(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2022-0103253
(43) 공개일자 2022년07월22일

(51) 국제특허분류(Int. Cl.)
G06F 21/53 (2013.01) *G06F 21/57* (2013.01)
(52) CPC특허분류
G06F 21/53 (2013.01)
G06F 21/57 (2013.01)
(21) 출원번호 10-2021-0005601
(22) 출원일자 2021년01월14일
심사청구일자 2021년01월14일

(71) 출원인
충남대학교산학협력단
대전광역시 유성구 대학로 99 (궁동, 충남대학교)
한국과학기술원
대전광역시 유성구 대학로 291(구성동)
(72) 발명자
장진수
대전광역시 유성구 어은로 57, 121동 1206호 (어은동, 한빛아파트)
강병훈
대전광역시 유성구 배울1로 119, 1211동 901호 (용산동, 대덕테크노밸리12단지아파트)
(74) 대리인
이은철, 김재문

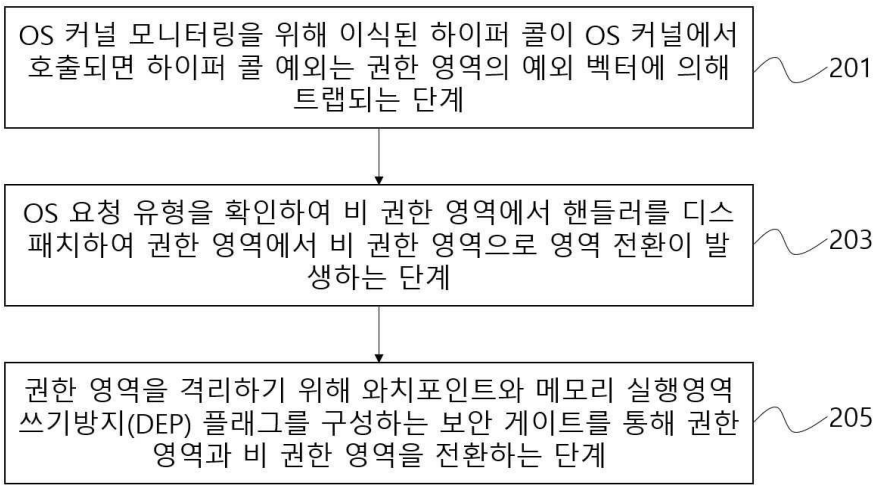
전체 청구항 수 : 총 7 항

(54) 발명의 명칭 **모바일 신뢰 실행 환경의 보안성 강화를 위한 장치**

(57) 요 약

본 발명은 모바일 신뢰 실행 환경의 보안성 강화를 위한 장치에 관한 것으로, 범용으로 신뢰 실행환경을 구축하기 위한 모바일 신뢰 실행 환경의 보안성 강화를 위한 장치에 관한 것이다. 본 발명의 일 실시예에 따르면, ARM 아키텍처를 기반으로 동작하는 모바일 기기에서 범용으로 활용 가능한 기술로써, 기존 상용 보안 기술에 대한 의존없이 어플리케이션의 안전한 실행을 보장하기 위한 신뢰 실행 환경을 구성하는 효과가 있고, 범용 하드웨어 기능인 디버깅 와치포인트와 쓰기영역 실행 방지 기능을 활용하여 모바일 신뢰 실행 환경을 구성하는 효과가 있다.

대 표 도 - 도2



이 발명을 지원한 국가연구개발사업

과제고유번호	1711103337
과제번호	2019-0-01343-002
부처명	과학기술정보통신부
과제관리(전문)기관명	정보통신기획평가원
연구사업명	정보통신방송혁신인재양성
연구과제명	융합보안핵심인재양성사업
기 여 율	1/1
과제수행기관명	한국인터넷진흥원
연구기간	2020.01.01 ~ 2020.12.31

명 세 서

청구범위

청구항 1

권한 영역과 비 권한 영역으로 분할하는 하이퍼 바이저;

와치포인트와 메모리 실행영역 쓰기방지를 제어하여 상기 권한 영역과 비 권한 영역을 스위치하는 영역 스위치부; 및

벡터의 무결성을 보호하기 위해 예외 발생 권한에 따라 분기에 의한 트랩으로 OS 커널모드와 하이퍼 바이저 모드로 스위칭하는 모드 스위치부;를 포함하는 것을 특징으로 하는 모바일 신뢰 실행 환경의 보안성 강화를 위한 장치.

청구항 2

제1항에 있어서,

상기 권한 영역은 페이지 테이블과 같은 쓰기 가능한 개체는 권한있는 데이터로 보호하며, 예외 벡터와 권한 지역코드를 페이지 단위에 따라 구분하는 것을 특징으로 하는 모바일 신뢰 실행 환경의 보안성 강화를 위한 장치.

청구항 3

제1항에 있어서,

상기 비 권한 영역은 모니터링되는 OS 커널에 대한 페이지 테이블 및 시스템 레지스터의 업데이트를 에뮬레이션하는 것을 특징으로 하는 모바일 신뢰 실행 환경의 보안성 강화를 위한 장치.

청구항 4

제1항에 있어서,

상기 영역 스위치부는 예외 벡터에서 OS 요청 유형을 확인하고 권한 영역을 격리하기 위해 와치포인트와 메모리 실행영역 쓰기방지(DEP) 플래그를 구성하는 보안 게이트를 통해 영역을 전환하는 것을 특징으로 하는 모바일 신뢰 실행 환경의 보안성 강화를 위한 장치.

청구항 5

제1항에 있어서,

상기 영역 스위치부는 비 권한 영역을 실행하는 동안 하이퍼 바이저 페이지 테이블 업데이트 요청시 예외 벡터에 의해 포착되는 하이퍼 콜을 이용하여 권한 영역으로 전환하는 것을 특징으로 하는 모바일 신뢰 실행 환경의 보안성 강화를 위한 장치.

청구항 6

제1항에 있어서,

상기 모드 스위치부는 OS 커널에서 트리거된 하이퍼 호출은 하위 권한의 예외로 인해 분기에 의해 트랩되고, 하이퍼 바이저 모드 실행에서 발생하는 모든 예외는 현재 권한에 대해 분기에 의해 트랩되는 것을 특징으로 하는 모바일 신뢰 실행 환경의 보안성 강화를 위한 장치.

청구항 7

모바일 신뢰 실행 환경의 보안성 강화를 위한 방법에 있어서,

OS 커널 모니터링을 위해 이식된 하이퍼 콜이 OS 커널에서 호출되면 하이퍼 콜 예외는 권한 영역의 예외 벡터에 의해 트랩되는 단계;

상기 OS 요청 유형을 확인하여 비 권한 영역에서 핸들러를 디스패치하여 권한 영역에서 비 권한 영역으로 영역 전환이 발생하는 단계;

상기 권한 영역을 격리하기 위해 와치포인트와 메모리 실행영역 쓰기방지(DEP) 플래그를 구성하는 보안 게이트를 통해 권한 영역과 비 권한 영역을 전환하는 단계;를 포함하는 것을 특징으로 하는 모바일 신뢰 실행 환경의 보안성 강화를 위한 방법.

발명의 설명

기 술 분 야

[0001] 본 발명은 모바일 신뢰 실행 환경의 보안성 강화를 위한 장치에 관한 것으로, 보다 상세하게는 범용으로 신뢰 실행환경을 구축하기 위한 모바일 신뢰 실행 환경의 보안성 강화를 위한 장치에 관한 것이다.

배 경 기 술

[0002] IoT(Internet of Things, 사물인터넷) 기술의 발달과 함께 보안에 대한 위협 또한 함께 증가하고 있다. IoT란 스마트폰, 홈 네트워크, 자동차, 각종 IT장치에 통신 기능을 내장하여 무선으로 인터넷에 연결하는 기술을 말하는데, 여기서 말하는 IT장치가 인터넷을 통해 외부에 노출됨에 따라, 결국 모든 사물이 해킹의 대상이 될 가능성이 있다.

[0003] 이처럼 늘어나는 보안 위협에 대비하여 선보인 기술 중 하나가 ARM의 Trustzone이다. 이러한 ARM Trustzone은 하나의 장치에 분리된 두 개의 환경을 제공하여 보안이 필요한 정보를 격리된 환경에서 안전하게 보호하는 기술이다.

[0004] 종래, ARM Trustzone이라는 상용 보안 기술을 활용하였으나, 이러한 기술이 제공되지 않는 저사양 기기에서는 신뢰 실행 환경을 구축하기 어려운 문제가 있다.

선행기술문헌

특허문헌

[0005] (특허문헌 0001) (특허문헌)등록특허 10-1324693호(2013.10.28.)

발명의 내용

해결하려는 과제

[0006] 본 발명은 상술한 문제를 해결하고자 고안한 것으로, 기존 상용 보안 기술에 대한 의존없이 어플리케이션의 안전한 실행을 보장하기 위한 신뢰 실행 환경을 구성가능하게 하는 모바일 신뢰 실행 환경의 보안성 강화를 위한 장치를 제공함에 목적이 있다.

[0007] 또한, 범용 하드웨어 기능인 디버깅 와치포인트와 쓰기영역 실행 방지 기능을 활용하여 모바일 신뢰 실행 환경을 구성하는 모바일 신뢰 실행 환경의 보안성 강화를 위한 장치를 제공함에 목적이 있다.

과제의 해결 수단

[0008] 본 발명의 일 측면에 따른 모바일 신뢰 실행 환경의 보안성 강화를 위한 장치는 권한 영역과 비 권한 영역으로 분할하는 하이퍼 바이저; 와치포인트와 메모리 실행영역 쓰기방지를 제어하여 상기 권한 영역과 비 권한 영역을 스위치하는 영역 스위치부; 및 벡터의 무결성을 보호하기 위해 예외 발생 권한에 따라 분기에 의한 트랩으로 OS 커널모드와 하이퍼 바이저 모드로 스위칭하는 모드 스위치부;를 포함한다.

[0009] 한편, 모바일 신뢰 실행 환경의 보안성 강화를 위한 방법에 있어서, OS 커널 모니터링을 위해 이식된 하이퍼 콜이 OS 커널에서 호출되면 하이퍼 콜 예외는 권한 영역의 예외 벡터에 의해 트랩되는 단계; 상기 OS 요청 유형을 확인하여 비 권한 영역에서 핸들러를 디스패치하여 권한 영역에서 비 권한 영역으로 영역 전환이 발생하는 단계; 상기 권한 영역을 격리하기 위해 와치포인트와 메모리 실행영역 쓰기방지(DEP) 플래그를 구성하는 보안

게이트를 통해 권한 영역과 비 권한 영역을 전환하는 단계;를 포함한다.

발명의 효과

[0010] 본 발명의 일 실시예에 따르면, ARM 아키텍처를 기반으로 동작하는 모바일 기기에서 범용으로 활용 가능한 기술로써, 기존 상용 보안 기술에 대한 의존없이 어플리케이션의 안전한 실행을 보장하기 위한 신뢰 실행 환경을 구성하는 효과가 있고, 범용 하드웨어 기능인 디버깅 와치포인트와 쓰기영역 실행 방지 기능을 활용하여 모바일 신뢰 실행 환경을 구성하는 효과가 있다.

도면의 간단한 설명

[0011] 도 1은 본 발명의 일 실시예에 따른 모바일 신뢰 실행 환경의 보안성 강화를 위한 장치를 간략히 나타낸 도면이다

도 2는 본 발명의 일 실시예에 따른 모바일 신뢰 실행 환경의 보안성 강화를 위한 방법을 나타낸 흐름도이다.

도 3은 자기 보호 접근 방식을 설명하기 위한 도면이다.

도 4는 본 발명의 일 실시예에 따른 모바일 신뢰 실행 환경의 보안성 강화를 위한 장치의 지역 내 권한 분리 지원을 설명하기 위한 도면이다.

도 5은 SelMon을 사용한 모드 및 지역 스위치를 나타낸 도면이다.

도 6는 원본 및 SelMon 강화 하이퍼 바이저가 있는 SPEC CPU2006을 나타낸 도면이다.

도 7는 원본 및 SelMon 강화 하이퍼 바이저와 함께 Phoronix Test Suite를 실행하여 평가한 OS 성능을 나타낸 도면이다.

