Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.31, NO.2, Apr. 2021 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2021.31.2.167

신뢰 실행 환경의 안전한 TPM 활용을 위한 보안 기술*

한 승 균,[1†] 장 진 수[2‡]
[1,2]충남대학교 (대학원생, 교수)

Secure Mobile TPM Adoption for the Trusted Execution Environment*

Seung-Kyun Han,[1†] JinSoo Jang[2‡]
[1,2]Chungnam National University (Graduate student, Professor)

요 약

ARM TrustZone과 같은 신뢰 실행 환경(Trusted Execution Environment, TEE) 기술은 보안에 민감한 데이터나 중요 로직을 보호하기 위해 사용되고 있다. 특히 암호 연산은 신뢰 실행 환경 내에서 실행되는 중요 로직 중의 하나로써 디지털 권한 관리(Digital Right Management, DRM)와 같은 보안기술 구현에 활용되고 있다. 하지만 Raspberry Pi와 같은 저사양 기기의 경우 높은 엔트로피를 제공하는 암호화 하드웨어 대신 상대적으로 취약한 소프트웨어 기반 암호화 기술(예: 의사 난수)에 의존한다. 본 논문은 저사양 기기상 신뢰 실행 환경의 보안성 향상을 위해 가상화 기술을 활용하여 모바일 TPM(Trusted Platform Module)과 신뢰 실행 환경을 안전하게 접목하는 방법을 제안한다. 제안된 방식은 신뢰 실행 환경 외에서 TPM 커널 모듈을 안전하게 재사용할 수 있으므로 기존 신뢰 실행 환경에 새로운 위협요소를 추가하지 않는다는 장점을 지닌다.

ABSTRACT

The trusted execution environment (TEE) such as ARM TrustZone is widely adopted to protect security-critical logic and data. Specifically, the crypto operation is generally hosted in the TEE and leveraged to build various trusted services (e.g., DRM). Although the crypto operation plays a critical role in the TEE, low-end devices depend on a software-based crypto service, which provides a limited randomness-entropy than that of the hardware-based crypto service. To alleviate this problem, we propose a way to efficiently combine the mobile trusted platform module (TPM) with the TEE. We utilize hardware-assisted virtualization to reuse and protect the TPM kernel driver instead of directly porting it to the TEE. By doing so, we minimize the potential threat that may be introduced by bloating the TEE.

Keywords: Trusted Execution Environment, Trusted Platform Module, Virtualization, Mobile Security

I. 서 론

오늘날 IT 기술의 발전은 사용자에게 다양한 편의를 제공하는 반면 개인정보 유출과 같은 보안사고 증가의 원인이 되기도 한다. 예를 들어, 토스, 카카오페이, 삼성페이 등의 간편 지불 시스템은 결제에 필요한 정보를 모바일 디바이스 내에 저장 가능하게 하지만 디바이스가 해킹될 경우 중요 정보가 외부로 유출될 위험성을 수반한다. 또한, 운송, 농업, 제조 등 다양한 분야에서 사물 인터넷 기술이 적용되면서 디바이스 내에 저장된 민감 정보를 탈취하기 위한 공격 또한 증가하고 있다.

Received(02. 03. 2021), Modified(03. 29. 2021), Accepted(03. 29. 2021)
* 본 논문은 2020년도 한국정보보호학회 동계학술대회에 발표한 우수논문을 개선 및 확장한 것임.
* 본 논문은 정보통신기획평가원(IITP-2019-0-01343, IITP-2020-0-01840) 및 한국연구재단(NRF-2020R1F1A1058305)의 지원을 받아 수행된 연구임.
† 주저자, yakmg3000@o.cnu.ac.kr
‡ 교신저자, jisjang@cnu.ac.kr(Corresponding author)

이러한 문제를 해결하기 위해 디바이스 내에 일반 실행 환경(REE)과 격리된 별도의 신뢰 환경을 구성하여 암호키나 연산 등 보안에 민감한 자원들을 안전하게 보호하기 위한 신뢰 실행 환경 기술이 활용되고 있다. 예를 들어, 모바일 및 사물 인터넷 기기들에서 주로 사용되는 ARM 프로세서의 보안 기능인 TrustZone은 메모리, 주변 장치 등을 보호하기 위한 하드웨어 기능 제공을 통해 디바이스 내에서의 신뢰 실행 환경 구성을 가능하게 한다. 이렇게 생성된 신뢰 실행 환경은 하드웨어 기반 암호화 기술[1]의 도입을 통해 다양한 보안 응용 서비스들을 제공한다. 가령, 암호화된 디지털 컨텐츠를 신뢰 실행 환경 내에서 안전하게 복호화하는 디지털 권한 보호(DRM), 하드웨어 암호화 엔진을 통해 생성된 키를 활용한 데이터 암/복호화 서비스 등이 구현될 수 있다. 즉, 두 기술은 상호보완 관계로서 하드웨어 암호화 기술은 높은 엔트로피를 기반으로 한 난수 및 암호화 키 생성을 가능하게 하고 신뢰 실행 환경은 이러한 암호화 자원에 대한 안전한 접근을 보장한다.

