https://doi.org/10.5392/JKCA.2020.20.02.015

현대의 보안부팅 개발 방식 분석: 플랫폼 환경을 중심으로

Analysis on Development Methodology of Modern Secure boot: Focusing on Platform Environment

김진우[*], 이상길[*], 이정국[*], 이상한[**], 신동우[**], 이철훈[*]
충남대학교 컴퓨터공학과[*], ETRI부설연구소[**]

**Jin-Woo Kim(jinu@cnu.ac.kr)[*], Sang-Gil Lee(sk0137@cnu.ac.kr)[*],
Jeong-Guk Lee(jeongguk.lee.k@o.cnu.ac.kr)[*], Sang-Han Lee(freewill71@nsr.re.kr)[**],
Dong-Woo Shin(dwshin@nsr.re.kr)[**], Cheol-Hoon Lee(clee@cnu.ac.kr)[*]**

요약

보안부팅은 부팅 단계에서 컴퓨터 시스템의 무결성에 대한 검증을 수행하고 그 결과에 따라 부팅 과정을 제어하는 보안 기술이다. 컴퓨터 시스템은 보안부팅을 통해 커널과 커널의 권한을 노리는 각종 악성코드의 위협으로부터 안전한 실행 환경을 구축할 수 있으며, 유사시 시스템의 복구를 지원하기도 한다. 보안부팅은 최근 해커의 공격으로부터 사용자의 정보를 보호하고, 악성 사용자로 인한 자사 제품의 악용을 방지하기 위해 현대의 다양한 컴퓨터 제조사에서 채택하기 시작하였다. 본 논문에서는 다양한 기업과 단체에서 개발하고 있는 보안부팅을 플랫폼별로 분류하여 알아보고, 각 보안부팅의 설계구조와 개발목적에 대한 분석과 설계상의 한계에 대해 조사를 수행하였다. 이는 시스템 보안 설계자에게 보안부팅 개발 방식의 다양한 정보와 시스템의 보안 설계에 참고자료로 활용될 수 있다.

■ 중심어 : | 시스템 보안 | 플랫폼 보안 | 보안부팅 | 임베디드 시스템 | 보안 요구사항 |

Abstract

Secure boot is security technology that verifies the integrity of the computer system in boot stage and controls the boot process accordingly. The computer system can establish a secure execution environment from the threat of various malwares by security boot and also supports the recovery when system in emergency case. Recently, Secure boot has been adopted by various modern computer manufacturers to protect users' information from hacker attacks and to prevent abuse of their products by malicious users. In this paper, we classify security boot developed by various companies and organizations by platform, and analyze the design and development purpose of each security boot and investigate the limitation of design. It can be used as a reference for system security designers in various information of security boot development method and security design of system.

■ keyword : | System Security | Platform Security | Secure Boot | Embedded System | Security Requirements |

* 본 연구는 2019년 ETRI부설연구소의 "임베디드 시스템을 위한 시큐어부팅 기술 안전성 강화방안 연구(2019-129)" 과제로 수행된 연구결과입니다.

접수일자 : 2019년 11월 05일
수정일자 : 2019년 11월 19일
심사완료일 : 2019년 11월 19일
교신저자 : 이철훈, e-mail : clee@cnu.ac.kr

I. 서 론

부트킷(BootKit)과 루트킷(RootKit)은 컴퓨터 시스템의 부팅 과정과 커널을 조작하여 사용자에게 피해를 주는 악성코드이다. 이들은 PC가 보급되기 시작하던 1980년대의 첫 부트킷 브레인 바이러스부터 현대의 각종 바이러스에 이르기까지 지속적으로 발전하면서 사용자를 위협하고 있다[1]. 이들은 자신이 감염된 사용자의 시스템에 지속적인 피해를 줄 수 있도록 자신의 존재를 조작, 은닉하여 사용자와 커널의 AV(Anti-Virus software, 안티바이러스 소프트웨어)가 이를 쉬이 탐지하지 못하도록 설계되었다. 이처럼 커널 상의 일반적인 AV로는 탐지하기 어려운 부트킷과 루트킷을 부팅 단계에서 검출하기 위해 보안부팅이 설계되었다.

보안부팅은 하드웨어 설계를 기반으로 시스템 부팅 중 인증되지 않은 부트 코드에 대한 실행을 제어함으로써 부트킷과 루트킷으로부터 시스템의 무결성을 보호하는 역할을 수행한다. 보안부팅은 현대의 PC 환경과 개인의 스마트폰, 자율주행차량과 금융환경과 같이 다양한 환경에서 채택하고 있는 보안 구조이며, 보안과 밀접하게 관련된 생체인식 기능과 금융 관련 애플리케이션을 비롯해 사용자 시스템의 실행 환경에 신뢰성을 보장하는 TEE(Trusted Execution Environment, 신뢰 실행 환경)를 제공하기 위해선 보안부팅이 필수로 요구된다[2].

보안부팅을 개발하는 주요 기업들은 대상의 시스템 환경과 사용 목적에 따라 조금씩 다른 설계를 하고 있다. 또한, 각 기업은 보안부팅 방식의 강화를 비롯한 시스템의 복구와 원격증명과 같이 다양한 방식으로 보안성을 강화하는 한편, 자사 제품에 최적화된 보안 기능을 제공하여 시스템의 성능 향상과 보안 수준에 대한 증명을 통해 해당 기업에 대한 신뢰도를 상승시키는 중요 마케팅 요소로 활용되기도 한다[3]. 하지만 보안부팅은 개발 주체의 목적과 이념에 따라 설계 방식이 서로 다르고, 설계에 따른 장·단점을 비교하기 위한 참고자료가 부족하여 보안부팅을 적용하거나 개발하고자 하는 경우 해당 정보에 대한 조사에 많은 시간을 할애해야 한다.

이에 본 논문에서는 다양한 환경에서 각기 다르게 적용된 현세대 보안부팅의 분석과 목적에 대한 세부적인 조사를 수행하고, 이를 플랫폼 환경별로 분류하여 보안부팅 개발의 참고자료로써 정보를 제공하고자 한다. 서론에 이어 2장 관련 연구에서는 부트킷과 루트킷의 공격 방식에 대한 설명을 비롯하여 보안부팅에 대한 기본적인 개념 설명을 할 것이며, 3장 보안부팅 조사 및 분석에서는 조사 대상으로 PC 환경의 펌웨어인 UEFI와 운영체제 Windows와 Linux, macOS의 보안부팅을, 스마트폰 환경에서는 스마트폰 환경의 대표 운영체제인 Android, iOS와 스마트폰 환경의 보안 솔루션인 Knox에 대한 분석을 통해 현대의 다양한 방법으로 개발된 보안부팅 설계에 대해 설명을 할 것이다. 4장 각 보안부팅의 비교에서는 3장의 조사 결과를 바탕으로 각 보안부팅에 대한 비교를 다루고, 5장 결론에서 본 논문에 대한 정리와 미래의 보안부팅 발전 방향에 대한 고찰로 마무리한다.