발명을 실시하기 위한 구체적인 내용

[0012] 본 발명의 실시예에서 제시되는 특정한 구조 내지 기능적 설명들은 단지 본 발명의 개념에 따른 실시예를 설명하기 위한 목적으로 예시된 것으로, 본 발명의 개념에 따른 실시예들은 다양한 형태로 실시될 수 있다. 또한, 본 명세서에 설명된 실시예들에 한정되는 것으로 해석되어서는 아니 되며, 본 발명의 사상 및 기술 범위에 포함되는 모든 변경물, 균등물 내지 대체물을 포함하는 것으로 이해되어야 한다.

[0013] 한편, 본 발명에서 제1 및/또는 제2 등의 용어는 다양한 구성요소들을 설명하는데 사용될 수 있지만, 상기 구성요소들은 상기 용어들에 한정되지는 않는다. 상기 용어들은 하나의 구성요소를 다른 구성요소들과 구별하는 목적으로만, 예컨대 본 발명의 개념에 따른 권리 범위로부터 벗어나지 않는 범위 내에서, 제1 구성요소는 제2 구성요소로 명명될 수 있고, 유사하게 제2 구성요소는 제1 구성요소로도 명명될 수 있다.

[0014] 이하 첨부된 도면을 참조하여 본 발명의 실시예를 설명한다. 본 발명의 실시예를 설명함에 있어서, 관련된 공지 기능 혹은 구성에 대한 설명이 본 발명의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우 그 설명을 생략하였다.

[0015] 도 1은 본 발명의 일 실시예에 따른 모바일 신뢰 실행 환경의 보안성 강화를 위한 장치를 간략히 나타낸 도면이다. 도 1에 도시된 바와 같이, 본 발명의 일 실시예에 따른 모바일 신뢰 실행 환경의 보안성 강화를 위한 장치는 하이퍼 바이저(100), 영역 스위치부(200), 모드 스위치부(300)를 포함한다.

[0016] 하이퍼 바이저(100)는 권한 영역과 비 권한 영역으로 분할하여 구성된다.

[0017] 여기서, 권한 영역은 페이지 테이블과 같은 쓰기 가능한 개체는 권한있는 데이터로 보호하며, 예외 벡터와 권한 지역코드를 페이지 단위에 따라 구분한다. 비 권한 영역은 모니터링되는 OS 커널에 대한 페이지 테이블 및 시스템 레지스터의 업데이트를 에뮬레이션한다.

[0018] 영역 스위치부(200)는 와치포인트와 메모리 실행영역 쓰기방지를 제어하여 상기 권한 영역과 비 권한 영역을 스위치하는 구성이다. 이러한 영역 스위치부는 예외 벡터에서 OS 요청 유형을 확인하고 권한 영역을 격리하기 위해 와치포인트와 메모리 실행영역 쓰기방지(DEP) 플래그를 구성하는 보안 게이트를 통해 영역을 전환한다. 또한 영역 스위치부는 비 권한 영역을 실행하는 동안 하이퍼 바이저 페이지 테이블 업데이트 요청시 예외 벡터에 의해 포착되는 하이퍼 콜을 이용하여 권한 영역으로 전환한다.

[0019] 모드 스위치부(300)는 벡터의 무결성을 보호하기 위해 예외 발생 권한에 따라 분기에 의한 트랩으로 OS 커널모

드와 하이퍼 바이저 모드로 스위칭하는 구성이다. 이러한 모드 스위치부는 OS 커널에서 트리거된 하이퍼 호출은 하위 권한의 예외로 인해 분기에 의해 트랩되고, 하이퍼 바이저 모드 실행에서 발생하는 모든 예외는 현재 권한에 대해 분기에 의해 트랩된다.

[0020] 한편, 본 발명의 일 실시예에 따른 모바일 신뢰 실행 환경의 보안성 강화를 위한 방법은 도 2에 도시된 바와 같이, OS 커널 모니터링을 위해 이식된 하이퍼 콜이 OS 커널에서 호출되면 하이퍼 콜 예외는 권한 영역의 예외 벡터에 의해 트랩되는 단계(201), OS 요청 유형을 확인하여 비 권한 영역에서 핸들러를 디스패치하여 권한 영역에서 비 권한 영역으로 영역 전환이 발생하는 단계(203), 권한 영역을 격리하기 위해 와치포인트와 메모리 실행영역 쓰기방지(DEP) 플래그를 구성하는 보안 게이트를 통해 권한 영역과 비 권한 영역을 전환하는 단계(205)를 포함한다.

[0021] 이하, 본 발명의 일 실시예에 따른 모바일 신뢰 실행 환경의 보안성 강화를 위한 장치에 대해 구체적으로 설명하기로 한다.

[0022] 본 발명의 일 실시예에 따르면, Thin Hypervisor 및 Trust-Zone과 같은 더 높은 권한의 신뢰 앵커가 모바일 OS를 보호하기 위해 채택되었다. 예를 들어 Samsung Knox 보안 플랫폼은 64 비트 장치를 위한 하드웨어 지원 가상화 기술을 기반으로 커널 무결성 모니터를 구현한다. OS 커널 무결성을 보호하지만 악용 가능한 버그를 포함하는 경우, 모니터링 플랫폼 자체가 공격자의 표적이 될 수 있다. 이 발명에서 우리는 더 높은 권한을 가진 신뢰 앵커를 도입하지 않고 커널 무결성 모니터를 자기 보호하는 휴대용 방법인 "SelMon"을 제안한다. 이를 위해 먼저 무결성 모니터 영역을 논리적으로 권한 및 비권한 영역으로 두 부분으로 분리한다.

[0023] 그런 다음 권한이 있는 지역 코드만 모니터 무결성을 손상시키는데 악용될 수 있는 중요한 데이터 개체(예 : 하이퍼 바이저 페이지 테이블)에 액세스 할 수 있도록 한다. 모니터 무결성 유지 측면에서 중요하지 않은 작업은 권한이 없는 영역에서 수행된다. 권한 분리 외에도 일반 하드웨어 기능, 와치포인트(watchpoint) 및 데이터 실행 방지 (DEP)를 활용하여 분리의 견고성을 보장하는 방법도 설명한다.

[0024] 먼저, 본 발명의 일 실시예에 따른 모바일 신뢰 실행 환경의 보안성 강화를 위한 장치의 기술을 설명하면 다음과 같다. 이러한 기술은 모바일 프로세서 기술의 발전으로 인해 보안 애플리케이션 개발을위한 다양한 옵션이 도입되었다. 특히 하드웨어 지원 하이퍼 바이저 및 TrustZone과 같은 권한있는 하드웨어 기능을 지원하면 신뢰 앵커 기반 보안 시설을 구축 할 수 있다. 예를 들어, 산업계와 학계에서는 32 비트 모바일 장치에서 모바일 OS 커널 무결성을 보호하기 위해 TrustZone을 사용했다. 64 비트 아키텍처에서 Samsung Knox 플랫폼은 하드웨어 지원 하이퍼 바이저를 신뢰 앵커로 활용하여 커널 무결성을 보장한다. 수억 개의 제조된 모바일 장치가 이러한 보안 아티팩트와 함께 실행되고 있다.

[0025] 불행히도 연구자들은 신뢰 앵커 자체가 공격으로부터 자유롭지 않다는 것을 증명했다(2017.Full TrustZone exploit for MSM8974. http://bits-please.blogspot.kr/2015/08/full-trustzone-exploit-for-msm8974.html). 하이퍼 바이저 또는 TrustZone 사용 여부에 관계없이 커널 무결성 보호는 OS가 박탈되고 OS의 보안 중요 작업(예 : 시스템 레지스터 및 페이지 테이블 업데이트)이 트랩, 확인되는 유사한 방식으로 수행된다. 신뢰 앵커에 의해 에뮬레이트된다. 이러한 OS 커널 작업 위임은 공격자가 신뢰 앵커의 취약성을 찾아 악용하기 위해 악용하는 OS와 신뢰 앵커 간의 연속적인 상호 작용을 위해 일부 채널을 생성한다.

[0026] 이 문제를 해결하기 위해 본 발명은 신뢰 앵커 보호를 위해 다른 상위 권한 구성 요소가 필요하지 않은 SelMon이라는 신뢰 앵커 자체 보호 메커니즘을 제안한다. SelMon을 통해 하이퍼 바이저 기반 커널 무결성 모니터가 향상되어 접근 방식의 타당성을 보여준다. 모니터 권한의 가상 분리와 일반적인 하드웨어 기능의 활용은 SelMon의 핵심이다. 모니터는 논리적으로 권한 영역과 비 권한 영역으로 구분된다. 비 권한 영역은 커널 시스템 레지스터 업데이트와 같은 OS 커널 모니터링을 위한 중요하지 않은 기본 작업을 수행한다. 반면에 권한 영역은 모니터 보안과 밀접하게 관련된 사전 정의된 중요 작업을 관리한다. 예를 들어, 페이지 테이블을 남용하여 모니터를 조작할 수 있다. 따라서 페이지 테이블 업데이트는 권한 영역에서 제한적으로 수행되는 보안에 중요한 작업 중 하나로 정의된다.

[0027] 적절한 보호를 위해 권한이 없는 영역이 활성화된 동안에는 권한있는 영역에 액세스하거나 실행할 수 없어야 한다. 비 접근성 적용을 위해 하드웨어 디버깅 기능인 와치 포인트(watchpoint)를 사용한다. 특히, 권한이 없는 영역에 대한 항목에서 권한이 없는 전체 코드가 악의적으로 수정할 수 없도록 전체 권한 영역에 대한 와치포인트(watchpoint) 모니터링을 활성화한다. 권한 지역 코드의 임의 실행을 방지하기 위해 데이터 실행 방지 (DEP)에 대한 하드웨어 지원을 이용한다. 특히 권한이있는 코드를 쓰기 가능한 페이지에 매핑하고 권한이 없는 영역

으로 전환할 때 DEP를 활성화한다. 와치 포인트 구성에 의해 보장되는 비 접근성으로 인해이 영역은 비 권한 코드 실행 중에 변경 불가능한 상태로 유지된다. 마지막으로 지역 간 스위치는 결정적인 방식으로 관리되어야 한다. 이를 위해 앞서 언급한 지역 스위치를 기반으로 한 구성을 처리하는 각 지역에 대한 진입 게이트를 설계했다.

[0028] SKEE [Ahmed M. Azab, Kirk Swidowski, Rohan Bhutkar, Jia Ma, Wenbo Shen, Ruowen Wang, and Peng Ning. 2016. SKEE: A lightweight Secure Kernel-level Execution Environment for ARM. In 23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016.] 및 중첩 커널 [Nathan Dautenhahn, Theodoros Kasampalis, Will Dietz, John Criswell, and Vikram Adve. 2015. Nested kernel: An operating system architecture for intrakernel privilege separation. In ACM SIGPLAN Notices, Vol. 50. ACM, 191??206.]과 같은 이전 작업에서도 동일한 권한으로 영역을 분할하는 방법(즉, 영역 내 권한 분리)을 도입했다. 중첩 커널은 x86의 CR0 레지스터에서 쓰기 보호 (WP) 비트를 사용하여 분리를 실현한다. WP 비트는 쓰기 금지된 개체를 업데이트 할 수 있도록 권한있는 부분이 실행될 때만 비활성화된다. SKEE는 가상화 구성을 위해 시스템 레지스터를 사용한다.

[0029] 예를 들어 64 비트 ARM 아키텍처에서 TTBR (변환 테이블 기본 레지스터)은 비 권한 영역이 활성화될 때 권한 영역을 매핑하지 않는 새 테이블로 업데이트된다. 안타깝게도 이러한 접근 방식은 64 비트 ARM 아키텍처에서 실행되는 권한있는 신뢰 앵커를 보호하는 데 효과적이지 않다. 쓰기 금지 (CR0.WP)를 해제하는 시스템 전체 구성은 ARM에서 지원되지 않고, 또한 SKEE에는 하이퍼 바이저 (ARM 용어의 EL2) 및 보안 커널 (TrustZone)과 같은 권한 모드에서는 사용할 수 없는 보조 페이징 지원이 필요하다.

[0030] Cortex-A57 및 Cortex-A53 멀티 코어 프로세서와 8GB DRAM을 제공하는 ARM Juno 개발 기판에 SelMon을 구현했다. OS 커널 무결성 보호를 수행하는 씬 하이퍼 바이저(Thin Hypervisor)는 SelMon으로 강화되어 접근 방식의 타당성을 보여준다. 하드웨어 지원 DEP와 4 개의 디버그 와치 포인트 중 하나는 권한이 없는 영역에서 권한을 분리하는 데 사용된다. SelMon은 지역 전환이 발생할 때마다 수행되는 하이퍼 바이저와 하드웨어 기능 구성의 권한 분리로 인해 하이퍼 바이저 성능을 저하시킨다.