하지만 이러한 신뢰 실행 환경 기반 보안 기술의 활용을 염두에 두고 디바이스를 설계하지 않는 한 필수 보안 하드웨어 기능이 누락되는 경우가 발생한다. 가령, Raspberry Pi와 같은 저가용 디바이스의 경우 하드웨어 암호화 엔진이 제공되지 않기 때문에 신뢰 실행 환경 내의 암호화 연산들은 상대적으로 안전하지 않은 소프트웨어 기반 의사 난수 생성기(Pseudo Random Number Generator, PRNG)에 의존한다. 또한, 신뢰 실행 환경 내에서 하드웨어 암호화 엔진에 접근하기 위한 소프트웨어가 구현되지 않은 경우도 존재한다.

본 논문에서는 저가형 기기의 신뢰 실행 환경에서 하드웨어 기반 암호화 기술을 안전하게 사용할 수 있는 방법을 제시한다. 모바일 TPM을 하드웨어 암호화 엔진으로 사용하고 TPM에서 생성된 결과 값(난수 또는 암호키)의 외부(일반 실행 환경) 노출 없이 신뢰 실행 환경으로 안전하게 읽어오기 위해 가상화 기술을 활용한다. 즉, TPM의 결과 값을 출력하기 위한 물리 메모리 영역을 가상화 기술의 중첩 페이징 기술을 활용하여 쓰기 전용으로 강제한다. 이를 통해 커널은 TPM에 요청을 전달할 수 있지만, TPM에 의해 생성된 값은 읽을 수는 없다. 대신 TPM의 결과 값은 신뢰 실행 환경 내에서 암호화 연산을 수행하는 프로세스에 의해서만 접근하여 사용할 수 있도록 보장한다.

시제품은 ARMv8 Cortex-A53 멀티코어 기반의 Raspberry Pi 3B+ 보드와 Infineon사의 TPM을 활용하여 구현하였다. 또한, 신뢰 실행 환경 소프트웨어로써 OP-TEE[2]를 사용하였다. 리눅스의 부팅과정과 관련된 소스 코드와 신뢰 펌웨어 소스 코드를 수정하여 가상화 기술의 중첩 페이징을 활성화하고 TPM과의 통신을 위한 물리 메모리를 가상화 수준에서 보호하였다.

2절 "관련 연구"에서는 신뢰 실행 환경, 가상화 기술, TPM과 관련된 관련 연구를 간략하게 소개한다. 3절 "배경 지식"에서는 ARM TrustZone, 가상화 기술, TPM, 커널 디바이스 드라이버에 관한 내용을 다룬다. 4절에서는 공격 모델을 상정하고, 5절 "제안 기법"에서는 본 논문에서 제안하는 기법을 상세하게 기술한다. 6, 7절에서는 제안된 기법의 구현 방법과 성능 및 가상화 기술 적용으로 인한 시스템의 성능 저하 측정 결과를 기술한다. 마지막으로 8절에서 결론을 맺는다.

II. 관련 연구

Trustvisor[3]는 가상화 기술의 중첩 페이지 테이블을 사용하여 보안이 필요한 중요한 로직(PAL, Pieces of Application Logic)을 일반 실행 환경의 OS 커널 및 어플리케이션과 분리할 수 있는 방안을 제시하였다. PrivateZone[4] 또한 가상화 기술을 활용하여 일반 실행 환경과 격리된 PrEE(Private Execution Environment)라는 독립적인 실행 환경을 제공함으로써 유저 어플리케이션의 중요 로직을 보호하였다.