II. 연구 배경

1. 부트킷과 루트킷의 위협

현대의 부트킷과 루트킷은 시스템 사용자에 대한 무차별적인 공격이 아닌 백도어, 랜섬웨어, DDoS(Distributed Denial of Service, 분산 서비스 거부)를 위한 봇넷(Botnet) 등 공격자가 의도하는 다양한 동작을 수행하도록 설계되고 있다[4]. 이들은 일반적으로 악성코드가 담긴 스팸메일[5], 네트워크 프로토콜의 취약점, 플래시 플레이어의 취약점과 SNS를 통한 사회공학적 공격 등, 다양한 방법을 통해 감염 대상의 시스템에 침투한다[6]. 컴퓨터 시스템의 증가와 네트워크 방식의 발전, SNS의 발달에 힘입어 부트킷과 루트킷의 감염으로 인한 피해 규모는 개인에서 사회에 이르기까지 널리 분포되어있다.

1.1 부트킷

부트킷은 부트 섹터인 BIOS의 MBR(Mater Boot Record)과 같이 주요 부팅 컴포넌트에 대한 공격을 수행하는 악성코드를 일컫는다[7]. 이들은 펌웨어 단계 직후의 부트 단계를 조작하여 올바르지 않은 부트로더를

실행하거나 실행하지 못하게 하는 등의 동작을 수행하며, 일반적으로 부트 섹터가 포함된 하드디스크 드라이버의 취약점을 노리는 방식으로 시스템을 감염시킨다.

최초의 부트킷으로 불리는 브레인 바이러스는 1987년 미 델라웨어 대학교에서 처음 발견되었다[1]. 브레인 바이러스는 부트 섹터를 훼손시킨 다음, 저장장치에 불량 섹터를 발생시키고 해당 영역에 바이러스 코드를 저장시킴으로써 자신의 존재를 숨길 수 있었다. 당시 브레인 바이러스는 시스템의 성능 저하를 야기한 대신 사용자의 데이터에 대한 훼손은 주지 않았기에 실질적인 피해는 크지 않았지만, 이후 부트킷 공격 방식이 발전함에 따라 MS-DOS의 부트 섹터를 감염시킨 BSIs(Boot Sector Infector)를 거쳐 최근에는 랜섬웨어 Petya까지, 시스템에 대한 무차별적 파괴를 수행하는 부트킷들이 등장하면서 부트킷으로 인한 피해 규모는 점차 늘어나게 되었다.

1.2 루트킷

루트킷은 커널 단계에서 동작하는 악성코드이다. 부트킷이 시스템의 전반적인 장애 발생이 목적이었다면, 루트킷은 커널의 권한에 대한 탈취를 통해 공격자가 심어놓은 프로그램을 AV의 탐지로부터 숨겨 시스템에 대한 지속적인 공격을 수행하는 것이 목적이다[8].

공격자는 스팸메일, 플래시 플레이어 취약점, 트로이목마 등 다양한 방법으로 루트킷을 사용자에게 침투시킨다. 이후 루트킷은 시스템의 정상적인 프로그램에 기생하거나, 서드파티 드라이버의 모습으로 시작 프로그램에 포함되기도 하는 등 사용자가 인지하기 어렵게 위장하여 자신을 은닉시키고, 사용자의 개인, 금융, 행동 패턴 등의 정보를 지속적으로 유출해 사용자를 위협에 빠뜨린다.

Windows, Linux와 같은 범용 운영체제에서는 루트 권한을 가진 사용자의 잘못된 실행만으로도 루트킷의 동작이 가능하므로, 운영체제 개발사에서는 이와 같은 공격으로부터 시스템을 보호하기 위해 다양한 방법으로 커널 환경의 무결성을 지키고자 하고 있다.

2. 보안부팅의 대두

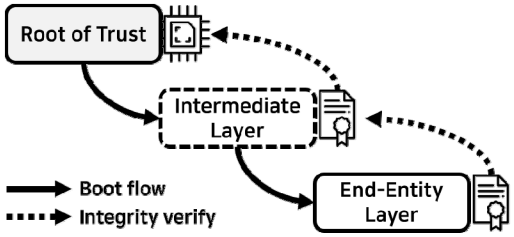

그림 1. 보안부팅의 동작 과정

보안부팅은 기존의 부트 방식인 BIOS MBR 구조의 취약함, 검증되지 않은 운영체제, 신뢰할 수 없는 실행 환경과 프로그램과 같이 다양한 보안 위협으로부터 사용자를 보호하기 위해 고안되기 시작하였다.

보안부팅의 기본적인 개념은 하드웨어적으로 보호되고 제조사에 의해 검증된 무결성 정보가 저장된 RoT(Root of Trust, 신뢰의 뿌리)를 기반으로 한다[9]. 보안부팅의 동작 과정은 [그림 1]과 같이 최하위에 존재하는 RoT가 다음 단계의 부트 컴포넌트에 대한 검증 절차를 수행하고, 무결성이 검증된 경우 이를 실행한다. 보안부팅은 이 과정을 반복하여 이전 단계의 부트 컴포넌트가 다음 단계의 부트 컴포넌트를 검증하는 신뢰 체인의 구조로 사용자에게 신뢰 가능한 환경을 제공한다. 보안부팅을 개발하는 여러 제조사와 개발사는 자사의 시스템에 최적화된 보안부팅을 제공할 수 있도록 서로 조금씩 다른 구조로 되어 있으며, 이는 3장에서 자세히 다루도록 하겠다.

Ⅲ. 플랫폼별 보안부팅 조사 및 분석

1. PC 환경의 보안부팅

1.1 UEFI secure boot

2019년 현재 개발되고 있는 대다수의 PC 운영체제는 UEFI(Unified Extensible Firmware Interface, 통일 확장 펌웨어 인터페이스)를 기반으로 개발되고 있다. 구식 BIOS 방식의 PC 펌웨어에서 더욱 진화한 환경을 제공하는 UEFI는 프로세서 제조 회사인 Intel에 의해 설계되어 과거에서는 지원하지 못했던 다양한 기능을 제공하고 있다. 특히나 UEFI specification

2.3.1 Errata c 버전에서부터는 마이크로소프트가 주도적으로 개발한 보안부팅 기능을 제공하기 시작하였으며, 이를 통해 RSA 인증서 기반의 운영체제 무결성 검증 절차를 도입할 수 있게 되었다[10].

UEFI 보안부팅은 운영체제의 부트로더에 대한 개발사의 RSA 인증서에 대한 서명 검증 절차를 통해 이루어지는데, RSA 인증 절차에 사용되는 공개키는 UEFI 규격을 사용하는 메인보드의 NVRAM(Non-Volatile RAM, 비휘발성 RAM) 영역에 저장된 공개키 데이터베이스를 기반으로 제공된다.