[0031] 그러나 원래 및 SelMon 강화 하이퍼 바이저에서 실행되는 Linux의 성능을 측정하고 비교한 Phoronix 벤치 마크 결과는 SelMon의 성능 영향이 제한적임을 나타낸다. 이때, 5 %의 최대 오버 헤드가 관찰되었다.

[0032] 본 발명에 따르면 다음과 같이 요약될 수 있다.

[0033] 고급 모바일 장치에서 실행되는 신뢰 앵커 (권한 소프트웨어) 보호를 목표로하는 SelMon을 소개하면, 우리의 접근 방식에는 와치포인트(watchpoint) 및 데이터 실행 방지(DEP)와 같은 일반적인 하드웨어 기능 외에 더 높은 권한을 가진 소프트웨어 또는 하드웨어 구성 요소가 필요하지 않다.

[0034] SelMon은 확장 가능하다. 와치포인트(watchpoint)와 데이터 실행 방지(DEP)가 커널 모드에서 지원되기 때문에 SelMon은 트러스트 앵커를 사용할 수 없을 때 OS를 자체 보호하도록 조정할 수 있다. 또한 유사한 하드웨어 기능을 지원하는 다른 아키텍처는 SelMon의 접근 방식을 활용할 수 있다.

[0035] -ARM 프로세서 보안 상태-

[0036] ARM 보안 확장[2018. ARM Architecture Reference Manual ARMv8, for ARMv8-A architecture profile]을 사용하면 프로세서 보안 상태를 보안 상태와 비보안 상태로 나눌 수 있다. 보안 상태에서는 사용자, 커널 및 모니터 모드를 사용할 수 있다. 비보안 상태에서는 사용자, 커널 및 하이퍼 바이저 모드가 지원된다. 보안 상태는 보안 구성 레지스터 (SCR_EL3)의 비보안 (NS) 플래그를 사용하여 구성된다. 예를 들어 NS 플래그가 설정된 경우 현재 보안 상태는 비 보안이다. 구성은 보안 상태와 비 보안 상태 사이의 스위치를 처리하기 위해 도입된 모니터 모드에서 수행된다. 일반적으로 이 프로세서 보안 상태 분리는 중요한 메모리 영역 및 주변 장치를 격리하는 TrustZone 기술 [2018. ARM Security Technology - Building a Secure System using TrustZone Technology]과 함께 활용되어 TEE(신뢰할 수있는 실행 환경)를 만들 수 있다.

[0037] -ARM 하드웨어 지원 가상화-

[0038] ARMv7 [2008. Architecture Reference Manual (ARMv7-A and ARMv7-R edition). ARM DDI C 406 (2008).]부터 하드웨어 지원 가상화가 지원된다.

[0039] 보안 애플리케이션 구축 측면에서 가상화의 주요 기능은 (1) 보조 페이징 및 (2) 하이퍼 바이저 트랩의 두 가지로 분류할 수 있다. 2차 페이징은 기본적으로 중간 물리적 주소 (즉, 가상 머신보기의 물리적 주소)를 실제 머

신 주소로 변환하는 것을 목표로 한다. 이 기능은 주로 보조 페이지 테이블을 만들어 다른 영역에서 특정 메모리 영역을 격리하거나 보호하는 데 사용된다. 예를 들어 하이퍼 바이저 기반 커널 무결성 모니터는 텍스트를 매핑하는 보조 페이지의 읽기 전용 권한을 설정하여 커널 텍스트 영역을 보호한다.

[0040] 하이퍼 바이저 트랩 구성은 하이퍼 바이저에서 확인하거나 에뮬레이션해야 하는 보안에 중요한 작업을 쉽게 모니터링할 수 있도록 한다. OS 커널의 가상 메모리 구성은 보안 하이퍼 바이저에 의해 트랩되고 확인되는 일반적인 예제 작업이다.

[0041] -하드웨어 디버깅 지원-

[0042] 외부 하드웨어 (예 : JTAG 디버거)를 통한 디버깅 외에도 ARM 및 x86과 같은 상용 프로세서는 권한있는 소프트웨어가 디버깅 예외를 트랩하고 처리할 수 ????있도록 하는 하드웨어 디버깅 기능을 지원한다. 예를 들어, 특정 주소에 하드웨어 중단 점(break point)을 설정하면 주소에 대한 명령이 실행될 때 중단 점 예외가 생성될 수 있다. 예외는 OS 커널과 같은 권한이 있는 소프트웨어에 의해 트랩된다. 반면에 와치 포인트는 유사한 방식으로 데이터 액세스를 모니터링하는데 사용할 수 있다. 특히, 와치 포인트 모니터링 영역에 대한 데이터 액세스 (읽기 또는 쓰기)가 발생하면 와치 포인트 예외가 생성되고 권한있는 소프트웨어에서 트랩 및 처리된다. 본 실시예에서는 와치 포인트(watch point)를 사용하여 보안에 중요한 메모리 영역에 대한 액세스를 모니터링한다. 참고로, 브레이크 포인트(break-point)는 특정 라인에서부터 디버깅 시작이라는 식이라면, 와치포인트(watchpoint)는 특정 필트의 특정 값부터 디버깅 시작이라는 식이다. 와치 포인트의 다음 속성이 활용된다.

[0043] 표 1은 비 보안 및 보안 상태에서 와치포인트 예외 생성을 위한 예제 구성이고, 표 2는 디버그 예외 라우팅을 위한 제어 플래그 설정을 나타낸다.

[0044] (표 1)

SSC	PAC	Security state	Watchpoint for
01	10	Non-secure	User
01	01	Non-secure	Kernel
11	00	Non-secure	Hypervisor
10	10	Secure	User
10	01	Secure	Kernel

[0045]

[0046] (표 2)

	Privilege level where debug exception is generated		
	User	Kernel	Hypervisor
Settings	**Privilege level that handles the exception**		
NS=1, TDE=0	Kernel	Kernel	Hypervisor*
NS=1, TDE=1	Hypervisor	Hypervisor	Hypervisor
NS=0, TDE=×	Kernel	Kernel	N/A

* Breakpoint instruction exception only

[0047]

[0048] 와치포인트 예외 생성.

[0049] 와치포인트 제어 플래그를 구성하여 특정 프로세서 모드에서 생성되는 와치포인트 예외를 설정할 수 있다. 예를 들어, 표 1에서 볼 수 있듯이 디버그 와치포인트 제어 레지스터 (DBGWCR)의 보안 상태 제어 (SSC) 및 권한있는 액세스 제어 (PAC) 플래그를 0b11 및 0b00으로 설정하여 하이퍼 바이저 모드에서 예외를 생성 할 수 있다. SSC 값에 따라 와치 포인트 예외가 TEE (보안 상태)에서도 생성될 수 있다. SSC 값을 0b10으로 설정하면 TEE의 사용자 모드 또는 커널 모드에서 예외를 생성할 수 있다.

[0050] 와치포인트 예외 라우팅.

[0051] 표 2에 나와있는 것처럼 프로세서 모드가 비 보안 상태인 경우 와치포인트 예외는 모니터 디버그 구성 레지스터 (MDCR_EL2)의 TDE (트랩 디버그 예외) 플래그에 따라 커널 또는 하이퍼 바이저에서 처리될 수 있다. 예를 들어, 하이퍼 바이저 모드에서 와치포인트 예외를 트랩하도록 TDE 플래그를 설정해야 한다. 반면에 프로세서가 TEE

(보안 상태, 즉 NS = 0)에 있는 경우 예외는 항상 커널 모드 (보안 OS)에 의해 트랩되고 처리된다.

[0052] 와치포인트 구성 요구 사항.

[0053] ARM 64 비트 아키텍처 (ARMv8)에서 와치포인트 모니터링의 최대 크기는 2GB이다. 크기는 2의 거듭 제곱으로 정렬되어야 한다. DBGWCR에서 마스크 플래그를 사용하여 모니터링 크기를 구성 할 수 있다. 또한 모니터링 시작 주소는 모니터링 크기에 맞춰야 한다. 예를 들어, 모니터링 크기가 4KB 인 경우 모니터링 시작 주소는 4KB에 맞춰야 한다. 시작 주소는 디버그 워치 포인트 값 레지스터 (DBGWVR)에서 구성 할 수 있다.

[0054] 크기 및 주소 구성 외에도 와치포인트 모니터링을 활성화하려면 여러 제어 플래그를 설정해야 한다. 첫째, DBGWCR의 활성화 플래그를 설정해야 한다. 둘째, 커널 디버그 활성화 (KDE) 플래그가 예외를 처리하는 동일한 모드에서 와치포인트 예외를 생성하도록 설정되어야 한다. 예를 들어, 하이퍼 바이저 모드에서 와치포인트 예외를 자체 관리하려면 KDE 및 TDE 플래그를 설정해야 한다. 마지막으로 프로세스 상태 (PSTATE)의 디버그 예외 마스크 플래그 (D)를 지워야 한다. 이 플래그는 예외가 발생하면 자동으로 설정되어 디버그 예외 생성을 비활성화 한다.

[0055] -커널 무결성 모니터-

[0056] TrustZone 및 하이퍼 바이저와 같은 트러스트 앵커는 커널 무결성 모니터를 구현하는데 사용되었다. TZ-RKP [Ahmed M Azab, Peng Ning, Jitesh Shah, Quan Chen, Rohan Bhutkar, Guruprasad Ganesh, Jia Ma, and Wenbo Shen. 2014. Hypervision across worlds: real-time kernel protection from the ARM trustzone secure world. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. ACM, 90??102.] 및 Sprobes [Xinyang Ge, Hayawardh Vijayakumar, and Trent Jaeger. 2014. Sprobes: Enforcing Kernel Code Integrity on the TrustZone Architecture. Proceedings of the Third Workshop on Mobile Security Technologies (MoST) (2014)]은 32 비트 장치의 TrustZone 기반 TEE에서 무결성 모니터를 찾는다. 64 비트 ARM 아키텍처를 기반으로 하는 최신 모바일 장치에서 하드웨어 지원 가상화 모니터링을 활성화하기 위해 기술도 활용된다. 삼성은 커널 텍스트와 데이터를 보호하기 위해 씬 하이퍼 바이저를 사용한다.

[0057] 한편 무결성 모니터가 배포된 위치에 관계없이 모니터는 일반적으로 다음 보안 속성을 적용한다. 첫째, 페이지 테이블은 커널에 의해 업데이트될 수 없다. 보안 확인 후에는 무결성 모니터만 업데이트를 에뮬레이션 할 수 있다. 이것은 메모리 보호를 보장한다. 둘째, 커널은 보안에 중요한 시스템 레지스터를 구성할 수 없다. 메모리 보호와 유사하게 중요한 시스템 레지스터는 모니터에 의해서만 업데이트된다. 이는 악의적으로 시스템 설정을 재구성하는 것을 방해한다 (예 : 메모리 관리 장치 (MMU) 비활성화 또는 페이지 테이블 다시 매핑). 모니터가 동기적으로 호출된다. 이러한 업데이트를 수행해야 할 때마다 이를 위해 업데이트를 위한 원래 커널 작업이 하이퍼 콜과 같은 트러스트 앵커 호출로 대체된다.

[0058] -권한 분리-

[0059] 소프트웨어의 권한을 분리하고 중요한 작업을 안전하게 실행하기 위한 다양한 접근 방식이 연구되었다. SKEE [17] 및 Hilps [21]는 메모리 변환 관련 기능을 활용하여 OS 커널에서 격리된 실행 환경을 만든다. 시스템 전체 메모리 WP는 중첩 된 커널 [27], Nexen [41] 및 HyperSafe [43]에서 OS와 하이퍼 바이저의 중요한 작업을 격리하기 위해 활용되었다. 이러한 접근 방식은 지역 내 권한 분리를 가능하게하지만 설명된대로 ARM 아키텍처의 트러스트 앵커 보호에 제한적으로 채택 할 수 있다.

[0060] 반면에 중요한 작업을 모니터링하기 위한 추가 계층도 도입되었다. Xen의 반 가상화 [18]는 페이지 테이블 업데이트와 같은 OS 커널의 중요한 작업을 확인하고 에뮬레이션한다. SVA (Secure Virtual Architecture) [26]는 OS 동작을 모니터링하기 위한 가상 명령 기반 추상화 계층을 제공한다.