Sanctuary[5]는 하드웨어 기반 메모리 보호 기능인 TZASC을 활용하여 유저 애플리케이션의 안전한 실행을 보장한다. 각 CPU 별로 다른 NSAID(Non-Secure Access Identity)를 부여하고, NSAID를 기반으로 특정 메모리 영역에 하나의 CPU만 접근할 수 있도록 TZASC를 설정한다. 해당 메모리 영역은 TZASC에 의해 설정된 전용 코어만 접근 가능하므로 새로운 신뢰 실행 환경을 제공 가능하다. SKEE[6]와 Hilps[7]는 ARM 아키텍처의 가상화 시스템과 관련된 기능들을 활용하여 신뢰 실행 환경을 구성하였다.

한편 Trustvisor[3]와 Fliker[8]는 x86 아키텍처상의 TPM을 활용하여 신뢰 실행 내에서 실행되는 코드에 대한 무결성을 검증하였다. fTPM[9]은

ARM TrustZone을 활용하여 소프트웨어 기반의 TPM을 구현하였다.

III. 배경 지식

3.1 ARM TrustZone

ARM TrustZone은 ARM 아키텍처에서 제공하는 보안 기술로서 시스템을 두 개의 논리적인 실행 환경-일반 실행 환경(REE) 및 신뢰 실행 환경(TEE)-으로 구성할 수 있게 한다 [Fig. 1]. ARM 아키텍처에서 정의된 CPU 모드 중 유저(EL0), 커널(EL1), 하이퍼바이저(EL2) 모드는 실행 환경과 무관하게 제공된다(TEE 하이퍼바이저의 경우 ARMv8.4 버전 이후부터 제공). 모니터 모드(EL3)는 보통 일반 실행 환경과 신뢰 실행 환경 사이의 변경에 필요한 작업(예: CPU 문맥 저장 및 복구)을 처리하며 시스템상의 가장 높은 권한을 가진다. 신뢰 실행 환경의 보안성을 보장하기 위해 다양한 TrustZone 하드웨어가 제공된다. 예를 들어, TZASC는 DRAM 상의 신뢰 실행 환경 영역에 대한 보호를 수행한다. TZMA와 TZPC는 각각 SRAM과 주변 장치에 대한 접근 제어를 수행한다. 가령, 특정 메모리 영역(예, 하드웨어와의 통신을 위한 공유 메모리)이 TZASC 설정에 의해 신뢰 실행 환경에서만 읽고 쓰기 가능하도록 설정되어 있다면, 주변 장치가 해당 메모리 영역에 접근하기 위해서는 TZPC 설정을 통해 신뢰 모드로 전환되어야 한다.

일반 실행 환경과 신뢰 실행 환경은 별도의 운영체제 및 어플리케이션을 기반으로 각각의 실행 환경을 구성한다. 본 논문에서는 신뢰 실행 환경에서 실행되는 어플리케이션을 신뢰 어플리케이션(Trusted application, TA)으로 명명하며, TA에게 서비스 요청을 하는 일반 실행 환경의 어플리케이션을 클라이언트 어플리케이션(client application, CA)으로 칭한다.

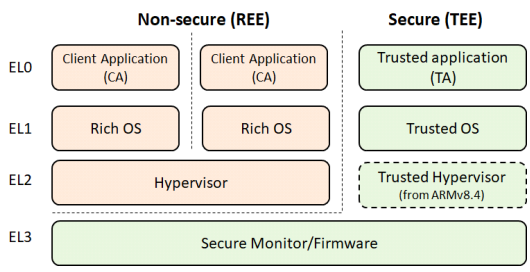

Fig. 1. ARMv8 Architecture

3.2 가상화 기술

최신 ARM 프로세서는 하드웨어 기반 가상화 기능을 지원한다. 하이퍼바이저(EL2) 모드는 가상 머신을 관리하기 위한 하이퍼바이저가 실행되는 특권 CPU 모드로서 커널 모드(EL1)보다 높은 권한을 가진다. 하이퍼바이저 모드에서는 캐시 관리나 시스템레지스터 변경 등 가상 머신에서 실행되는 운영체제의 특권 명령 실행을 감시하고 트랩 할 수 있다. 또한, 가상 머신의 중간 물리 주소(IPA, Intermediate Physical Address)를 실제 머신 주소로 변환하기 위한 중첩 페이징 기술을 지원한다. 이러한 하드웨어 기반 가상화 기술은 다양한 보안 기술을 구현하기 위해 활용되어 왔다. 가령, 삼성 모바일 폰의 보안 기술인 녹스(Knox)는 운영체제의 무결성을 보호하기 위해 중첩 페이징을 활용하여 중요 영역(코드 및 데이터)의 접근 권한을 읽기 전용으로 강제한다.