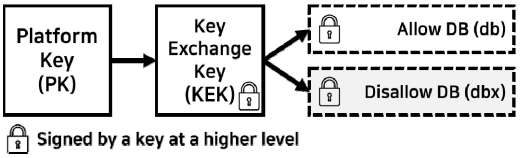

그림 2. UEFI의 키 데이터베이스 구조

공개키 데이터베이스는 [그림 2]와 같이 허용된 공개키와 만료된 공개키의 데이터베이스를 따로 관리한다. 만일 실행하고자 하는 부트로더의 인증서에 사용되는 공개키가 과거 취약점이 드러나 더는 사용되지 않는 만료된 공개키의 데이터베이스에 존재하는 경우 이에 대한 실행을 방지할 수 있다. UEFI 보안부팅의 통일된 키 데이터베이스 관리 방식과 부팅 과정은 PC 환경의 서로 다른 OS 간 멀티 부팅에도 적용되기 용이한 구조로서 그 호환성 또한 뛰어난 것으로 증명되었다.

만일 공식적으로 UEFI 보안부팅을 지원하는 부트로더를 개발하기 위해서는 UEFI 보안부팅을 개발한 마이크로소프트의 보안 인증 절차를 거쳐야 한다[11]. 보안 인증 절차에는 부트로더 설계의 보안 수준 검증과 호환성, 코드 저작권과 같이 중요 요소에 대한 검증을 수행하며, 절차를 통과하는 경우 부트로더에 대한 인증서가 발급된다. 해당 보안 인증 절차를 신청하기 위해서는 마이크로소프트사에 인증 절차 비용으로 99달러를 지급해야 하기에, 자유로운 소프트웨어 개발을 추구하는 리눅스 진영에서는 해당 정책에 대한 일부 반발이 일기도 했다[12].

UEFI 보안부팅은 부트로더의 RSA 인증서에 대한 서명 검증이라는 단순명료한 방식을 채택함으로 설계의 호환성과 범용성을 얻을 수 있었지만, 설계의 한계로 인한 취약점이 존재한다. 우선, 하드웨어 설계상 물리적 탈취에 매우 취약하다. UEFI 규격을 사용하는 메인보드는 플래시 ROM을 통해 펌웨어에 대한 정보를 저장하고, NVRAM을 통해 공개키 데이터베이스와 각종 설정을 저장한다. 플래시 ROM의 경우 암호화가 이루어지지 않기에 영역에 저장된 값을 읽거나 ROM-Writer를 통한 수정이 가능하며, NVRAM의 경우 NVRAM에 저장된 값을 유지하기 위해 장착된 배터리를 제거하는 경우 내부에 저장된 값이 전부 초기화되기도 한다[13]. 이렇게 물리적 탈취로부터 취약한 환경에서 시스템 장치에 대한 별도의 보호를 하지 않는다면, 공격자는 손쉽게 시스템을 훼손시킬 수 있을 것이다. 다음으로는 CSM(Compatibility Support Module, 호환성 지원 모듈)의 존재이다[14]. UEFI에서 동작하지 못하고 구형 BIOS에서만 동작하는 응용프로그램을 위해 제공되는 해당 모듈은 UEFI 환경에서도 BIOS 구조와 같은 환경을 제공하기 위해 설계되었지만, 그로 인해 보안부팅이 지원되지 않던 과거 BIOS 환경의 취약점이 UEFI 환경에서도 적용될 수 있다는 가능성이 존재한다. 결국, CSM으로 인해 발생하는 취약점 우려를 해소하기 위해 인텔에서는 2020년 이후 제공되는 UEFI 환경에서는 CSM을 제거하기로 결정했다.

1.2 Microsoft Windows

UEFI 보안부팅을 주도적으로 개발한 마이크로소프트는 자사의 운영체제인 Windows 8부터 적극적으로 보안부팅을 지원하고 있다. 우선 Windows에서는 UEFI 보안부팅을 통해 검증된 Windows 부트로더를 기반으로 부트 컴포넌트에 대한 연쇄적인 RSA 인증서 검증을 수행한다. 마이크로소프트에서는 UEFI 보안부팅 이후의 해당 단계를 신뢰 부팅(Trusted boot) 과정이라 정의하였으며, 여기서 자사의 퍼스트파티 드라이버 외의 서드파티 드라이버에 대해 효율적인 인증 절차를 제공하기 위해 ELAM(Early Launch Anti-Malware, 멀웨어 차단 조기 실행) 드라이버를 설계, 도입하였다[15].

ELAM은 부팅 단계에서 Windows는 자사의 인증서로 인증된 퍼스트파티 드라이버 외, 서드파티 드라이버

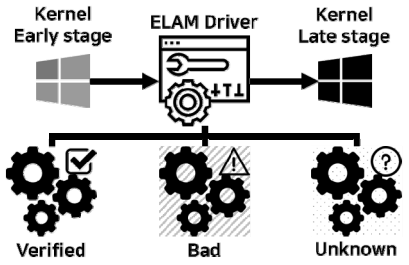

그림 3. Microsoft Windows의 ELAM 구조

를 실행하기 전 해당 드라이버에 대한 보안성을 검증하는 단계이다. ELAM은 [그림 3]과 같이 드라이버에 대한 정보를 인증된, 금지된, 알 수 없는 세 가지의 상태로 분류하여 저장하고 있다. ELAM 저장된 정보를 기반으로 Windows의 보안 설정에 따라 부팅 과정에서 실행하고자 하는 드라이버가 만일 보안이 검증되지 않았거나, 과거 취약점이 발견되어 만료된 드라이버의 경우 이에 대한 실행 여부를 제어할 수 있는 기능이 있다. 또한, 여기서 사용되는 ELAM의 보안 규칙을 기준으로 악성코드로부터 안전한 드라이버를 개발하기 위한 지침을 제공하기도 한다[16].

Windows 환경에서 ELAM에 의해 검증받은 드라이버를 개발하기 위해서는 몇 가지 조건이 요구된다. 첫 번째로는 드라이버 개발자 또는 개발사가 마이크로소프트의 악성코드 방지 그룹인 MVI(Microsoft Virus Initiative)에 가입돼 있어야 하고, MVI에서 제공하는 악성코드 방지 개발 환경을 사용하여야 한다[17]. 두 번째로는 개발한 드라이버가 WHQL(Windows Hardware Quality Lab) 테스트를 통해 Windows 환경에서의 호환성과 안정성을 인정받아야 한다는 것이다.