[0061] KCoFI [24]와 Virtual Ghost [25]는 OS의 제어 흐름을 보호하고 애플리케이션을 보호하기 위해 SVA를 채택한다. Apparition [28]은 Virtual Ghost를 강화하여 악성 OS의 부 채널 공격으로부터 보호 된 응용 프로그램을 보호한다. HypSec [34]는 모놀리식 하이퍼 바이저 (예 : KVM)를 재 설계하여 신뢰할 수 있는 소규모 코어 바이저를 신뢰할 수 없는 대규모 호스트 바이저로부터 분리한다. 격리는 보조 페이징을 사용하고 서로 다른 권한이 있는 CPU 모드 (각각 하이퍼 바이저 및 커널)에서 구획(코어 바이저 및 호스트 바이저)을 실행하여 적용된다.

[0062] 우리는 신뢰할 수 있는 추가 계층을 채택하지 않고 보안 아티팩트를 효과적으로 분리하여 계층을 사용할 수 없을 때 추가 옵션을 제공하는 다른 접근 방식을 제안한다.

[0063] -모바일 장치 보안-

[0064] 다양한 하드웨어 기능을 활용하여 모바일 장치용 보안 응용 프로그램을 구현했다. 콜드 부팅 공격을 방지하기 위해 Sentry [23]는 민감한 데이터와 코드를 DRAM 대신 ARM SoC에 저장한다. ARMlock [47]은 메모리 도메인과 도메인 액세스 제어 레지스터 (DACR)를 사용하여 하드웨어 기반 오류 격리를 가능하게 한다. Norax [20]은 ARMv8의 하드웨어 지원을 통해 XOM (실행 전용 메모리)을 활성화한다. ARM 프로세서에 대한 보안 확장으로서 TrustZone [7]은 장치 보안을 향상시키기 위해 광범위하게 탐색되었다. 예를 들어 TLR [37]과 ObC [33]는 TrustZone을 사용하여 타사 개발자에게 TEE를 제공한다. TrustOTP [42]는 TrustZone을 사용하여 일회용 암호 토큰을 보호한다. Komodo [29], TrustShadow [31] 및 CaSE [46]는 TrustZone에서 제공하는 격리를 기반으로 응용 프로그램의 일부를 보호한다. 마지막으로 vTZ [32]는 TrustZone을 가상화하여 개별 가상 머신에 TEE를 제공한다.

[0065] -공격 모델-

[0066] 커널 무결성을 모니터링하고 씬 하이퍼 바이저로 구축되는 신뢰 앵커가 있다고 가정한다. 모니터는 앞에서 커널 무결성 모니터에 설명된대로 OS 커널을 보호한다. 즉, OS 커널은 페이지 테이블 및 시스템 레지스터 업데이트와 같은 보안에 중요한 작업을 수행하는 권한을 박탈당한다. 대신 모니터가 이러한 작업의 확인 및 에뮬레이션을 담당한다.

[0067] 그러나 Google 연구원 [12]이 보여주는 것처럼 모니터 (하이퍼 바이저) 자체가 공격자에 의해 손상될 수 있다. 특히 모니터 자체는 공격자에게 임의의 메모리 액세스 및 제어 흐름 하이재킹 기능을 부여하는 취약성을 포함할 수 있다. 상승된 권한으로 공격자는 모니터를 조작하여 공격을 영속화할 수 있다. 이를 위해 페이지 테이블 및 시스템 구성 레지스터와 같은 중요한 하이퍼 바이저 구성 요소가 남용된다. 본 실시예의 접근 방식인 SelMon은 취약점이 있는 경우에도 하이퍼 바이저의 핵심 구성 요소의 무결성을 보호하는 방식으로 하이퍼 바이저를 강화하는 것을 목표로 한다. 보다 구체적인 분석 및 분류 보안에 중요한 구성 요소와 SelMon의 디자인은 아래에서 설명하기로 한다.

[0068] -동기-

[0069] 앞서 공격 모델에 설명된대로 무결성 모니터 자체가 취약할 수 있다. 모니터링 플랫폼 보안을 강화하기 위한 기본 단계로 하이퍼 바이저 및 보안 (및 비보안) 커널 모드에서 실행되는 신뢰 앵커를 자체 보호하는 것을 목표로 하는 SelMon을 제안한다. SelMon의 핵심 기술은 동일한 권한으로 모놀리식으로 실행되는 소프트웨어의 권한 분리를 실현하는 것이다. 특히, 하드웨어 와치포인트 및 DEP 지원의 이점을 누리는 SelMon은 더 높은 권한의 하드웨어 및 소프트웨어 구성 요소에 의존하지 않는다.

[0070] 이전의 자기 보호 접근 방식. Sel-Mon을 소개하기 전에, 본 실시예는 도 3에 도시된 바와 같이, 상위 권한 구성 요소에 의존하지 않고 소프트웨어 권한을 분리하는 중첩 커널 [27], SKEE [17] 및 Hilps [21]와 같은 기존 접근 방식을 분석한다. 그런 다음 이러한 접근 방식이 최신 ARM 아키텍처에서 신뢰 앵커 (권한 소프트웨어)를 보호하는 데 충분하지 않은 이유를 설명한다.

[0071] 세 가지 접근 방식은 먼저 페이지 테이블 업데이트와 같은 보안에 중요한 작업을 수행하는지 여부에 따라 시스템을 두 개의 구획, 즉 권한있는 영역과 비 권한 영역으로 구분한다. 권한 영역으로 들어가고 나가는 것을 처리하는 코드에서, 게이트는 특히 제안된 접근법의 견고성을 지배하는 시스템 구성을 관리한다. 따라서 ARM에서 권한이 부여된 소프트웨어를 보호하기 위해 게이트 설계가 일반적으로 적용되는지 확인하여 각 접근 방식의 실행 가능성을 분석한다.

[0072] (목록 1 : CR0에서 WP 비트를 구성하는 중첩 된 커널 항목에 대한 게이트 코드의 일부)

[0073] 1 entry :

[0074] 2 ......

[0075] 3 mov % cr0, %rax //Get current CR0 value (현재 CR0 값 가져 오기)

[0076] 4 and ~CR0_WP, %rax //Clear WP bit in copy (사본에서 WP 비트 지우기)

[0077] 5 mov %rax, %cr0 //Write back to CR0 (CR0에 다시 쓰기)

[0078] 6 cli //Disable interrupts (인터럽트 비활성화)

[0079] 7 ......

[0080] -중첩 커널-

[0081] 페이지 테이블과 같은 중요한 커널 개체를 보호하기 위해 중첩 커널은 모든 개체에 대해 읽기 전용 권한을 적용한다. 그런 다음 권한있는 부분이 개체를 업데이트할 수 있도록 중첩 커널은 CR0의 WP 비트를 사용하여 시스템 전체 읽기 전용 설정을 해제한다. 목록 1의 3-5 행에서 볼 수 있듯이 권한 영역에 대한 항목에서 쓰기 방지를 비활성화 한다(권한 영역에서 나갈 때 그 반대도 마찬가지 임). 불행히도 ARM 아키텍처는 이러한 제어 비트 구성 가능성(즉, WP)을 제공하지 않는다. 페이지 테이블 속성은 MMU가 활성화되어있는 한 항상 적용된다.

[0082] 목록 2 : 제로 레지스터 (xzr)를 사용하여 TTBR 값을 구성하는 SKEE 게이트 코드의 일부.

[0083] 1 ......

[0084] 2 msr DAIFset,0x3 // Mask all interrupts(모든 인터럽트 마스킹)

[0085] 3 mrs x0,ttbr1_el1 // Read existing TTBR1 value(기존 TTBR1 값 읽기)

[0086] 4 str x0,[sp, #-8]! // Save existing TTBR1 value(기존 TTBR1 값 저장)

[0087] 5 msr ttbr1_el1,xzr // Load Zero to TTBR1(TTBR1에 0로드)

[0088] 6 isb

[0089] 7 tlbi vmalle1 // Invalidate the TLB(TLB 무효화)

[0090] 8 isb

[0091] 9 ......

[0092] -SKEE-

[0093] ARM에서 WP 구성 지원이 없기 때문에 SKEE는 다른 접근 방식을 사용한다. 권한 영역과 비 권한 영역에 대해 두 개의 서로 다른 페이지 테이블을 준비한다. 권한 영역에 대한 페이지 테이블에만 해당 영역에 대한 매핑이 있으며, 이는 권한 영역에 대한 항목에서 활성화된다. 영역 전환 중에 페이지 테이블을 전환하는 이 접근 방식은 TTBR을 업데이트하는 명령을 실행하여 게이트를 악용하여 악성 페이지 테이블을 매핑 할 수 있다는 단점이 있다. 이 문제를 해결하려면, TTBR을 결정적으로 전환하는 SKEE는 상수 주소 (0x0)에서 권한 영역에 대한 페이지 테이블을 찾고 목록 2에서 볼 수 있듯이 TTBR 업데이트에서 제로 레지스터 (XZR)를 사용한다. 주소 (0x0)가 그렇지 않은 경우 가능한 경우, 본 실시예에서 가상화 기술 (즉, 2 차 페이징)을 사용하여 커널 관점에서 특정 메모리 영역을 0x0으로 다시 매핑 할 것을 권장한다. 기기 구현에 따라 주소 0x0은 하드웨어 주변기기 용으로 예약 될 수 있다. 예를 들어 Juno 개발 보드는 0x0 [10]에 Boot ROM을 배치하므로 Juno 보드에서 SKEE를 활성화하려면 메모리를 다시 매핑해야한다. 안타깝게도 추가 변환 계층 (즉, 보조 페이징)은 하이퍼 바이저 모드 및 보안 상태 (즉, TEE 커널)의 커널 모드에서 지원되지 않는다. 결과적으로 SKEE 접근 방식은 이러한 권한 모드에서 실행되는 소프트웨어의 보호를 위해 채택하기가 어렵다.

[0094] 목록 3 : TCR (변환 구성 레지스터)을 구성하는 Hilps 게이트 코드의 일부. 가상 주소 범위 및 페이지 크기에 대한 플래그의 유효성이 검사된다.

[0095] 1 ......

[0096] 2 1:

[0097] 3 mrs x5,tcr_el1 // Read the current TCR

[0098] 4 and x5,x5, # 0xffffffffffdfffff // Set T1SZ flags

[0099] 5 orr x5,x5, # 0x400000

[0100] 6 msr tcr_el1,x5 //Configure TCR

[0101] 7 isb

[0102] 8

[0103] 9 mov x6,# 0xc03f // Check the TCR setting

[0104] 10 mov x7,# 0x1b //(1) virtual address range

[0105] 11 movk x6,# 0xc07f,lsl #16 // and (2) trans. granule size

[0106] 12 movk x7,# 0x8059,lsl #16

[0107] 13 and x5,x5,x6

[0108] 14 cmp x5,x7

[0109] 15 b.ne 1b // If invalid, configure TCR again

[0110] 16 ......

[0111] -Hilps-

[0112] 목록 3에 나와있는 것처럼 Hilps는 TCR의 TxSZ 필드를 활용한다. 가상 주소의 범위는 TxSZ를 구성하여 조정할 수 있다. 가상 주소 범위를 동적으로 변경하여 특정 가상 주소 영역을 즉시 숨길 수 있다. 이를 사용하여 Hilps는 비 권한 영역 실행 중에 권한 영역을 숨긴다. 이 접근 방식은 공격자가 목록 3의 6 행에있는 TCR 업데이트 명령을 남용 할 때 잠재적으로 취약 할 수 있다. 이는 가상 주소 범위뿐 아니라 TGx (translation granule) 필드를 구성하여 번역 과립 크기를 조정할 수 있기 때문이다. TCR에서. 현재 4KB, 16KB 및 64KB 번역 그래뉼 크기는 ARM 64 비트 아키텍처. 세분화된 크기는 최소 페이지 크기뿐만 아니라 워킹 페이지 테이블의 인덱스 크기를 결정한다. 예를 들어, 4KB 그래뉼은 가상 주소의 9 비트 슬라이스를 인덱스로 사용하는 반면 16KB 그래뉼은 11 비트를 사용한다. 4KB 그래뉼은 1GB, 2MB 및 4KB 페이지 크기를 허용하지만 페이지 테이블 수준에 따라 16KB 그래뉼과 함께 32MB 및 16KB 페이지를 사용할 수 있다. 따라서 페이지 테이블 항목과 특정 범위의 가상 주소를 걷는 항목 수는 변환 단위 크기에 의해 결정된다. 즉, 특정 가상 주소가 다른 페이지 테이블 항목에 의해 변환될 수 있기 때문에 그래뉼 크기를 변경하면 비 결정적인 동작이 발생할 수 있다. 공격자는 이를 악용하여 임의의 의도하지 않은 명령을 실행할 수 있다. 예를 들어 목록 3에서 유효성 검사 논리 (9-14 행)가 이전 코드 (1-7 행)와 다른 4KB 크기 페이지의 경계에 있는 경우 과립 크기 구성을 16KB에서 4KB로 변경한다. (또는 그 반대의 경우) 시스템 상태 (예 : 캐시 상태 및 희소 물리적 메모리 할당)에 따라 비 결정적으로 유효성 검사를 우회할 수 있다.