3.3 TPM 및 커널 디바이스 드라이버

3.3.1 TPM(Trusted Platform Module)

TPM(Trusted Platform Module)은 TCG(Trusted Computing Group)에서 정의한 보안 사양을 구현한 하드웨어 암호화 모듈로써 비대칭 키 생성, 암/복호화, 해싱, TPM 간의 키 서명 및 마이그레이션, 난수 생성 등을 수행한다. 보드에 물리적으로 부착되어 사용되며, 시스템에서 실행되는 소프트웨어는 TCG에서 정의한 명령을 사용하여 TPM을 제어할 수 있다.

3.3.2 커널 디바이스 드라이버

디바이스 드라이버는 하드웨어를 제어하기 위해 커널 모듈 형태로 동작하는 프로그램이다. 사용자가 open(), read(), write()와 같은 입출력 함수를 호출하여 제어하고자 하는 디바이스 파일에 접근하면, 커널 디바이스 드라이버는 파일 오퍼레이션(file_operation) 구조체를 통해 사용자가 호출한 서비스에 대응된 커널 함수를 호출하여 디바이스를

제어한다.

3.3.3 TPM 디바이스 드라이버

TPM을 제어하기 위한 오픈소스 TPM 디바이스 드라이버는 리눅스에 기본적으로 적재되어 제공된다.

TPM 디바이스 드라이버에서 정의된 함수들은 사용자가 요청한 TPM 서비스와 관련된 TCG 명령어를 공유 메모리에 기록하는 방식으로 TPM 하드웨어와 통신한다. 가령, 사용자가 TPM 기반 난수 생성 서비스를 요청하면 디바이스 드라이버는 사용자의 요청에 대한 TPM 명령을 공유 메모리에 기록한다. TPM은 디바이스 드라이버에 의해 기록된 명령을 읽고 난수를 생성 후 공유 메모리에 반환한다. 디바이스 드라이버는 해당 난수를 사용자 영역에 반환한다.

IV. 공격 모델

일반적으로 신뢰 실행 환경이 적용된 환경에서 가정하는 공격 모델을 상정한다. 즉, 일반 실행 환경의 운영체제는 신뢰할 수 없으며, 공격자는 자신이 원하는 임의의 코드를 실행할 수 있다. 이러한 강력한 권한을 바탕으로 일반 실행 환경 내의 소프트웨어를 공격하는 것을 물론, 신뢰 실행 환경에 대한 공격 또한 수행할 수 있다. 특히, 본 논문에서 가정하는 주요 공격은 신뢰 실행 환경과 사용자가 추가한 하드웨어 간의 통신 채널을 변조하거나 도청하는 것이다. 가령, 공격자는 TPM과 데이터를 주고받는 공유 메모리에서 TPM의 출력값(난수 또는 암호키)을 가로채거나 변조할 수 있다. 또한, 공격자는 해당 하드웨어와 통신하는 악의적인 디바이스 드라이버 적재를 통해 하드웨어의 동작을 임의로 조작할 수 있다.

반면, 공격자는 기기에 대한 물리적인 접근 권한을 가지고 있지 않다고 가정한다. 따라서 JTAG디버거와 같은 하드웨어 장비를 이용한 분석이나 물리적인 메모리 접근에 기반을 둔 공격(예, 콜드부트 공격[12]) 등은 불가능하다. 또한, 디바이스 전력 사용량 분석, 최신 CPU 취약점을 악용한 공격[13,14] 등은 고려하지 않는다. 또한, 본 논문에서는 모니터링 기법(예, TIMA: TrustZone-based Integrity Measurement Architecture[15])에 의해 커널 정적 영역의 무결성이 보장된다고 가정한다.

V. 제안기법

TPM 하드웨어와 통신하기 위한 공유 메모리는 커널 영역에 할당되어 사용된다. 따라서 공격자에 의해 커널이 제어될 경우 TPM이 생성한 중요 데이터(난수, 암호키 등)가 유출될 위험성이 존재한다. 또한, 3.1장에서 논의했던 것과 같이 하드웨어가 직접 신뢰 실행 환경과 통신하기 위해선 TZASC, TZPC의 설정이 필수적이다. 하지만 TZASC, TZMA, TZPC 등과 같은 TrustZone 기능의 설정은 신뢰 실행 환경에서만 허용되기 때문에 신뢰 실행 환경 내의 코드 수정이 불가피하다. 하지만 신뢰 실행 환경 내의 펌웨어 및 운영체제에 대한 변경 및 새로운 코드 추가는 공격 벡터를 증가시키는 결과로 이어질 수 있다[16,17]. 따라서 본 논문에서는 신뢰 실행 환경에 대한 변경을 최소화하기 위해 신뢰 실행 환경 외부의 신뢰 기반(가상화의 중첩 페이징 기술)을 바탕으로 공유 메모리 영역을 커널로부터 보호함으로써 TPM과 안전한 통신을 수행하는 방안을 제시한다. 방법을 제안한다. 특히, TPM의 결과 값을 안전하지 않은 일반 실행 환경의 사용자 영역으로 반환하는 대신 신뢰 실행 환경으로 전달하여 안전하게 사용할 수 있도록 한다.