이처럼 Windows는 ELAM을 통해 관리자 권한을 노린 루트킷의 악성 서드파티 드라이버 공격을 효율적으로 방어할 수 있도록 설계하였으며, 이외에도 Windows 전용 AV인 Windows Defender와 TPM(Trusted Platform Module, 신뢰 플랫폼 모듈)지원 환경을 제공함으로써 사용자에게 추가적인 보안 환경을 제공하고 있다.

1.3 Apple macOS

애플의 macOS는 자사의 PC 모델인 iMac, MacBook 제품에 탑재되는 것을 고려하여 설계된 OS이다. 기본적으로 macOS는 UEFI 보안부팅을 지원하였는데, 애플은 UEFI 부팅 과정의 신뢰성에 의문을 제기하였다. 물리적 탈취에 대한 위협을 비롯하여 부팅 과정이 전부 메인 프로세서 영역에서 동작하는 UEFI 보안부팅은 역공학으로부터 안전하지 않다는 것이었다. 실제로 메인 프로세서에 대한 역공학이 성공하는 경우 부팅 과정의 데이터 누출을 비롯하여 시스템의 무결성에 대한 위협 가능성이 존재했기 때문에, 2018년, 애플은 이와 같은 우려를 해소하기 위해 자사 macOS 전용의 보안 보조프로세서인 T2 칩을 개발하여 자사의 제품에 탑재하기 시작하였다[18].

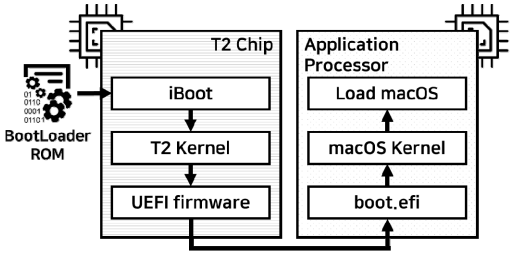

그림 4. Apple T2 Chip의 보안부팅 흐름

애플의 프로세서 분류에서 보안프로세서 제품군인 T-Series로 구분되는 T2 칩은 내부에 전용 메모리와 보안 목적을 가진 보조프로세서를 포함한 SoC(System on Chip, 단일 칩 시스템)로 설계되었으며, 로직 보드의 형태로 macOS 제품에 탑재된다. 탑재된 T2 칩은 메인프로세서를 대신하여 macOS 제품의 부팅을 수행하며, [그림 4]와 같이 하드웨어에 대한 초기화를 수행하는 애플의 독자 부트로더 iBoot를 비롯한 UEFI의 부팅 연산을 처리한다. T2 칩은 기존의 UEFI 환경과 다르게 부팅 연산을 SoC 내부에 탑재된 ARM 프로세서의 보안영역에서 처리하기 때문에 부팅 관련 정보를 밖으로 노출하지 않으며, 연산에 요구되는 데이터 또한 AES 암호화하여 처리함으로써 부팅 간 시스템에 대한 무결성과 기밀성을 보호할 수 있도록 설계되었다. 마지막으로 T2 칩에서 UEFI 보안부팅까지 연산이 완료된 시점에서 T2 칩은 부팅에 관련된 권한을 메인프로세서로 넘겨주게 되며, 남은 부팅 과정은 메인프로세서에서

수행할 수 있도록 설계되었다.

T2 칩은 단순 보안부팅을 위한 보안 실행 환경 이외에도 추가적인 보안 기능과 시스템의 성능 향상을 지원하고 있다. 우선 보안을 위한 생체인식정보의 저장영역이다. T2 칩은 애플의 지문인식 기술인 TouchID와 안면인식 기술인 FaceID, 음성 지원 인공지능 어시스턴트 Siri에서 사용하는 생체정보를 외부로부터 격리된 T2 칩 내부의 저장소에 저장함으로써 이를 안전하게 보호한다. 다음으로는 시스템의 킬-스위치(Kill-Switch) 기능이다. T2 칩은 시스템을 구성하는 장치의 모든 무결성 정보를 가지고 있으며, 만일 애플로부터 인가되지 않은 장치가 연결된 경우 시스템을 강제로 잠글(Lock) 수 있는 기능이 존재한다[19]. 마지막으로 T2 칩과 메인프로세서와의 병렬 연산을 통한 시스템의 성능 향상이다. 과거에는 메인프로세서가 수행하던 macOS의 저장소 암호화 기능인 FileVault 2의 암호화 연산과 각종 고화질 이미지와 영상에 대한 연산을 T2 칩 내부에 탑재된 보조프로세서와 ISP(Image Signal Processor, 이미지 신호 처리기)를 통해 병렬로 처리할 수 있도록 지원하여 macOS 시스템의 메인 프로세서에 가해지는 연산 부담을 상당수 해소하게 되었다[20].

1.4 Linux

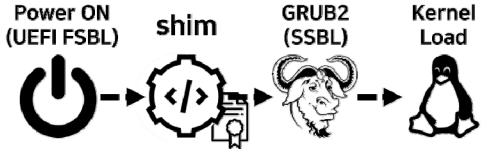

그림 5. shim 구조가 적용된 Linux의 부트 과정

UEFI 보안부팅을 지원하기 위해서는 마이크로소프트 주관의 인증서를 발급받아야 하고, 인증서 발급에 비용을 지급해야 한다는 점은 리눅스 진영에서 그다지 반기는 일은 아니다. 더욱이 GNU가 개발하여 UEFI 환경의 리눅스 부팅에 사용되는 멀티 부트로더인 GRUB2는 GNU의 GPL(General Public License, 일반 공중 사용 허가서)을 따르기 때문에 소프트웨어 개발에 비용이 발생하는 인증서를 발급받는 것은 사상에 어긋나는 정책이었고, GRUB2는 마이크로소프트의 인증 정책 대상이 될 수 없었다[21]. 결국, 이와 같은 리눅스 환경에서 UEFI 보안부팅을 위해서 UEFI와 GRUB2를 이어줄 연결 장치가 요구되었다.

이러한 과정에서 주요 리눅스 배포기업에서는 심(shim)을 채택하게 되었다[22]. 심은 오픈소스로 개발되어 [그림 5]에서와 같이 FSBL(First Stage BootLoader)에 해당하는 UEFI 펌웨어 부팅 단계 직후에서 동작하는 단순 API 코드 집합이다. 기존에는 간단한 프로그램의 실행 또는 다른 부트로더를 실행하는 역할을 담당하고 있었지만, 주요 리눅스 배포기업에서는 이러한 심을 UEFI와 GRUB2 간의 보안부팅을 지원해주기 위한 가교역할을 수행하도록 설계를 하게 되었다. GRUB2는 보안부팅의 인증 대상이 될 수 없으니 대신 자신들이 개발한 심에 대해 보안부팅 인증을 수행하고, GRUB2에 대한 검증은 심 내부에 별도로 설계한 것이다. 본 설계를 통해 UEFI는 심에 대한 인증서 검증을, 심은 GRUB2에 대한 인증서 검증을 수행할 수 있도록 되었으며, 리눅스 환경에서도 원활한 보안부팅 기능을 지원할 수 있게 되었다.