[0113] -시스템 설계-

[0114] 시스템 설계를 설명하기 전에 (5.1)SelMon을 설명하고, (5.2)보안에 중요한 개체, (5.3)권한 분리, (5.4)권한 영역 보호, (5.5)지역 전환, (5.6)디버깅 활동과의 호환성 순으로 설명하기로 한다.

[0115] (5.1)SelMon은 모니터링 플랫폼 자체 보호 측면에서 운영 중요도에 따라 씬 하이퍼 바이저 (무결성 모니터)를 권한있는 영역과 비 권한 영역으로 분할한다. 아래 보안에 중요한 개체에 설명된 대로 하이퍼 바이저 예외 벡터, 페이지 테이블 및 시스템 제어 레지스터를 권한 영역에서 격리되는 보안에 중요한 객체로 정의한다. 반면에 권한이 없는 영역은 OS 커널 무결성 보호를 위한 기본 작업을 수행한다. 이 영역은 취약할 수 있으므로 공격자가 악용할 수 있다고 가정한다. 그러나 비 권한 영역에서 트리거된 공격은 비 권한 영역에서 효과적으로 격리되므로 하이퍼 바이저는 보존된다. SelMon이 다음 조건을 만족하기 때문이다.

[0116] 첫째, 비 권한 영역은 하이퍼 바이저 관리 페이지 테이블에 액세스할 수 없다. 따라서 직접 메모리 조작 시도가 방해를 받는다. 둘째, 보안에 중요한 명령이 권한이 없는 영역에 존재하지 않는다. 따라서 공격자는 예외 벡터 주소와 같은 시스템 구성을 변경할 수 없다. 마지막으로, 공격자는 권한이 없는 영역이 활성화되는 동안 전체 권한이 있는 영역이 실행 불가능하도록 강제되기 때문에 권한이 있는 영역에서 중요 명령을 재사용할 수 없다. 지역 간 스위치는 미리 정의된 방법을 통해 관리된다. 본 실시예에서는 권한 영역의 보안을 보장하는 방식으로 각 영역에 대한 항목을 관리하는 보안 게이트를 설계했다.

[0117] (5.2)보안에 중요한 개체

[0118] 시스템 설계에 있어 보안에 중요한 개체를 설명하면 다음과 같다.

[0119] 공격자가 악용할 수 있는 중요 개체는 적절하게 확인하고 격리해야 한다. 하이퍼 바이저 예외 벡터, 페이지 테이블 및 하이퍼 바이저 관련 시스템 레지스터는 보호해야 하는 중요한 구성 요소이다. 이러한 작업에서 본 실시예에 따르면, 씬 하이퍼 바이저가 프로덕션 모바일 장치에 배포되어 있으며 그 작업은 가볍고 보안에 전념한다

고 생각한다. 그러나 일반적으로 분류된 구성 요소는 OS를 포함한 권한있는 소프트웨어를 구축하기 위한 공통적이고 전제 조건이다.

[0120] 핵심. 따라서 OS 커널 및 번들 하이퍼 바이저 (예 : KVM)와 같은 더 복잡한 소프트웨어를 보호하기 위해 분석을 확장할 수 있다.

[0121] 도 4는 본 발명의 일 실시예에 따른 모바일 신뢰 실행 환경의 보안성 강화를 위한 장치의 지역 내 권한 분리 지원을 설명하기 위한 도면이다. 도 4에 도시된 바와 같이, SelMon은 디버깅 와치포인트 및 DEP와 같은 일반 하드웨어 기능을 사용하여 지역 내 권한 분리를 지원한다. 권한 영역의 엄격한 격리 및 보호를 위해 지역 간 스위치는 하드웨어 기능을 적시에 구성하는 미리 정의된 좁은 인터페이스에 의해 수행된다.

[0122] 시스템 설계에 있어 보안에 중요한 개체를 나열하면 다음과 같다. 페이지 테이블, 하이퍼 바이저 예외 백터, 시스템 제어 레지스터가 있다.

[0123] (5.2.1) 페이지 테이블은 하이퍼 바이저가 관리하는 두 가지 유형의 페이지 테이블 (하이퍼 바이저 및 보조 페이지 테이블)이 있다. 유형에 관계없이 테이블을 보호해야하는 중요한 개체로 간주한다. 먼저, 하이퍼 바이저 모드에 대한 페이지 테이블이다. 가상화 확장은 하이퍼 바이저 모드에서 가상 주소를 지원한다. MMU를 활성화하려면 하이퍼 바이저가 페이지 테이블과 페이지 테이블 기본 레지스터를 구성해야한다. 페이지 테이블을 구성하여 하이퍼 바이저 코드 및 데이터를 읽기 전용으로 설정할 수 있다. DEP와 같은 보안 기능도 활성화 할 수 있다. 그러나 공격자가 자신의 권한을 하이퍼 바이저로 에스컬레이션하면 보호 설정이 더 이상 유효하지 않다. 예를 들어, 공격자는 하이퍼 바이저 텍스트 페이지를 쓰기 가능하게 만들고 코드를 자유롭게 수정할 수 있다 [12]. 보조 페이지 테이블. 앞서 커널 무결성 모니터 설명에서 언급했듯이 모니터 (씬 하이퍼 바이저)는 OS 커널 텍스트와 코드가 불변 (읽기 전용)이 되도록 보조 페이지 테이블을 관리한다. 그러나 이 페이지 테이블은 하이퍼 바이저 무결성을 손상시키기 위해 악용될 수도 있다. 공격자는 하이퍼 바이저 메모리 영역이 OS 커널에 매핑되도록 보조 페이지 테이블 항목을 조작할 수 있다. 그렇게 하면 더 낮은 권한을 가진 커널에서도 하이퍼 바이저가 악의적으로 수정된다.

[0124] (5.2.2) 하이퍼 바이저 예외 벡터. 예외 벡터는 현재 예외에 해당하는 핸들러 루틴을 전달하는 코드이다. 모든 예외 발생은 프로그램 카운터를 예외 벡터의 미리 정의된 위치로 변경한다. 하이퍼 바이저 예외 벡터는 예외 발생 권한 (현재 또는 하위)에 따라 다른 분기를 갖는다. 예를 들어 OS 커널에서 트리거된 하이퍼 호출은 하위 권한의 예외로 인해 분기에 의해 트랩된다. 그러나 하이퍼 바이저 모드 실행에서 발생하는 모든 예외는 현재 (하이퍼 바이저) 권한에 대해 분기에 의해 트랩된다. 예외 벡터는 코드 디스패처 역할을 하므로 공격자가 제어 흐름을 악성 코드로 리디렉션하지 못하도록 벡터의 무결성을 보호해야 한다.

[0125] (5.2.3) 시스템 제어 레지스터. 시스템 제어 레지스터는 공격의 대상이 될 수 있다. 예를 들어 TTBR을 수정하여 악성 페이지 테이블을 매핑할 수 있다. 벡터 기본 주소 레지스터 (VBAR)는 현재 예외 벡터를 악의적인 벡터로 대체하기 위해 남용된다. 이전 작업 [16, 30]은 공격자가 시스템 레지스터를 악용하는 것을 방지하기 위해 OS의 코드 영역에서 모든 보안에 중요한 명령을 제거했다. SelMon은 유사한 접근 방식을 적용한다. 중요한 명령은 권한 영역에만 존재한다. 또한 공격자는 제어 흐름을 권한있는 영역의 임의 위치로 리디렉션할 수 없다.

[0126] (5.3)권한 분리

[0127] 다음으로 하이퍼 바이저 권한의 논리적 분리에 대해 설명하기로 한다. 권한 분리는 앞서 기술한 보안에 중요한 개체에서 분류된 중요 개체에 대한 접근성을 기반으로 수행된다.

[0128] (5.3.1) 권한 영역.

[0129] 중요한 개체는 권한있는 영역에서 격리된다. 페이지 테이블과 같은 쓰기 가능한 개체는 권한있는 데이터로 보호된다. 또한 예외 벡터와 권한 지역 코드는 페이지 단위 (4KB)에 따라 구분된다. 도 4와 같이 페이지 권한을 다르게 설정했다. 예외 벡터와 권한 코드는 모든 실행 가능, 권한 코드의 페이지는 쓰기 가능한 권한이 있는 반면 예외 벡터 페이지는 읽기 전용이다. 쓰기 가능한 권한은 비 권한 영역에 대한 항목에서 권한있는 코드의 실행 가능성을 동적으로 조정하기 위해 제공된다. 아래 권한 영역 보호 설명을 위한 섹션에서 권한 영역의 액세스 제어 메커니즘에 대해 논의하기로 한다.

[0130] (5.3.2) 비 권한 영역.

[0131] 비 권한 영역은 OS 커널 모니터링에 필요한 일반 작업을 수행한다. 앞에서 커널 무결성 모니터에서 설명했듯이 모니터는 모니터링되는 OS 커널에 대한 페이지 테이블 및 시스템 레지스터의 업데이트를 에뮬레이션해야 한다.

이전 작업 [12, 16, 30]과 유사하게 커널 TTBR과 같은 시스템 레지스터가 하이퍼 바이저에 의해 업데이트되도록 무결성 모니터를 구현했다. 특히 커널 페이지 테이블은 모니터의 권한이 없는 영역에서 관리하도록 강제한다. SelMon은 하이퍼 바이저에 적용되어 이전 접근 방식에 비해 트러스트 앵커를 보호하는 데 있어 타당성과 우수성을 강조한다. 따라서 커널 구현 무결성 모니터는 주요 기여가 아니다. 그러나 SelMon의 성능 평가를 위해 모니터링 커널의 최소한의 기능이 구현되었다.

[0132] (5.4)다음으로 권한 영역 보호에 대해 설명하기로 한다.

[0133] 하이퍼 바이저 무결성을 손상시키기 위해 악용될 수 있는 중요한 작업은 독점적으로 권한이 있는 영역에서 수행된다. 또한 페이지 테이블과 같은 중요한 데이터 개체는 권한 영역에서 격리된다. 따라서 권한이 없는 영역이 활성화되면 권한이 있는 영역을 실행하거나 액세스할 수 없어야 한다.

[0134] 비 실행 가능성.

[0135] 권한 영역이 실행되지 않도록 하기 위해 DEP 시행을 위한 하드웨어 기능을 이용한다. 하이퍼 바이저 시스템 제어 레지스터 (SCTLR_EL2)에서 WXN (Writable execute never (WXN) 플래그가 설정되면 하이퍼 바이저 영역의 모든 쓰기 가능한 페이지를 실행할 수 없다. 이 기능을 활용하기 위해 권한이 있는 코드의 페이지 권한을 쓰기 가능으로 구성한다((3)권한 분리의 권한 영역 섹션). 따라서 비 권한 영역의 항목에서 WXN 플래그를 설정하여 권한있는 영역의 비 실행 가능성을 동적으로 적용 할 수 있다.

[0136] 비 접근성.

[0137] 비 실행 기능 외에도 권한이 없는 영역이 활성화되는 동안 권한 영역에 액세스 할 수 없도록 해야한다. 그렇게 함으로써 페이지 테이블과 같은 중요한 데이터 개체뿐만 아니라 쓰기 가능한 권한으로 매핑된 권한있는 코드를 보호 할 수 있다. 이를 위해 일반 하드웨어 기능인 워치 포인트를 사용하여 권한 영역에 대한 읽기 또는 쓰기 액세스를 모니터링한다. 특히 하이퍼 바이저의 항목에 와치포인트를 구성하여 전체 권한 영역을 모니터링한다. 그런 다음 영역이 비 권한 영역으로 전환되면 와치포인트가 활성화된다.