5.1 신뢰 실행 환경의 안전한 TPM 활용

가상화 기반 쓰기-전용 메모리 구성 및 접근 제어를 통해 신뢰 실행 환경에서 안전하게 TPM을 활용하는 방법을 제안한다[Fig. 2].

우선 가상화 기술의 중첩 페이징을 활성화하여 TPM에 요청을 전달하기 위한 메모리 영역과 TPM

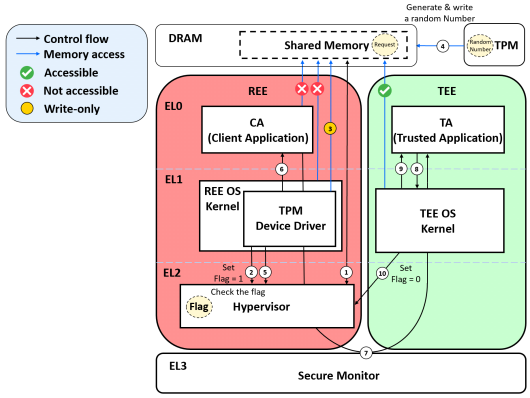

Fig. 2. Protected TPM access from TEE

의 응답을 기록하기 위한 메모리 영역을 일반 실행 환경에서 접근 불가능하도록 설정한다. 따라서 TPM 디바이스 드라이버를 포함한 모든 커널 프로세스는 해당 공유 메모리 영역들에 접근 불가능해진다. 이후, TPM 디바이스 드라이버를 실행하는 CPU의 중첩 페이지 테이블을 조작하여 TPM에게 요청을 전달하기 위한 공유 메모리 영역을 일시적으로 쓰기-전용 메모리로 재설정함으로써 사용자의 TPM 사용 요청이 처리될 수 있도록 한다. 일반 실행 환경에서는 공유 메모리에 대한 읽기 권한이 부여되지 않으므로 공격자는 TPM이 생성한 데이터를 읽을 수 없다. 이러한 제안 기법의 상세한 실행 흐름은 다음과 같다.

① 하이퍼바이저는 중첩 페이징[10] 기술을 이용하여 TPM과 통신하기 위한 공유 메모리 영역을 일반 실행 환경에서 접근 불가능하도록 강제한다. 단, 신뢰 실행 환경에서는 중첩페이징의 설정과는 무관하게 공유 메모리에 접근할 수 있다. ② TPM 디바이스 드라이버는 TPM 하드웨어에 난수 생성을 요청하기 위해 하이퍼바이저에게 공유 메모리에 대한 쓰기-전용 권한(Write-Only)을 요청한다. ③ 하이퍼바이저는 공유 메모리에 대한 쓰기 전용 권한을 현재 CPU 코어에게 부여한다. 쓰기-전용 권한을 부여받은 코어는 TPM의 난수 생성을 위한 요청 메시지를 공유 메모리에 기록한다. ④ TPM은 요청에 따라 난수를 생성 후 응답 전용 공유 메모리에 기록한다. ⑤ 디바이스 드라이버는 다시 하이퍼바이저를 호출하여 요청 전달을 위한 공유 메모리의 쓰기-전용 권한을 해제한다. 이는 TPM의 난수 생성 요청 후 즉시 이루어진다. ⑥ 디바이스 드라이버는 응답 전용 공유 메모리에 대한 물리 주소를 CA에게 전달한다. ⑦ CA는 디바이스 드라이버로부터 전달받은 물리 주소와 함께 난수 생성 알림 메시지를 TA에게 전송한다. ⑧ TA는 해당 물리 주소를 신뢰 실행 환경 OS에 전달하여 난수를 읽어올 것을 요청한다. ⑨ 신뢰 실행 환경 OS는 공유 메모리의 물리 주소를 신뢰 실행 환경 메모리 영역에 매핑하여 난수를 읽은 후 TA에게 반환한다. ⑩ TA의 난수 사용 완료 후, 신뢰 실행 환경 OS는 난수 사용이 완료되었음을 하이퍼바이저에 전달하여 추후 TPM 요청 전달을 위한 공유 메모리의 쓰기 전용 권한 요청이 수용될 수 있도록 한다.

