다. 따라서 공격이 일어날 수는 있지만 실제 그 가능성은 희박하다고 하겠습니다. 그러나 서비스 인프라에 보호장치가 제대로 갖춰져 있지 않다면 공격자가 가상 머신에 대한 관리자의 접근을 침해할 수 있을지도 모릅니다. 이 같은 침해는 특별히 고급 기술 없이도 가능합니다.

이 문제에 대처하는 한 가지 방법은 서버 프로비저닝입니다. 이를 통해 각 서버는 일단의 고유한 크립토그래픽 키로 인코딩됩니다. 이 방식을 따른다면 어느 한 서버가 침해를 당해도 다른 서버는 안전합니다.

8.2.1 위험

이런 유형의 공격에 대비를 하지 않는다면 서비스 인프라가 여러 유형의 공격에 취약해질 수도 있습니다. 서비스 인프라와 데이터 침투, 프라이버스 침해, 사용자 사칭을 통해 접근 가능한 키로 서버 사칭이 가능지도 모를 일입니다.

8.3 사용자를 위한 API 를 구축해 프라이버시 속성을 관리한다

사용자는 누구나 서비스 API 를 통해 3 자에게 어떤 정보를 제공할지 통제할 수 있어야 합니다. 정보는 데이터의 종류 별로 분류하고 보안 등급을 부여해야 합니다. 사용자는 본인 계정의 모델링에 사용되는 데이터의 유형과 등급을 불러서 확인할 수 있어야 합니다. 또한 데이터의 유형에 조건을 부과하여 파트너에게 데이터 접근권을 주거나 회수할 수 있어야 합니다.

그 방식은 인증 받은 API 이거나 일반 또는 파트너 별로 단순히 예 또는 아니오 통제를 할 수 있는 GUI 가 될 수 있습니다.

8.3.1 위험

사용자가 IoT 서비스 업체에 제공할 데이터를 통제하지 못한다면, 서비스 업체 또는 서비스 업체가 이용하는 파트너에서 보안 침해가 발생했을 때 사용자는 데이터가 노출될 위험에 처하게 됩니다. 사용자마다 위험도의 차이가 있기 때문에 각 사용가 본인의 니즈에 따라 프라이버시 제약사항을 미세조정할 수 있어야 합니다. 이 인터페이스를 제공하면 통제가 가능하게 됩니다. 사용자는 본인의 니즈에 맞춰 통제수단을 스스로 조정할 수 있어야 합니다. 예컨대, oneM2M 는 사용자가 (TS-0003 를 통해) 서비스 업체의 프라이버시 환경을 설정할 수 있습니다.

8.4 오탐/미탐 평가 모델 정의

오탐 분석은 지극히 복잡한 주제이지만 어떤 기술이 오탐을 보일 가능성이 높은지 간단하게 찾아내는 방법이 있습니다. 다음 각 항목을 평가하면 됩니다.

- 데이터 소스가 *믿을 만한가*?
- 데이터가 훼손되거나 스푸핑 당할 수도 있는가?
- 데이터 소스가 아날로그 도메인에서 오는 것인가?
- 데이터를 여러 소스에서 제공 받을 수 있는가?
- 보강 데이터 소스가 같은 엔드포인트 시스템에 존재하는가?
- 보강 데이터 소스가 훼손이나 스푸핑하기 쉬운가?
- 도구는 데이터 소스를 조작할 때 쉽게 이용할 수 있는가?
- 데이터 소스를 조작할 때 전문성이나 비용은 어느 정도 필요한가?
- 데이터 소스에 연결된 장치는 *믿을 만한가*?

위와 같은 속성 모두 데이터의 *신뢰도* 평가에 사용할 수 있는가? 이것은 지극히 중요한 문제입니다. 현실에 영향을 미치는 중요한 결정이 잠재적으로 유해한 효과를 낳을 수도 있기 때문입니다. 엔지니어링 팀에서 신뢰도 모델을 만들어 중요한 의사결정에 연루되는 데이터 소스마다 이를 적용하는 것이 중요합니다. 데이터 소스의 무게가 신뢰하기 어려운 정도라면 가장 합리적이고 안전한 대책을 강구해야 합니다.

여기서, 이런 결정을 내리는 주체는 엔니지어링 팀만이 *아니*라는 점을 기억해야 합니다. 회사 수뇌부, 법무팀, 보험팀도 잠재적으로 위험이 있는 시나리오에서는 올바른 대응방안의 결정에

관여해야 합니다. 그러면 엔지니어링 팀에서 올바른 의사결정 프로세스를 *검증과 재현이 가능한 방식*으로 기술에 인코딩해야 합니다.

이 프로세스는 매우 까다롭습니다. 중요한 시나리오에서 기술이 대응하는 방식에 관하여 조직 전체의 주의를 요하기 때문입니다. 신뢰도는 어느 한 기술에 적용하기 어려운 속성입니다. 임베디드 기술은 더더욱 그렇습니다.

8.4.1 위험

엔지니어링 팀에게 오탐을 평가하는 모델이 없다면 더 중요한 사건이 일어나는 동안 사소한 문제를 분석하느라 너무 많은 시간을 쓸 수도 있습니다. 그러면 조직에서 분석한 메트릭스가 생산 현장에서 일어나고 있는 사건의 유형에 대해 명확한 방향성을 제시하지 못할 위험이 커질 수도 있습니다. 그렇게 되면 로깅과 모니터링 인프라의 가치가 저하되고 그 비싼 리소스를 효과적으로 사용하는 조직의 능력도 타격을 받게 됩니다.

9 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.

부록 A 문서 관리

문서 이력

버전	날짜	변경 내용	승인권자	편집자 / 회사
1.0	2016-02-08	New PRD CLP.12	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	oneM2M 참고문헌 추가	IoT Security Group	Rob Childs GSMA

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.