심은 내부에 MOK(Machine Owner Key)라는 키 데이터베이스 구조를 통해 GRUB2를 비롯한 각종 서드파티 드라이버 인증서에 대한 공개키를 체계적으로 보관하고 있다. MOK의 키 관리 방식은 UEFI 보안부팅의 키 데이터베이스와 유사하게 허용된 키와 만료된 키를 구분하여 해당 드라이버의 실행 여부 제어가 가능하다. 하지만 MOK은 타 PC 운영체제 환경과 다르게 같은 리눅스 환경이라 할지라도 운영체제 개발자 또는 개발사가 다른 경우 MOK의 키 정보를 전부 갱신해야 한다는 번거로운 점이 존재한다. 이와 같은 호환성 문제로 인해 리눅스 진영은 아직 보안부팅을 적극적으로 권유하지 않으며, 각 사용자가 커널을 안전하게 사용하도록 권장하고 있다.

2. 스마트폰 환경의 보안부팅

2.1 Apple iOS

애플의 모바일 운영체제인 iOS는 스마트폰 시장에서 장기간 개발되어온 대표적인 운영체제이기도 하다. 특히나 iOS는 개발 초창기부터 애플에 의해 금지된 커널의 기능을 해제하기 위한 부트킷과 루트킷 공격, 통칭

탈옥(Jailbreak)으로 불리는 해커들의 공격으로부터 지속적으로 위협을 받고 있다. 애플은 이와 같은 공격으로부터 iOS를 보호하기 위해 보안부팅을 비롯한 다양한 보안 장치를 통해 지속해서 보안을 강화하고 있다[23].

우선 iOS는 ARM 보안프로세서를 사용하며, 외부와 격리된 프로세서 내부의 보안영역인 Trustzone에 저장된 자신의 무결성 정보를 RoT로 사용하며, 매 부팅 시 이를 기반으로 무결성의 검증을 수행한다[24]. 무결성 검증 과정에서는 연쇄적인 부트로더의 RSA 인증서 검증 방식을 사용하며, 만일 부트 코드의 무결성이 훼손된 경우 부팅을 중지한다. 다음으로는 커널의 무결성을 모니터링하는 기술인 KIP(Kernel Integrity Protection, 커널 무결성 보호)를 통해 임의의 앱으로 인한 커널 데이터의 조작, 훼손을 방지한다. 마지막으로 지속적인 업데이트이다. 애플은 iOS를 업데이트하면서 다양한 편의 기능을 제공함과 함께 시스템의 보안을 강화하는데, iOS가 업데이트되는 경우 이전 펌웨어 버전으로의 롤백(Roll-back)을 금지한다. 이를 통해 과거 펌웨어 버전의 취약점을 악용하고자 하는 공격으로부터 시스템을 안전하게 보호할 수 있다.

각종 취약점을 악용한 iOS에 대한 공격은 현재에도 활발히 이루어지고 있지만, 스마트폰 환경이 도입된 초창기에 비해서는 그 수가 현저히 줄었다고 한다. 그 사유로는 타인에 의한 부트킷 공격에 성공한다 할지라도 사용자가 이를 재부팅 하는 경우 보안부팅 단계에서 바로 공격 사실을 확인할 수 있으며, 사용자가 의도적으로 탈옥을 수행하는 경우 보안부팅이 요구되는 iOS의 각종 편의 기능을 제공받지 못하기 때문이다. 더욱이 모바일 기기 사용자의 보안인식수준 성장으로 인해 단순 호기심과 개인 만족을 위한 iOS의 탈옥은 다양한 보안 장치를 무력화시키고 자신의 보안을 포기하는 어리석은 행동이라고 생각하게 된 영향 또한 존재한다[25].

2.2 Google Android

모바일 환경의 OS인 안드로이드는 킷캣(Kitkat)으로 불리는 4.4 버전부터 보안부팅을 도입하였으며, iOS와 마찬가지로 ARM의 보안프로세서와 이를 통한 Trustzone 기술을 사용한다. 오픈소스 기반, 특유의 가변성으로 인해 다양한 기업에서 자사의 환경에 최적화된 구조로 새로이 디자인하는 경우가 많은 안드로이드는 다양한 환경에서도 효율적인 보안부팅 기능을 지원하고 있다[26].

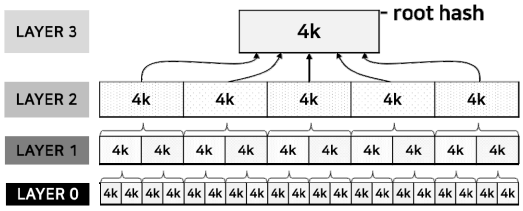

그림 6. dm-verity의 hash table 구조

첫 번째로 안드로이드는 RSA 인증서 기반의 보안부팅인 검증된 부팅(Verified boot)을 지원한다. RSA-2048 기준을 사용하는 다른 보안부팅 환경과 다르게 안드로이드에서는 최대 RSA-4096까지 지원하고 있으며 시스템, OEM, 벤더 등으로 구분된 부트 파티션을 통해 각 파티션 에 대한 무결성 검증을 수행한다. 두 번째로는 롤백방지이다. 안드로이드 또한 과거의 보안이 취약한 버전의 안드로이드를 실행하지 못하도록 관리하고 있다. 마지막으로는 dm-verity(device-mapper-verity)이다. 안드로이드는 다양한 기업에서 개발을 진행하고 있는 만큼, 다양한 애플리케이션에 대한 효율적인 무결성 관리가 요구된다. 이를 해결하기 위해 dm-verity는 저장소의 각종 영역에 대한 SHA256 해시를 수행하고, 이를 계층화하여 [그림 6]과 같은 트리 구조의 해시 테이블로 관리한다. 이로 인해 안드로이드의 검증된 부팅은 현재 저장소 영역에 대한 무결성을 검증하고자 하는 경우 테이블의 최상위 루트 해시만을 비교해도 충분하다. 만일 일부 저장소 영역에 대한 수정이 있는 경우에도 전체 영역에 대한 해시를 다시 수행하는 것이 아닌, 해당 영역과 관련된 테이블에 대해서만 해시를 갱신하면 충분하므로 시스템의 무결성을 효율적으로 관리할 수 있다.