5.2. 공유메모리 무결성 보호

공유 메모리에 대한 쓰기 전용 권한이 여러 CPU 코어에 동시에 제공될 경우 공격자는 TPM에 의해 생성된 난수 값을 자신이 원하는 값으로 변조할 수 있다. 이는 난수를 기반으로 TA에 의해 생성되는 암호키의 비밀성을 훼손할 수 있다. 이러한 문제는 일반 실행 환경에서 접근 불가능한 하이퍼바이저 영역의 플래그를 운용하여 방지한다. 즉, 하이퍼바이저는 공유 메모리에 대한 쓰기 전용 권한을 부여하는 동시에 플래그의 값을 설정하여 다른 CPU 코어가 해당 공유 영역에 대한 쓰기 권한을 추가적으로 부여받는 상황을 방지한다. 한 번 설정된 플래그 값은 공유 메모리상의 난수 값을 신뢰 실행 환경에서 읽은 후에 다시 해제된다.

VI. 구 현

본 연구의 프로토타입은 라즈베리 파이 3B+에서 구현되었으며, 일반 실행 환경 OS로서 raspbian-stretch(Kernel 4.14.98)를 사용하였다. 신뢰 실행 환경은 OP-TEE 버전 3.4를 포팅하여 구성하였다. TPM은 Infineon사의 OPTIGA™ TPM SLB 9670를 사용하였다. 또한, TPM을 제어하기 위한 어플리케이션은 Infineon에서 제공하는 eltt2(Embedded Linux Tpm Toolbox 2 for TPM 2.0)를 CA와 TA로 분리하여 구현하였다. 쓰기-전용 메모리 권한 부여를 요청하기 위한 하이퍼콜 및 공유 메모리에 대한 물리 메모리 주소를 CA에게 전달하기 위한 코드가 TPM 디바이스 드라이버에 추가되었다. 또한, TPM과의 공유 메모리에서 값을 읽는 역할을 수행하는 디바이스 드라이버의 일부 코드는 신뢰 실행 환경으로 이관되었다.

VII. 실험 및 평가

7.1 보안성 평가

4장의 공격 모델에서 논의하였듯이 공격자는 임의의 코드(악의적 디바이스 드라이버)를 자유롭게 수행할 수 있다. 하지만 TPM이 생성한 값은 보호되는 메모리 영역 내에 존재하기 때문에 해당 값을 읽거나 변조할 수 없다.

한 가지 공격 가능한 시나리오는 경쟁 관계를 이

용하는 것이다. 즉, 신뢰 실행 환경에서 TPM이 생성한 값을 읽기 전에 공격자는 TPM에 대한 다른 요청을 전달하여 공격자가 예상 가능한 값(예: 해시 값)으로 TPM 생성 값을 변조할 수 있다. 즉, TPM의 결과 값이 보호 메모리 영역에 쓰인 이후, 해당 영역에 대한 쓰기 권한이 해제되기 직전에 인터럽트와 같은 하드웨어 이벤트의 발생에 의해 공격자가 실행 흐름을 획득할 수 있다면 공격자는 TPM에 대한 새로운 요청을 전달할 수 있다.

이러한 공격은 TPM에 대한 요청을 전달하는 버스 컨트롤러 동작을 검증하는 방식으로 방어할 수 있다. 즉, 버스 컨트롤러가 사용하는 MMIO 영역을 중첩 페이징으로 보호하고 TPM에 전달되는 요청 메시지들을 하이퍼바이저 내의 메모리에 로깅 한다. 이후, 저장된 로그를 TPM 생성 값을 사용하는 시점에 확인하여 TPM 출력 값의 변조 여부를 검증할 수 있다. 가령, 정상적인 TPM 요청 후 추가적인 TPM 요청 메시지가 기록되어 있다면 악의적인 요청으로 간주할 수 있다. 이러한 추가적인 방어 기법은 향후 연구에서 다룰 예정이다.