[0138] 와치포인트 모니터링의 시작 주소는 2의 거듭 제곱에 맞춰야하는 모니터링 크기에 맞춰야 한다(앞서 설명한 하드웨어 디버깅 지원 참조). 이 요구 사항을 충족하기 위해 권한있는 영역에 8MB를 할당하고 8MB 단위에 맞게 영역의 시작 주소를 관리한다.

[0139] (5.5)지역 전환

[0140] 도 5은 SelMon을 사용한 모드 및 지역 스위치를 나타낸 도면이다. 이러한 도 5를 참조하여 권한 영역과 비 권한 영역 간의 영역 전환 흐름을 설명하기로 한다. 앞서 커널 무결성 모니터에 설명된 대로 OS 커널 모니터링을 위해 이식된 하이퍼 콜이 OS 커널에서 호출되면 하이퍼 콜 예외는 권한 영역의 예외 벡터에 의해 트랩된다. 벡터는 OS 요청 유형 (예 : TTBR 구성 또는 페이지 테이블 업데이트)을 확인하고 권한이 없는 영역에서 적절한 핸들러를 디스패치한다. 이 시점에서 권한 영역에서 비 권한 영역으로 영역 전환이 발생한다. 섹션 (4)권한 영역 보호에서 논의된 것처럼 권한 영역을 격리하기 위해 와치포인트와 DEP 플래그를 구성하는 보안 게이트를 통해 영역을 전환한다.

[0141] 반면에 권한이 없는 영역을 실행하는 동안 하이퍼 바이저 페이지 테이블 업데이트와 같은 보안에 중요한 작업에 대한 요청이 있을 수 있다. 예를 들어 하이퍼 바이저는 OS의 페이지 테이블을 업데이트하기 위해 OS 커널 영역에 대한 매핑을 생성해야 한다. 이 요청을 처리하려면 권한 영역으로 전환해야 한다. 권한 영역에서 예외 벡터에 의해 포착되는 하이퍼 콜을 이용하여 이 스위치를 관리한다. 영역의 각 항목에 대한 자세한 내용은 다음 섹션에서 설명하기로 한다.

[0142] (목록 4) : 와치포인트 관련 레지스터를 구성하는 하이퍼 바이저 항목.

[0143] 1 // ...(Omitted : save kernel registers) ...

[0144] 2 // Setup hypervisor trap for debug exception

[0145] 3

[0146] 4 L1:

[0147] 5 mov x4,# MDCR_TDE // Get the MDCR value

[0148] 6 msr MDCR_EL2,x4 // Set the MDCR

[0149] 7 isb // Synchronization barrier

[0150] 8 mov x5,# MDCR_TDE // Get the MDCR value

[0151] 9 cmp x4,x5 // Validate the set value

[0152] 10 b.ne L1 // If invalid, loop back

[0153] 11

[0154] 12 L2:

[0155] 13 mov x4,# MDSCR_KDE_MDE // Get the MDSCR value

[0156] 14 msr MDSCR_EL1,x4 // Set the MDSCR

[0157] 15 isb // Synchronization barrier

[0158] 16 mov x5,# MDSCR_KDE_MDE // Get the MDSCR value

[0159] 17 cmp x4,x5 // Validate the set value

[0160] 18 b.ne L2 // If invalid, loop back

[0161] 19

[0162] 20 L3:

[0163] 21 mov x4,# WPVALUE // Get the monitoring addr.

[0164] 22 msr DBGWVR0_EL1,x4 // Set the watchpoint addr.

[0165] 23 mov x5,# WPVALUE // Get the monitoring addr.

[0166] 24 cmp x4,x5 // Validate the set value

[0167] 25 b.ne L3 // If invalid, loop back

[0168] 26

[0169] 27 L4:

[0170] 28 mov x4,# WPCTLR_LO // Get the control flag (low)

[0171] 29 mov x5,# WPCTLR_HI // Get the control flag (high)

[0172] 30 add x4,x4,x5 // Get the full control value

[0173] 31 msr DBGWCR0_EL1,x4 // Configure the watchpoint

[0174] 32 mov x5,# WPCTLR_LO // Get the control flag (low)

[0175] 33 mov x6,# WPCTLR_HI // Get the control flag (high)

[0176] 34 add x5,x5,x6 // Get the full control value

[0177] 35 cmp x4,x5 // Validate the configured value

[0178] 36 b.ne L4 // If invalid, loop back

[0179] (5.5.1) 하이퍼 바이저 진입.

[0180] 커널 컨텍스트를 저장하는 것 외에도 SelMon은 CPU 모드가 하이퍼 바이저로 전환될 때 와치포인트 관련 레지스터를 구성한다 (목록 4). MDCR_EL2의 TDE 플래그는 와치포인트 예외를 하이퍼 바이저로 라우팅하도록 설정된다. KDE 및 모니터 디버그 이벤트 (MDE) 플래그는 하이퍼 바이저 모드에서 와치포인트 예외 생성을 허용하도록 설정된다. 그런 다음 DBGWCR 및 DBGWVR을 설정하여 전체 권한 영역을 모니터링한다. 와치포인트 구성이 정적이기 때문에 레지스터에 대해 미리 정의된 상수 값을 로드한다. 또한 와치포인트 레지스터를 설정한 후 공격자가 와치

포인트 기반 보호를 무력화하기 위해 구성 명령을 악용하지 못하도록 현재 구성된 값이 미리 정의된 값인지 확인한다. 이러한 레지스터를 설정한 후에도 PSTATE의 디버그 (D) 플래그가 마스킹되므로 와치포인트가 비활성화된 상태로 유지된다. 하드웨어 디버깅 지원 섹션에서 언급했듯이, PSTATE.D는 예외가 발생할 때마다 항상 마스킹(삭제)된다.

[0181] (목록 5) : DEP 및 와치 포인트를 활성화하는 비 권한 영역에 대한 항목.

[0182] 1 L5:

[0183] 2 mov x4,# SCTLR_WXN_LO // Get the SCTLR value

[0184] 3 add x4,x4, # SCTLR_WXN_HI

[0185] 4 msr SCTLR_EL2,x4 // Set the SCTLR

[0186] 5 isb // Synchronization barrier

[0187] 6

[0188] 7 mov x5,# SCTLR_WXN_LO // Get the SCTLR value

[0189] 8 add x5,x5, # SCTLR_WXN_HI

[0190] 9 cmp x4,x5 // Validate the set value

[0191] 10 b.ne L5 // If invalid, loop back

[0192] 11

[0193] 12 tlbi ALLE2 // Invalidate the TLB

[0194] 13 isb // Synchronization barrier

[0195] 14

[0196] 15 adr x4,savedPrivStack

[0197] 16 str sp,[x4] // Save the priv stack

[0198] 17 adr x4,savedNonPrivStack

[0199] 18 ldr x4,[x4]

[0200] 19 mov sp,x4 // Switch to the non - priv stack

[0201] 20

[0202] 21 msr DAIFclr, #8 // Enable the debug exception

[0203] 22 b non_priv_reg // Jump to the non - priv region

[0204] (5.5.2) 비 권한 영역으로의 입장.

[0205] 비 권한 영역에 대한 항목에서 권한 영역이 숨겨진다. 이는 비 권한 영역으로의 영역 전환을 처리하는 게이트에 의해 실현된다 (목록 5). 권한 영역의 실행 불가를 강제하기 위해 SCTLR_EL2에서 WXN 플래그를 설정한다. 따라서 쓰기 가능으로 설정된 권한있는 지역 코드는 실행 불가능하도록 강제된다. 이 시스템 레지스터를 설정한 후 구성된 값이 미리 정의된 값과 같은지 확인하여 구성 오용을 방지한다.

[0206] SCTLR_EL2도 MMU를 제어하므로 공격자가 이 명령을 실행하여 MMU를 비활성화할 수 있다. 이러한 시도를 방지하기 위해 하이퍼 바이저 가상 주소를 물리적 주소와 동일한 것으로 매핑한다. 따라서 MMU가 비활성화된 경우에도 SCTLR_EL2 설정을 따르는 값 확인 루틴은 여전히 ????유효하다. WXN 사용의 단점은 TLB 플러시가 필요하다는 것이다. 하이퍼 바이저 모드는 주소 공간 식별자 (ASID)를 제공하지 않기 때문에 하이퍼 바이저에 대한 전체 TLB를 플러시한다. 반면에 ASID는 보안 및 비보안 상태 모두에 대해 커널 모드에서 사용할 수 있다. 따라서 SelMon이 커널 모드 보호에 적용될 때 성능 향상을 기대한다. WXN을 활성화한 후 권한이 있는 영역과 권한이 없는 영역에 대한 스택을 각각 저장하고 복원한다.

[0207] 스택은 코어 단위로 할당되지만 현재 CPU ID를 기반으로 하는 스택 피벗 논리는 생략된다. 마지막으로 PSTATE에서 디버그 마스크 비트를 지우고 권한이 없는 영역으로 이동하여 와치포인트 모니터링을 활성화한다.

[0208] (목록 6) : 권한 영역으로의 입장. 와치포인트를 자동으로 비활성화하는 하이퍼 콜 (hvc)은 예외 벡터에 의해 트랩된다. 권한 영역에 들어가기 전에 DEP가 비활성화된다.

[0209] 1 /* ********** Non - privileged region ********** */

[0210] 2 hvc # REQNO // Hypervisor call

[0211] 3

[0212] 4 /* ************ Exception vector ************ */

[0213] 5 // ...(Omitted : verify the current exception) ...

[0214] 6

[0215] 7 L6:

[0216] 8 mov x4,# SCTLR_NOWXN_LO // Get the SCTLR value

[0217] 9 add x4,# SCTLR_NOWXN_HI

[0218] 10 msr SCTLR_EL2,x4 // Set the SCTLR

[0219] 11 isb // Synchronization barrier

[0220] 12

[0221] 13 mov x5,# SCTLR_NOWXN_LO // Get the SCTLR value

[0222] 14 add x5,# SCTLR_NOWXN_HI

[0223] 15 cmp x4,x5 // Validate the set value

[0224] 16 b.ne L6 // If invalid, loop back

[0225] 17

[0226] 18 tlbi ALLE2 // Invalidate TLB

[0227] 19 isb // Synchronization barrier

[0228] 20

[0229] 21 adr x4,savedNonPrivStack

[0230] 22 str sp,[x4] // Save the non - priv stack

[0231] 23 adr x4,savedPrivStack

[0232] 24 ldr x4,[x4]

[0233] 25 mov sp,x4 // Switch to the priv stack

[0234] 26

[0235] 27 b priv_reg // Jump to the privileged region

[0236] (5.5.3) 권한 영역으로의 입장.

[0237] 권한 영역에 대한 항목은 예외 벡터에 의해 트랩되는 하이퍼 호출을 호출하기 위해 비 권한 코드가 필요하다. 하드웨어 디버깅 지원 섹션에 설명된 대로 예외는 PSTATE.D를 자동으로 설정하여 와치포인트 모니터링을 비활성화하므로 권한있는 영역에 액세스할 수 있다. 또한 벡터는 SCTLR_EL2의 WXN을 지워서 권한있는 코드를 실행 가능하게 만들어 데이터 실행 방지(DEP)를 끈다(Turn Off). 스택은 권한 영역에 대해서도 전환된다.

[0238] 권한 코드와 달리 권한 영역 (목록 6)으로 가는 게이트는 활성화된 영역에 관계없이 항상 액세스 가능(실행 가능)하다. 따라서 공격자는 게이트 코드 중간에서 직접 점프하여 데이터 실행 방지(DEP)를 비활성화할 수 있다.

이렇게 하면 하이퍼 콜을 호출하지 않고 권한 영역을 실행할 수 있다.

[0239] 그러나 이러한 악의적인 동작은 예외가 없어 와치포인트가 여전히 활성화되어 있기 때문에, SelMon에 의해 쉽게 탐지될 수 있다. 특히 와치포인트 모니터링을 비활성화하지 않고 게이트 코드를 계속 실행하면 예외 신드롬 레지스터 (ESR)의 예외 클래스 (EC) 필드를 조사하여 구별할 수 있는 와치포인트 예외가 발생한다. 예를 들어 목록 6의 22 행은 비 권한 영역 스택의 저장소인 savedNonPrivStack이 활성 와치포인트에서 모니터링하는 권한있는 데이터의 일부이기 때문에 와치포인트 예외를 발생시킨다.

[0240] (5.5.4) 하이퍼 바이저를 종료한다.