안드로이드는 8.0 버전인 오레오(Oreo)부터 각종 보안 업데이트와 함께 검증된 부팅의 버전을 2로 업데이트하였다. 또한, 안드로이드를 개발하는 다양한 기업에서 독자적으로 보안부팅을 강화할 수 있는 환경인 프로

젝트 트레블(Project Treble) 설계를 지원하는 등, 개발 환경을 위한 다양한 보안 기능을 제공하고 있다[27].

하지만 안드로이드는 자체적으로 가진 일부 취약점이 존재한다. 안드로이드는 사용자에게 애플리케이션 설치에 대한 권한을 비교적 자유롭게 제공하기 때문에, 사용자가 악성 애플리케이션에 과도한 권한을 부여하는 경우 이를 악용한 각종 백도어와 트로이 목마와 같은 루트킷의 위협으로부터 사용자를 보호할 수 없다. 또한, 안드로이드를 개발하는 기업이 많아짐에 따라 관련 디바이스 제조사의 보안 신뢰성 검증이 부족하다는 지적을 받고 있다. 실제로 2019년 중국의 기업 화웨이(HUAWEI)의 백도어 보안위협 의혹으로 인해 구글은 뒤늦게 화웨이의 안드로이드 사용을 금지했고, 이 사건으로 인해 안드로이드 기기의 보안에 대한 불신이 증가하기도 하였다[28].

2.3 Samsung Knox

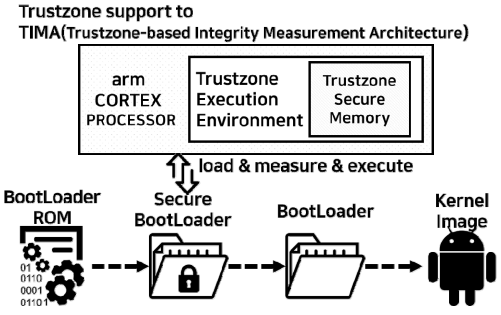

그림 7. Samsung Knox의 부팅 과정 다이어그램

안드로이드를 개발하는 스마트폰 제조사인 삼성은 스마트폰의 보안을 강화하기 위한 보안 솔루션인 Knox를 자사 제품에 탑재해 출시하고 있다. Knox는 안드로이드 상에서 강화된 보안이 필요한 기업 사용자를 위한 다양한 기능을 제공하고 있으며, 주요 정보의 격리, 사용자의 생체인식정보 관리 등을 수행하고 있다[29].

삼성은 [그림 7]과 같이 ARM Trustzone과 함께 자사의 추가적 보안 설계를 통해 보안부팅을 강화하고 있다. 우선 Knox가 탑재된 제품은 보안부팅 단계에서 자사의 보안 부트로더에 대해서 SSBK(Samsung Secure Boot Key, 삼성 보안 부트 키)를 기반으로 무결성 검증을 수행한다. SSBK는 제품의 OTP(One-Time Programmable) 퓨즈에 저장되어 수정이 불가능한 공개키로 현재 실행하고 있는 부트로더가 삼성에 의해 승인된 부트로더인지를 확인한다. 다음으로는 Knox 보증 퓨즈이다. Knox는 보안부팅이 올바르게 이루어지지 않거나, 커널에 대한 공격이 발생해 자신의 무결성이 훼손되었다고 판단하는 경우, 해당 퓨즈의 값을 설정하고 시스템에 저장된 중요 정보들을 전부 잠근다. 내부의 중요 정보를 전부 잠금으로 인해 발생하는 불편이 정보 유출로 인한 피해보다 낫다는 차선의 대책을 위해 설계된 방식이기도 하다. Knox는 위와 같은 보안부팅 관련 보안 기술 이외에도 독자적인 롤백 방지 기술을 비롯한 디바이스의 고유키, 원격증명을 위한 공개키 쌍 등 다양한 형태로 시스템을 보호하고 있다.

Knox는 위와 같이 다양한 방식으로 시스템의 무결성을 보호하고 있으며, 일부 규격은 그 보안 우수성을 인정받아 안드로이드의 보안 표준에 포함되기도 하였다[30]. 하지만 Knox의 보안 기능을 최대한 제공하고자 하는 경우 독자적으로 설계된 회로가 많은 점, 특정 기기 환경을 강제하는 점은 범용성의 측면에서 일부 아쉬운 상황이다.

3. 그 외 임베디드 환경의 보안부팅

3.1 Xilinx secure boot

임베디드 장치의 보안부팅 중요성이 강조됨에 따라, 다양한 임베디드 장치 개발기업에서는 보안부팅을 위한 설계를 추가시키기 시작하였다.

대표적으로 FPGA 제품 제조, 개발사인 자일링스(Xilinx)는 자사의 SoC 칩 Zynq-7000 이후의 설계에 보안부팅을 위한 회로를 추가해 제공하고 있다[31]. 추가한 회로에는 보안부팅에 사용할 RSA 키와 커널 이미지의 암호화에 사용하는 AES 키를 저장하기 위한 OTP 퓨즈가 포함되어 있으며, 이를 자일링스의 SDK 환경을 통해 간단하게 구현, 개발할 수 있도록 환경을 지원한다[32].

3.2 Cerberus project

임베디드 환경은 목적과 환경에 따라 매번 다른 설계

가 요구되고, 보안부팅 환경 또한 서로 다른 형태로 구현되는 경우가 일반적이었다. 하지만 마이크로소프트는 데이터 서버를 다루는 오픈소스 그룹인 OCP(Open Compute Project)의 설계를 기반으로 임베디드 구조의 클라우드 서버 설계 규격인 프로젝트 올림푸스(Project Olympus)를 제안하였다[33]. 당 프로젝트는 다양한 프로세서 기업과 소프트웨어 개발기업이 동참하고 있는 대규모 프로젝트로 발전하였으며, 현재에도 꾸준히 설계, 개발되고 있다.

프로젝트 올림푸스는 보안부팅과 관련된 규격에 대해서도 설계에 포함시켰는데, 이것이 프로젝트 케르베로스(Project Cerberus) 이다[34]. 프로젝트 케르베로스는 케르베로스 규격의 칩을 시스템을 포함하여 시스템과 연결되는 모든 장치에 탑재시키고, 해당 장치의 부팅부터 런타임 상태까지 모든 상태의 무결성을 보호하는 역할을 수행한다. 즉 보안부팅의 RoT를 메인 시스템에 국한시키는 것이 아닌, 메인 시스템과 연결된 모든 시스템에서 제공하는 설계가 가능한 것이다. 또한, 케르베로스가 탑재된 장치는 마찬가지로 케르베로스가 탑재된 장치와 신뢰된 연결을 통해 통신을 수행하며, 악성 공격으로부터 시스템을 보호할 수 있도록 설계되었다.