7.2 성능평가

제안된 기법의 성능평가를 위해 운영체제 및 TPM을 활용하는 어플리케이션의 성능 저하를 측정하였다. 운영체제 성능평가를 위해서는 LMBench[11]를 사용하였으며, 어플리케이션 성능평가는 신뢰 실행 환경 및 리눅스 환경에서 TPM을 활용하는 어플리케이션의 성능을 비교하는 방식으로 수행하였다.

7.2.1 OS Microbenchmarks

LMBench를 사용하여 9개의 주요 OS 작업에 대한 실행 시간 및 처리량을 측정하였다. [Fig.3]의 그래프는 리눅스 환경 대비 가상화가 활성화된 상태에서의 LMBench[11]의 성능을 나타낸다. 즉, 100%에 가까울수록 성능 저하가 적음을 나타낸다. socket은 소켓에 의한 로컬 통신, simple read는 일정 크기의 메모리 할당 후 할당된 메모리 영역 읽기, mmap은 메모리 매핑 생성 및 해제, mmap_read는 메모리 매핑 생성 후 매핑된 메모리 영역 읽기, null은 null system call 등을 나타낸다. 실험 결과에서 볼 수 있듯이 주로 메모리 접근

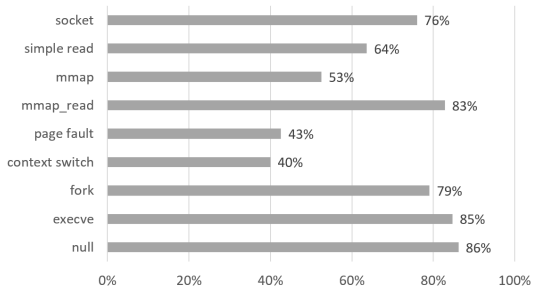

Fig. 3. LMBench result with nested paging-enabled environment (normalized to Linux)

및 페이지 테이블과 관련된 작업을 수행하는 (mmap, page fault, context switch) 테스트들에서 많은 성능 저하가 관찰되었다.

이러한 성능 저하는 제안된 기술의 중첩 페이징 기술 활용에 따른 MMU에서의 추가적인 메모리 주소 변환과 TLB 캐시 사용의 증가에 기인한 것으로 분석되었다. 즉, 기존의 가상-물리 주소 변환에서 중첩 페이징에 의한 가상-중간물리-물리 주소로의 추가적인 메모리 변환 과정이 수행됨에 따라 성능 저하가 발생한다. 여기에 메모리 보호를 위한 중첩 페이지 테이블 구성에 따른 추가적인 성능 저하가 발생한다. 32비트 환경에서 중첩 페이징을 이용한 메모리 매핑은 1GB, 2MB, 4KB 단위로 가능하다. 가장 큰 단위인 1GB 단위로 메모리 페이지를 매핑할 경우, 페이지 테이블 엔트리의 수가 감소하면서 TLB 캐시에 대한 부담을 줄일 수 있다. 하지만, TPM이 사용하는 공유 메모리 보호를 위해 물리 메모리의 일부분이 그보다 작은 단위인 2MB 및 4KB로 분할 매핑되면서 TLB 캐시에 대한 압박이 증가하고 이에 따른 추가 성능 저하가 발생하였다.

7.2.2 Application Benchmarks

일반적인 리눅스 환경에서 TPM 디바이스 드라이버를 통해 TPM 하드웨어를 사용하는 어플리케이션과 제안된 기법을 바탕으로 신뢰 실행 영역을 통해 TPM을 활용하는 어플리케이션의 성능을 측정하고 비교하였다. TPM 디바이스 드라이버는 일반적인 디바이스 드라이버 개발방식과 비슷하게 구현된다. 즉, open(), ioctl() 등의 시스템 콜 인터페이스를 통해 어플리케이션과 통신하며, copy_from_user와 같은 커널 함수를 통해 어플리케이션의 데이터를 커널 영역으로 읽어온다. 따라서 유저-커널 간의 통신을 위

한 작업들에 소요되는 시간이 일반적인 TPM 디바이스 드라이버의 성능을 좌우한다. 반면, 제안된 방식은 기존 작업들 외에 쓰기-전용 메모리 설정을 위한 하이퍼바이저 호출, 신뢰 실행 환경으로의 CPU 모드 전환, TPM 값 활용을 위한 TA 도입 등의 추가적인 요소들이 요구된다. 이러한 추가적인 작업들은 제안된 방식의 주요 성능 저하 요인이며 실험 결과 기존 방식 대비 최대 27%의 성능 저하가 관찰되었다.