[0241] 하이퍼 바이저를 종료하면 일반 레지스터 및 와치 포인트 구성을 포함하여 저장된 커널 컨텍스트가 복원된다. 이 루틴은 권한이 없는 지역에도 노출되기 때문에 공격자는 구성을 업데이트하는 지침을 남용하려고 할 수 있다. 그러나 종료 루틴은 CPU 모드를 커널에 대한 권한을 박탈하고 커널에서 하이퍼 콜 호출 직후 지점으로 돌아가는 eret 명령어로 종료된다. 따라서 공격자는 하이퍼 바이저 권한도 잃게 된다.

[0242] (5.6)디버깅 활동과의 호환성

[0243] 와치포인트는 서로 다른 프로세서 모드 간에 공유되는 리소스이다. 그러나 SelMon은 하이퍼 바이저 모드 진입 및 종료시 이전 설정을 저장하고 복원하므로 현재 보호 모드 외부의 와치포인트 사용을 방해하지 않는다. 예를 들어 사용자 및 커널 모드 디버거, 즉 데이터 중단 점(breakpoint)을 설정하기 위해 와치포인트를 사용하는 gdb [9] 및 kgdb [11]은 SelMon이있는 상태에서도 계속 사용할 수 있다. 또한 DEP의 동적 재구성은 시스템 제어 레지스터가 각 모드에 대해 뱅크되기 때문에 다른 모드의 보안에 영향을 주지 않는다. 하이퍼 바이저 및 신뢰할 수 있는 펌웨어와 같은 권한있는 소프트웨어를 디버깅하려면 일반적으로 JTAG 인터페이스가 있는 하드웨어 디버거를 사용한다. 하드웨어 디버거는 데이터 액세스를 모니터링하도록 와치포인트를 구성하기 때문에 원하지 않는 구성 손상을 방지하기 위해 SelMon을 비활성화해야 한다.

[0244] (표 3) : SelMon 구축에 활용되는 시스템 레지스터.

[0245]

System register	Description
DBGWVR & DBGWCR	Configure watchpoint
DAIFCLR	(Un)mask debug exception
MDCR	Configure watchpoint exception handling location
SCTLR	Configure DEP
ESR	Validate trapped exception

[0246] (6) 구현

[0247] 코드 줄. SelMon은 수동으로 검사하거나 공식적으로 검증할 수 있을 만큼 충분히 작다 [35]. 현재 하이퍼 바이저 구현의 LoC는 중요한 작업 에뮬레이션을 호출하기 위한 커널 패치를 포함하지 않고 어셈블리에서 약 770 개이다 (각각 권한있는 영역과 비 권한 영역의 경우 250 LoC). 구현에서 권한이 없는 영역에서 수행하는 작업은 페이지 테이블 및 TTBR 업데이트와 같은 중요한 OS 커널 작업을 에뮬레이션하는 것으로 제한된다 (이는 Sel-Mon의 실행 가능성을 입증하는 데 충분하다). 이전 작업에서 설명한 추가 보안 구성 요소 [16] (예 : 커널 메모리 이중 매핑 추적)는 구현되었다. 따라서 커널 모니터가 완전히 구현되면 권한이 없는 영역의 LoC가 더 커질 수 있다.

[0248] Thin Hypervisor. 하이퍼 바이저 구현을 위해 Linux 장치 트리 파일 (juno.dtsi)을 수정하여 물리적 메모리 범위를 0xFE000000에서 0xFEFFFFFF (16MB)까지 예약했다. 그런 다음 하이퍼 바이저 격리를 위해 예약된 영역을 제외한 모든 물리적 메모리 범위를 매핑하는 보조 페이지 테이블을 생성했다. 0xFE000000에서 보조 페이지 테이블을 찾는다. 페이지 테이블 설정이 완료되면 가상화 변환 테이블 기본 레지스터 (VTTBR_EL2), 가상화 변환 제어 레지스터 (VTCR_EL2) 및 하이퍼 바이저 구성 레지스터 (HCR_EL2)와 같은 보조 페이징을 활성화하기 위한 제어 레지스터가 구성된다.

[0249] 커널 패치. 하이퍼 바이저에서 커널 무결성 모니터링 기능을 구현하기 위해 이전 접근 방식 [16, 30]을 따른다. TTBR 및 VBAR와 같은 보안에 중요한 시스템 레지스터를 설정하는 지침은 하이퍼 바이저 호출 (예 : hvc)로 대체된다. 또한 페이지 테이블 관리 함수 (예 : set_pte 및 set_pmd)에 하이퍼 바이저 호출을 삽입하여 하이퍼 바이저에서 쓰기 방지된 커널 페이지 테이블의 업데이트를 확인하고 에뮬레이션한다. 커널 텍스트를 호스팅하는 물리적 메모리는 보조 페이지 테이블에서 읽기 전용으로 설정되어 텍스트의 패치도 변경할 수 없다. SelMon. 씬

하이퍼 바이저에 SelMon을 적용하기 위해 하이퍼 바이저 메모리 (16MB)를 각각 8MB 씩 두 개의 절반으로 나눈다.

[0250] 각 지역. 권한 영역 위치와 크기는 2의 거듭 제곱에 맞춰져 있으므로 와치포인트 설정 요구 사항 (하드웨어 디버깅 지원 섹션)을 충족한다. 구현시 SelMon을 활성화하기 위해 하나의 와치포인트만 필요하다. 그러나 각 영역의 위치와 크기는 여러 와치포인트를 사용하여 유연하게 조정할 수 있다.

[0251] 마지막으로 표 3은 SelMon이 지역 내 권한 분리를 실현하기 위해 사용하는 시스템 레지스터를 요약한 것이다. 이들은 고급 ARM 프로세서 (예 : ARMv8-A [6]) 사양의 일부로 정의되며 제조업체에서 의도적으로 비활성화하거나 수정하지 않는 한 일반적으로 프로덕션 장치에서 지원된다.

[0252] 시스템 레지스터를 사용할 수 있더라도 프로덕션 장치에 SelMon을 배포하려면 제조업체의 개입이 필요하다. 이는 부팅시 제조업체의 키를 사용하여로드 된 이미지 (예 : 보안 OS 및 하이퍼 바이저)의 무결성을 확인하는 보안 부팅 [15]이 일반적으로 최신 장치에 사용되기 때문이다. 마이크로 컨트롤러의 로우 엔드 사양 인 ARMv8-M은 워치 포인트를 포함한 디버그 기능도 정의한다.

[0253] 그러나 ARMv8-A와 ARMv8-M 간의 아키텍처 불일치로 인해 저가형 장치에서 SelMon을 빌드하려면 더 많은 설계 고려 사항이 필요할 것으로 예상된다. 섹션 8에서 이에 대해 논의한다.

[0254] (7) 평가

[0255] 이 섹션에서는 먼저 SelMon의 가능한 공격 표면을 분석하고 효과적으로 차단되는 방법에 대해 논의한다. 그런 다음 SelMon에서 발생한 성능 오버 헤드를 측정한다.

[0256] (7.1) 보안 분석

[0257] SelMon의 보안은 일반적인 하드웨어 기능 (예 : 와치포인트 및 DEP)의 적절한 구성에 따라 다르다. 공격자가 구성을 성공적으로 조작하면 SelMon을 우회 할 수 있다. 이와 관련하여 SelMon이 공격자가 기능에 액세스하는 것을 효과적으로 제한하는 방법을 설명한다.

[0258] 하이퍼 바이저 코드. 이전 작업 [16, 17, 21, 27]과 유사하게 비 권한 지역 코드에는 중요 시스템 레지스터 (예 : TTBR 및 VBAR)를 구성 할 수 있는 명령이 포함되어 있지 않다고 가정한다. 또한 페이지 테이블 및 와치포인트 업데이트 지침은 해당 영역에 없다. 비 권한 지역 코드는 하이퍼 바이저 페이지 테이블에서 읽기 전용 권한으로 매핑되기 때문에 공격자는 지역을 수정할 수 없다. 공격자의 나머지 옵션은 쓰기 가능 영역에 악성 코드를 삽입하고 실행하는 것이다. 그러나 이 공격은 비 권한 영역에 대한 진입 점에서 항상 DEP를 활성화하기 때문에 단순히 방지된다.

[0259] 반면에 보안에 중요한 작업은 권한이 있는 영역에서 수행된다. 섹션 (5.4)권한 영역 보호에서 설명했듯이 권한 영역은 비 권한 영역 활성화 중에 액세스할 수없고 실행 가능하지도 않다. 따라서 공격자는 권한있는 영역을 조작 할 수 있을뿐만 아니라 해당 영역에서 중요한 명령을 재사용할 수 없다.

[0260] 모드 스위치. 섹션 5.5.1 및 5.5.4에서 설명했듯이 와치포인트는 하이퍼 바이저로 들어가고 나올 때 구성된다. 하이퍼 바이저 시작 및 종료 핸들러는 읽기 전용으로 매핑되지만 현재 영역 권한에 관계없이 여전히 실행 가능하다. 따라서 공격자는 와치포인트 구성 지침을 남용하려고 할 수 있다. TDE, KDE 및 MDE 플래그 중 하나를 지우면 감시 지점 모니터링이 비활성화되어 공격자가 권한있는 영역에 액세스 할 수 있다. 와치포인트 주소 (DBGWVR) 및 제어 (DBGWCR) 레지스터를 악의적으로 재구성하면 와치포인트 보호도 비활성화된다.

[0261] 하이퍼 바이저 항목에서 와치포인트 관련 레지스터는 미리 정의된 값으로 설정된다. 따라서 구성 직후 상수 값으로 확인할 수 있다. 모드가 커널로 다시 전환되면 레지스터 값이 커널 설정으로 복원된다. 커널 설정은 권한있는 영역에 저장되기 때문에 공격자가 조작 할 수 없다. 또한 공격자가 하이퍼 바이저 권한을 직접 잃기 때문에 종료 루틴을 악용하는 것은 유리하지 않다. 즉, 종료 루틴의 마지막 명령인 eret은 CPU 모드를 커널로 전환한다. 지역 전환. 리전 스위치간에 SelMon은 PSTATE에서 DEP 및 디버그 마스킹 (D) 플래그를 구성한다. PSTATE.D는 즉 치값 (# 8)을 사용하여 구성되므로 공격자는 악성 값을 전달하는 일반 레지스터로 플래그를 재구성하라는 명령을 남용할 수 없다. 시스템 제어 레지스터 (SCTLR)를 변경해야하므로 DEP 구성에 주의해야 한다. 와치포인트 구성 보호와 유사하게 SCTLR에 기록 된 후 구성된 값을 확인한다. MMU와 같은 다른 시스템 설정은 SCTLR에 의해 제어되기 때문에 DEP 플래그뿐만 아니라 다른 플래그도 보호해야한다. SCTLR 값은 확인이 간단하도록 일정해야한다. 기록된 값을 미리 정의된 합법적인 값과 비교하기만 하면 된다. 구성과 검증 사이의 타이밍 차이로 인해 MMU를 잠시 비활성화 할 수 있다. 하나, 가상 주소를 실제 주소와 동일하게 매핑하기 때문에 검증

은 여전히 ????효과적이고 결정적이다. 따라서 공격자는 확인을 우회할 수 없다. 특히, 영역 전환이 성공하려면 사전 정의된 인터페이스인 하이퍼 콜을 사용하여 권한 영역으로의 게이트를 실행해야 한다. 하이퍼 콜 호출없이 게이트로 점프하는 모든 시도는 와치 포인트 예외를 생성한다. 마지막으로 하이퍼 바이저 모드에서는 기능이 작기 때문에 인터럽트를 활성화하지 않는다. 따라서 게이트 코드 실행 중 인터럽트는 보안 관점에서 고려할 필요가 없다 [27]. 성공적인 지역 전환을 위해. 하이퍼 콜 호출없이 게이트로 점프하는 모든 시도는 와치 포인트 예외를 생성한다. 마지막으로 하이퍼 바이저 모드에서는 기능이 작기 때문에 인터럽트를 활성화하지 않는다. 따라서 게이트 코드 실행 중 인터럽트는 보안 관점에서 고려할 필요가 없다 [27]. 성공적인 지역 전환을 위해. 하이퍼 콜 호출없이 게이트로 점프하는 모든 시도는 와치 포인트 예외를 생성한다. 마지막으로 하이퍼 바이저 모드에서는 기능이 작기 때문에 인터럽트를 활성화하지 않는다. 따라서 게이트 코드 실행 중 인터럽트는 보안 관점에서 고려할 필요가 없다 [27].