이처럼 오픈소스 방식의 체계적인 시스템 설계를 바탕으로 개발되고 있는 프로젝트 올림푸스는 기술의 표준화를 위한 기업 간의 성공적인 협업으로 평가받고 있으며, 기기 간의 호환성과 보안성을 모두 고려한 케르베로스 규격 또한 차세대 보안 방식으로 기대되고 있다.

Ⅳ. 각 보안부팅의 비교

본 논문의 3장에 설명한 각 보안부팅을 종합하여 정리하면 [표 1]과 같다. 세 종류의 플랫폼 환경 중, 대표적으로 PC 환경의 보안부팅을 비교해보자면, 마이크로소프트의 경우 범용성이 뛰어나고 TPM 등을 통해 기능의 확장이 가능하지만, 기존 UEFI 환경의 취약점인 물리적인 탈취로 인한 위협은 해결하지 못했다. 애플의 경우 자사 제품에 최적화된 보안을 지원하지만 이로 인해 하드웨어 장치의 추가나 변경이 어려운, 일부 경직된 시스템 환경을 제공하고 있으며, 리눅스의 경우 개발의 자유도를 위해 보안부팅의 비중을 크게 중요시하지 않은 모습을 보인다.

이처럼 보안부팅이 개발되는 방향은 비록 같은 플랫폼 환경일지라도 이를 개발하는 각 기업과 단체의 목적에 따라 무결성과 가용성을 고려한 설계가 다름을 알 수 있다. 더욱이 임베디드 요소가 강한 스마트폰 환경의 경우 기업별 독자적 보안부팅 설계의 차이가 더욱 도드라지며, 이는 보안성을 중요하게 고려하는 소비자에게 어필하기 위한 마케팅 요소로도 활용되기도 한다.

표 1. 보안부팅을 지원하는 각 환경의 비교

Name	Provider	Target Env.	Root of Trust	Purpose of Design	Weakness/Vulnerability
Windows	Microsoft	PC UEFI	BootROM	Universality, Scalability	Device hijack
macOS	Apple		BootROM, T2 Chip	Optimization	Restricted environment
Linux	Open-source	PC UEFI, Embedded	BootROM	Scalability, Free to use	Lack of development
iOS	Apple	Smartphone	Security data in AP, Network Attestation	Optimization, System Recovery	Jailbreak Threat
Android OS	Google		BootROM, OTP fuses	Universality, Variability, Efficiency	Many Malicious application
Knox	Samsung	Smartphone, Embedded	SSBK, SAK, Knox Warranty fuses	Optimization, for Enterprise	Company-dependency
Zynq-7000	Xilinx	Embedded	OTP fuses	Easy to development	Depends on development
Project Cerberus	Open-source	Various system	ROM in Cerberus	Universality, Standardization	Unknown

V. 결론

현 컴퓨터 디바이스 시장은 기존의 컴퓨터 생태계 환경에서 나아가 의료, IoT, 커넥티드 카, 드론 등 다양한 분야가 생겨나고 있고, 나날이 그 규모가 발전하고 있다. 이들이 인간의 삶과 밀접하게 연결됨에 따라 이를 위해 일정 수준의 보안이 요구되고 있다[35]. 보안부팅은 그 보안 수준을 충족시키기 위한 중요 시스템 무결성 보장 방법으로 고려되고 있으며, 이는 특히 사람의 생명과 밀접한 연관을 지닌 의료, 군사 분야에서 요구되는 사항이기도 하다.

이에 본 논문은 현대의 보안부팅 개발 양상을 조사하기 위해 보안부팅이 적용된 대표적 컴퓨터 플랫폼들인 PC와 스마트폰, 임베디드의 세 가지 환경으로 나누어 알아보고자 하였다. 주요 조사 분석 대상은 보안부팅의 수준과 포함된 기술, 그리고 보안 구조의 전반적인 목적이었다.

조사 결과 같은 플랫폼에서 발전한 각 보안부팅은 일부 유사한 요소도 있었지만, 이를 개발한 기업에 따라 서로 다른 목적이 있음을 알 수 있었다. 특히 애플의 경우 자사의 제품에 최적화된 보안 수준을 유지하기 위해 시스템의 전반적인 경직도가 존재하였으며, 마이크로소프트의 경우 범용성과 가용성이 고려된 Windows의 보안 부팅을 지원함과 동시에 오픈소스 프로젝트인 올림푸스 개발에 기여하는 등 자사의 기술을 통해 확장성이 뛰어난 시스템을 구축하고자 하는 노력이 드러났다. 현재 전자제품 시장은 이처럼 적용 플랫폼과 각 기업의 성향, 사상에 따라 서로 다른 보안부팅이 개발되고 있음을 알 수 있었다.

하지만 보안부팅의 활발한 발전 이면에는 몇 가지의 한계점이 드러난다. 대표적인 두 요인을 꼽자면 우선 지나친 기업 의존도이다. 현대 전자제품 시장의 보안부팅은 보안부팅을 개발한 기업 제품의 보안성을 강조하기 위한 요소로 활용되며, 이를 굳이 규격화하거나 공개하려 하지 않고 있다. 공개되지 않은 보안부팅 구조는 결국 제로데이 공격에 취약할 수밖에 없으며, 다양한 검증 방식을 통한 교차검증에도 어려움이 있다. 보안 시스템이 필요한 소비자는 이를 유념하여 신중한 제품선택을 해야 할 것이다. 다음으로는 가용성의 훼손이다. 보안부팅은 크든 작든 결국 시스템의 조작과 변경을 방지하는 구조로 설계되며, 이는 사용자의 선택권을 제한하는 결과로 이어진다. 보안부팅 구조가 강력할수록 이러한 경향이 짙어지므로, 기업에서는 보안성과 가용성을 적절히 조율한 보안부팅을 개발해야 할 필요가 있다.

향후 연구에서는 본 논문에서 조사한 정보를 바탕으로 보안부팅의 규격화에 요구되는 보안 기술에 대한 조사와 더불어 미래의 보안부팅에 대한 개념적인 설계를 제안하고자 한다. 앞으로 보안부팅이 더욱 발전함에 따라 많은 시스템에서 보안부팅을 필수적인 보안 기능으로 채택되리라 기대하고 있다. 본 논문에서 조사한 정보가 보안부팅을 채택하고자 하는 다양한 분야의 개발자에게 기여할 수 있기를 기대한다.

참 고 문 헌

[1] A. Matrosov, *Rootkits and Bootkits: Reversing Modern Malware and Next Generation Threats*, No Starch Press, 2019.

[2] M. Sabt, M. Achemlal, and A. Bouabdallah, “Trusted Execution Environment: What It is, and What It is Not,” 2015 IEEE Trustcom/BigDataSE/ISPA, pp.57-64, 2015.

[3] G. Bowen and W. Ozuem, *Computer-Mediated Marketing Strategies: Social Media and Online Brand Communities*, IGI Global, 2014.