VIII. 결 론

저사양 디바이스상 신뢰 실행 환경의 보안성 향상을 위해 모바일 TPM을 도입하고 이를 신뢰 실행 환경에서 안전하게 사용하기 위한 방법을 제안하였다. 가상화 기술을 활용하여 TPM과의 통신을 위한 메모리를 보호하고 신뢰 실행 환경에서 안전하게 사용 가능함을 보였다. LMBench 마이크로 벤치마크 성능 측정 결과 가상화 기술 활용에 기인한 최대 60% 정도의 성능 저하가 관찰되었다. 하지만 제안된 방식을 활용한 어플리케이션 성능평가에서는 최대 27% 정도의 성능 저하를 보였다.

References

[1] ARM, “Arm® TrustZone® CryptoCell-712 Revision 1.19,” ARM limited, June 2018.

[2] Linaro, “OP-TEE Documentation,” TrustedFirmware.org, April 2021.

[3] McCune, Jonathan M., et al., "TrustVisor: Efficient TCB reduction and attestation," 2010 IEEE Symposium on Security and Privacy, pp. 1-1, May 2010.

[4] Jang, Jinsoo, et al., "Privatezone: Providing a private execution environment using arm trustzone," IEEE Transactions on Dependable and Secure Computing, pp. 797-810, Oct. 2016.

[5] Brasser, Ferdinand, et al., "SANCTUARY: ARMing TrustZone with User-space Enclaves," Network and Distributed Systems Security (NDSS), pp. 1-1, Feb. 2019.

[6] Azab, Ahmed M., et al., "SKEE: A lightweight Secure Kernel-level Execution Environment for ARM," Network and Distributed Systems Security (NDSS), pp. 1-1, Feb. 2016.

[7] Cho, Yeongpil, et al., "Dynamic Virtual Address Range Adjustment for Intra-Level Privilege Separation on ARM," Network and Distributed Systems Security (NDSS), pp. 1-1, Feb. 2017.

[8] McCune, Jonathan M., et al., "Flicker: An execution infrastructure for TCB minimization," Proceedings of the 3rd ACM SIGOPS/EuroSys European Conference on Computer Systems, pp. 315-328, April 2008.

[9] Raj, Himanshu, et al., "fTPM: A Software-Only Implementation of a {TPM} Chip," USENIX Security Symposium, pp. 841-856, Aug. 2016.

[10] ARM, “Arm Architecture Reference Manual,” Armv8, June 2020.

[11] McVoy, Larry W., and Carl Staelin, "lmbench: Portable Tools for Performance Analysis," USENIX annual technical conference, pp. 23-39, Jan. 1996.

[12] Halderman, J. Alex, et al., "Lest we remember: cold-boot attacks on encryption keys," Usenix Security Symposium, pp. 45-60, Aug. 2008

[13] Kocher, Paul, et al., "Spectre attacks: Exploiting speculative execution," IEEE Symposium on Security and Privacy (SP), pp. 1-19, May 2019.

[14] Lipp, Moritz, et al., "Meltdown: Reading kernel memory from user space," USENIX Security Symposium, pp. 973-990, Aug. 2018.

[15] Ahmed M. Azab, et al., "Hypervision Across Worlds: Real-time Kernel Protection from the ARM TrustZone

Secure World," Proceedings of the 2014 {ACM} {SIGSAC} Conference on Computer and Communications Security, pp. 90-102, Nov. 2014

[16] Sun, He, et al., "Trustice: Hardware-assisted isolated computing environments on mobile devices," 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, pp. 367-378, June 2015.

[17] Cho, Yeongpil, et al., "Hardware-assisted on-demand hypervisor activation for efficient security critical code execution on mobile devices," USENIX Annual Technical Conference, pp. 565-578, June 2016.

〈저자소개〉

한 승 균 (Seung-Kyun Han) 학생회원
2020년 2월: 한밭대학교 컴퓨터공학과 졸업
現 충남대학교 컴퓨터융합학부 석사과정
〈관심분야〉 시스템 보안, 신뢰 실행 환경

장 진 수 (JinSoo Jang) 종신회원
2007년 8월: 아주대학교 정보및컴퓨터공학부 졸업
2008년 2월~2012년 8월: 앰코테크놀러지코리아 전산팀
2014년 8월: KAIST 정보보호대학원 석사
2018년 2월: KAIST 정보보호대학원 박사
現 충남대학교 컴퓨터융합학부 조교수
〈관심분야〉 시스템 보안, 신뢰 실행 환경