[4] J. P. John, A. Moshchuk, S. D. Gribble, and A. Krishnamurthy, “Studying spamming botnets using Botlab,” Proceedings of the 6th USENIX symposium on Networked systems design and implementation, pp.291-306, 2009.

[5] https://www.microsoft.com/security/blog/2017/06/27/new-ransomware-old-techniques-petya-adds-worm-capabilities/

[6] C. Wressnegger, F. Yamaguchi, D. Arp, and K. Rieck, “Comprehensive Analysis and Detection of Flash-Based Malware,” Proceedings of the 13th International Conference on Detection of Intrusions and Malware, and Vulnerability

Assessment, pp.101-121, 2016.
[7] https://encyclopedia.kaspersky.com/glossary/bootkit/
[8] https://encyclopedia.kaspersky.com/glossary/rootkit/
[9] W. A. Arbaugh, D. J. Farber, and J. M. Smith, "A secure and reliable bootstrap architecture," Proceedings, 1997 IEEE Symposium on Security and Privacy (Cat. No.97CB36097), pp.65-71, 1997.
[10] UEFI, *Unified Extensible Firmware Interface Specification Version 2.3.1 Errata C*, UEFI Forum, 2012.
[11] https://techcommunity.microsoft.com/t5/Windows-Hardware-Certification/Microsoft-UEFI-CA-Signing-policy-updates/ba-p/364828
[12] https://lwn.net/Articles/447381/
[13] https://www.welivesecurity.com/2018/09/27/lojax-first-uefi-rootkit-found-wild-courtesy-sednit-group/
[14] V. Bashun, A. Sergeev, V. Minchenkov, and A. Yakovlev, "Too young to be secure: Analysis of UEFI threats and vulnerabilities," 2013 14th Conference of Open Innovations Association, pp.16-24, 2013.
[15] https://docs.microsoft.com/ko-kr/windows-hardware/drivers/install/early-launch-antimalware
[16] https://docs.microsoft.com/ko-kr/windows-hardware/drivers/install/elam-driver-requirements
[17] https://docs.microsoft.com/ko-kr/windows/security/threat-protection/intelligence/virus-initiative-criteria
[18] Apple, *Apple T2 Security Chip*, Apple Inc., 2018.
[19] https://www.slashgear.com/apple-t2-chip-confirmed-to-have-kill-switch-for-diy-repairs-12553488/
[20] https://appleinsider.com/articles/19/04/09/apples-t2-chip-makes-a-giant-difference-in-video-encoding-for-most-users
[21] https://en.wikipedia.org/wiki/GNU_GRUB
[22] https://docs.fedoraproject.org/en-US/Fedora/18/html/UEFI_Secure_Boot_Guide/sect-UEFI_Secure_Boot_Guide-Implementation_of_UEFI_Secure_Boot-Shim.html
[23] Apple, *iOS Security: iOS 12.3*, Apple Inc., 2019.
[24] ARM, *ARM Security Technology: Building a Secure System using Trustzone® Technology*, ARM Limited., 2009.
[25] https://www.wired.com/story/ios-jailbreak-new/
[26] https://source.android.com/security/verifiedboot
[27] https://source.android.com/devices/architecture
[28] https://news.joins.com/article/23472884
[29] Samsung Research America, *Whitepaper: Samsung Knox Security Solution*, Samsung Electronics, 2017.
[30] https://android-developers.googleblog.com/2014/07/knox-contribution-to-android.html
[31] Xilinx, *Zynq-7000 SoC Technical Reference Manual*, Xilinx, 2018.
[32] E. Peterson, *Secure Boot of Zynq-7000 SoC (XAPP1175)*, Xilinx, 2019.
[33] https://github.com/Project-Olympus
[34] B. Kelly, *Project Cerberus Security Architecture Overview Specification*, Open Compute Project, 2017.
[35] 고재용, 이상길, 김진우, 이철훈, "IoT 보안 요구사항 및 보안 운영체제 기반 기술 분석," 한국콘텐츠학회논문지, 제18권, 제4호, pp.164-177, 2018.

저 자 소 개

김 진 우(Jin-Woo Kim) 준회원

- 2017년 8월 : 충남대학교 컴퓨터공학과(공학사)
- 2017년 9월 ~ 현재 : 충남대학교 컴퓨터공학과 석사과정

〈관심분야〉 : 임베디드 시스템, 임베디드/IoT 보안, 실시간 시스템

이 상 길(Sang-Gil Lee) 정회원

- 2014년 2월 : 충남대학교 컴퓨터공학과(공학사)
- 2016년 2월 : 충남대학교 컴퓨터공학과(공학석사)
- 2018년 2월 : 충남대학교 컴퓨터공학과 박사과정 수료

〈관심분야〉 : 실시간 운영체제, 임베디드 시스템

이 정 국(Jeong-Guk Lee) 준회원

- 2011년 2월 : 인하대학교 정보통신공학과(공학사)
- 2012년 1월 ~ 2018년 4월 : LIG 넥스원 재직
- 2019년 3월 ~ 현재 : 충남대학교 컴퓨터공학과 석사과정

〈관심분야〉 : 실시간 운영체제, 임베디드 시스템

이 상 한(Sang-Han Lee) 정회원

- 2016년 2월 : 경북대학교 대학원 전자공학과(공학박사)
- 1997년 3월 : 국방과학연구소 연구원
- 2000년 2월 : ETRI부설연구소 연구원
- 2019년 현재 : ETRI부설연구소 책임연구원

〈관심분야〉 : 암호화 SoC, 보안부팅

신 동 우(Dong-Woo Shin) 정회원

- 2015년 2월 : 한국과학기술원 전기 및 전자공학과(공학석사)
- 2015년 1월 ~ 현재 : ETRI부설연구소 연구원 재직

〈관심분야〉 : 신호처리, 컴퓨터 구조, 보안기술

이 철 훈(Cheol-Hoon Lee) 정회원

- 1983년 2월 : 서울대학교 전자공학과(공학사)
- 1988년 2월 : 한국과학기술원 전기 및 전자공학과(공학석사)
- 1992년 2월 : 한국과학기술원 전기 및 전자공학과(공학박사)
- 1983년 3월 ~ 1986년 2월 : 삼성전자 컴퓨터사업부 연구원
- 1992년 3월 ~ 1994년 2월 : 삼성전자 컴퓨터사업부 선임연구원
- 1994년 2월 ~ 1995년 2월 : Univ. of Michigan 객원연구원
- 1995년 5월 ~ 현재 : 충남대학교 컴퓨터공학과 교수
- 2004년 2월 ~ 2005년 2월 : Univ. of Michigan 초빙연구원

〈관심분야〉 : 실시간 시스템, 운영체제, 고장허용 컴퓨팅, 로봇 미들웨어