[0262] 인터럽트가 활성화된 경우에도 단일 예외 벡터가 권한 영역과 비 권한 영역 모두에 사용되기 때문에 SelMon의 보안을 방해하지 않는다. 이에 본 실시예에서는 인터럽트가 권한 영역에 의해 안전하게 처리되도록 강제할 수 있다. 신뢰할 수 없는 커널. 공격자가 신뢰할 수 없는 커널을 공모하는 것을 고려해야 한다. 예를 들어, 공격자는 하이퍼 바이저 권한이 확보되면 악성 커널 코드로 돌아갈 수 있다. 따라서 하이퍼 바이저 모드가 활성화되면 하이퍼 바이저 이외의 다른 영역을 실행할 수 없다. 이 공격 벡터를 제거하는 것은 간단하다. 하이퍼 바이저 페이지 테이블에서 실행 안 함 (NX) 권한으로 커널 영역을 매핑하기 만하면 된다.

[0263] 실제 공격에 대한 효과. SelMon은 소프트웨어 취약점을 탐지하거나 제거하는 것을 목표로 하지 않기 때문에 공격자가 트러스트 앵커의 기존 취약점을 악용하는 것을 막을 수 없다 ([3, 4, 12]). 그러나 시스템의 기본 방어 수단으로 SelMon은 트러스트 앵커의 무결성을 보호하고 중요한 작업을 격리하여 공격자의 시스템 손상 능력을 크게 제한한다.

[0264] 7.2 성능

[0265] 하이퍼 바이저, 애플리케이션 및 OS 성능을 측정하여 SelMon의 오버 헤드를 평가한다. 특히, 커널 무결성 모니터 (예 : TTBR 업데이트)를 구현하는 데 필요한 하이퍼 바이저 운영 프리미티브의 성능을 측정한다. 그런 다음 SPEC CPU2006, LMBench 및 Phoronix Test Suite라는 세 가지 벤치 마크를 실행하여 원본 및 SelMon 강화 하이퍼 바이저에서 실행되는 애플리케이션 및 OS의 성능을 측정했다.

[0266] (표 4)

Operation	Baseline	SelMon	Stdev	Overhead
Mode switch	3.2	4.2	0.63	1.31×
TTBR update	4.0	5.2	1.03	1.30×
Page table update	4.4	6.5	1.08	1.47×

[0267]

[0268] 표 4는 원래 및 강화된 하이퍼 바이저의 간단한 작동 성능. SelMon의 각 사례에 대한 10 회 실행의 평균 지연 시간 (μs) 및 표준 편차가 표시된다.

[0269] 7.3 하이퍼 바이저

[0270] SelMon을 채택하면 권한 분리 및 도메인 전환으로 인해 오버 헤드가 발생한다. 시스템 설계의 (5.1)SelMon 섹션에서 설명했듯이 OS 커널을 모니터링하는 씬 하이퍼 바이저에 SelMon을 적용했다. 그런 다음 TTBR, 페이지 테이블 및 커널 메모리 업데이트와 같은 하이퍼 바이저 작업의 성능이 평가된다. 특히, SelMon 강화 및 원본 하이퍼 바이저에서 수행된 각 작업에 대한 시간을 측정하고 비교한다.

[0271] 표 4에서 볼 수 있듯이, SelMon은 커널과 하이퍼 바이저 간의 간단한 전환에 31 %의 오버 헤드를 부과한다. 이 오버 헤드에는 하이퍼 바이저 항목, 비 권한 영역에 대한 항목 및 하이퍼 바이저 종료에 대한 일정한 시간이 포함된다. 간단한 전환 시간 외에도 TTBR 업데이트 케이스는 비 권한 영역에서 업데이트 명령을 실행하는 데 더 많은 시간이 필요하다. 명령 실행시 추가 영역 전환이 필요하지 않기 때문에 해당 명령에 비해 오버 헤드가 증가하지 않는다.

[0272] 모드 스위치. 반면에 페이지 테이블 업데이트 케이스에는 비 권한 영역과 권한 영역 간의 전환이 필요하다. 업데이트 요청은 비 권한 영역에서 하이퍼 바이저 호출을 호출하여 전송되며, 이 호출은 영역을 권한으로 전환한다. 페이지 테이블이 업데이트되면 비 권한 영역이 영역 전환 게이트를 통해 다시 활성화된다. 이러한 추가 작

업은 다른 두 경우에 비해 SelMon의 오버 헤드를 증가시키는 주요 요인이다.

[0273] OS 커널 액세스 성능도 추정된다 (표 5). 단일 읽기 및 쓰기 작업은 단순 스위치와 유사한 약 30 %의 오버 헤드를 부과했다. 그러나 처리 데이터 크기가 증가함에 따라 오버 헤드가 가려졌다. 쓰기 및 읽기 작업의 크기가 80 바이트보다 크면 오버 헤드가 각각 6 % 및 2 %로 크게 감소했다. OS 커널 영역을 하이퍼 바이저에 미리 매핑한다. 따라서 페이지 테이블 업데이트 (즉, 하이퍼 바이저 또는 보조 페이지 테이블)에 대한 오버 헤드는이 결과에 포함되지 않는다.

[0274] 7.4 애플리케이션 및 OS 벤치 마크

[0275] SelMon이 애플리케이션에 미치는 성능 영향은 LMBench 및 SPEC CPU2006 벤치 마크를 실행하여 평가된다. 반면에 우리는 Phoronix Test Suites에서 제공하는 커널 테스트 모음을 사용하여 OS 커널에 부과 된 오버 헤드를 평가했다.

[0276] LMBench. 오픈 및 execve와 같은 기본 OS 작업의 지연 시간을 두 가지 환경 (원래 및 강화 된 하이퍼 바이저에서 실행되는 OS)에서 측정한다. 표 6은 마이크로 초 단위의 결과를 나타낸다. syscall, read, write, stat, open, signal handler 및 sock를 포함하여 짧은 지연 시간만 도입하는 간단한 작업은 SelMon의 큰 영향을 나타내지 않았다. 이는 각 테스트 케이스가 실행 시간을 측정하는 동안 하이퍼 바이저를 호출하지 않기 때문이라고 생각한다. 보다 구체적으로, 하이퍼 바이저가 컨텍스트 전환 (TTBR 업데이트) 및 OS 페이지 테이블 업데이트에 대해 호출된다는 점을 고려하면 이러한 테스트에는 측정 중에 이러한 OS 작업이 포함되지 않는다.

[0277] (표 5)

	8 bytes		80 bytes		800 bytes	
Operation	Baseline	SelMon	Baseline	SelMon	Baseline	SelMon
Read	3.8	5.1 (σ: 0.56, 1.34×)	7.9	8.1 (σ: 1.1, 1.02×)	50.7	52.1 (σ: 3.63, 1.02×)
Write	4.4	5.7 (σ: 0.67, 1.29×)	7.8	8.3 (σ: 0.82, 1.06×)	51	53.7 (σ: 2.98, 1.05×)

[0278]

[0279] 표 5 : 원본 및 SelMon 강화 하이퍼 바이저에 대한 읽기 및 쓰기 작업의 성능. 10 회 실행 (μs 단위)의 평균 지연 시간, 표준 편차 및 오버 헤드가 설명되어 있다. 작업 크기는 8 바이트에서 800 바이트까지 다양하다. 대상 메모리 영역은 미리 매핑된다. 따라서 작업에는 페이지 테이블 업데이트의 오버 헤드가 포함되지 않는다.

[0280] (표 6)

	Baseline	SelMon	Stdev	Overhead
syscall	0.690	0.690	0.004	1.00×
read	1.545	1.546	0.009	1.00×
write	1.310	1.313	0.004	1.00×
stat	4.939	4.953	0.012	1.00×
open+close	10.886	10.956	0.209	1.01×
signal handler	6.532	6.534	0.225	1.00×
sock stream	26.093	26.531	0.919	1.02×
fork+exit	565.940	745.142	7.134	1.32×
fork+execve	1466.65	1899.20	60.620	1.29×
\bin\sh -c	5639	7249	37.824	1.29×

[0281]

[0282] 표 6 : 원본 및 SelMon 강화 하이퍼 바이저에서 Linux OS의 LMBench 결과. 각 SelMon 테스트 케이스의 10 회 실행의 평균 지연 시간 (μs) 및 표준 분할이 설명되어 있다.

[0283] 도 6는 원본 및 SelMon 강화 하이퍼 바이저가 있는 SPEC CPU2006을 나타낸 도면이다. 결과는 원래 하이퍼 바이저의 측정 값으로 정규화된다(낮을수록 좋음).

[0284] 도 7는 원본 및 SelMon 강화 하이퍼 바이저와 함께 Phoronix Test Suite를 실행하여 평가한 OS 성능을 나타낸 도면이다. SelMon을 사용한 평균 5 회 실행은 원래 하이퍼 바이저의 실행으로 정규화된다(낮을수록 좋음).

[0285] 그러나 마지막 세 가지 테스트 사례, 즉 fork, execve 및 shell 실행은 약 30 %의 오버 헤드를 부과했으며, 이는 하이퍼 바이저 운영 프리미티브에 부과된 SelMon의 오버 헤드와 일치한다. 섹션 7.3에 설명된 대로 모드 및

지역 스위치의 지속적인 대기 시간으로 인해 오버 헤드가 발생했다. 결과는 마지막 세 사례가 새 프로세스 생성을 위해 OS 페이지 테이블을 업데이트하는 하이퍼 바이저 작업을 자주 호출함을 나타낸다. 또한 최근 세 가지 테스트 사례의 상대적으로 높은 지연 시간으로 인해 TTBR 업데이트를 지원하기 위해 하이퍼 바이저를 호출하는 여러 컨텍스트 스위치가 있을 수 있다.

[0286] SPEC CPU2006. 도 6는 강화된 하이퍼 바이저를 사용한 CPU2006 테스트 케이스의 성능을 나타낸다. 결과는 원래 하이퍼 바이저로 측정된 성능으로 정규화된다.

[0287] LMBench의 결과와는 달리 CPU2006으로 측정된 오버 헤드는 무시할 수있는 수준이었다. sjeng에서 최대 2 %의 오버 헤드가 관찰되었다. 특히 SelMon이 포함 된 bzip2는 더 나은 성능을 나타낸다.

[0288] 이 결과의 원인을 철저히 분석하지는 않았지만 다양한 요인 (예 : CPU 스로틀 링)에 의해 발생하는 노이즈 때문일 것으로 예상된다. 결과는 하이퍼 바이저를 호출하는 작업 (즉, 페이지 테이블 및 TTBR 업데이트) 부분이 각 애플리케이션에서 중요하지 않음을 의미한다. LMBench에 비해 런타임 (초)이 더 길기 때문에 대부분의 SelMon 오버 헤드가 가려졌다.

[0289] Phoronix 테스트 스위트. SelMon 채택으로 인해 OS 커널에 미치는 성능 영향을 평가한다. 도 7에서 볼 수 있듯이 Phoronix Test Suite의 10 가지 커널 테스트 프로그램이 사용된다. 오버 헤드는 OpenSSL 및 MAFFT에서 1 %에서 5 %까지 다양하다. MAFFT 외에도 7-ZIP 및 bzip2는 4 %의 상대적으로 높은 오버 헤드를 도입한다. 도 7의 bzip2는 도 6의 bzip2 (최대 50MB)보다 훨씬 더 많은 입력 (256MB)을 사용한다. 대용량 파일 압축에는 디스크 읽기 및 복사 작업으로 인해 빈번한 컨텍스트 전환 및 페이지 오류가 필요하다. 이로 인해 컨텍스트 스위치 및 페이지 오류 처리기에 삽입 된 하이퍼 호출로 인해 하이퍼 바이저가 더 자주 호출된다.

[0290] 마지막으로 SPEC CPU2006과 비교할 때 Phoronix의 대부분의 테스트 사례는 커널 집약적인 작업으로 인해 더 높은 오버 헤드를 초래한다.

[0291] 8 토론

[0292] SelMon을 하이퍼 바이저로 이식하여 SelMon의 효율성을 보여 주지만 커널 모드에서 실행되는 소프트웨어를 보호하도록 조정할 수 있으며 필요한 하드웨어 기능 (예 : DEP 및 와치포인트)도 제공한다. 이 기능은 일반적으로 보안 상태에서 모두 사용할 수 있기 때문에 REE (풍부한 실행 환경)에서 호스팅되는 OS 커널과 TrustZone에서 만든 TEE는 SelMon으로 무장하여 자체 보호를 수행 할 수 있다. 이를 실현하기 위해서는 약간의 엔지니어링 노력이 필요하다. 예를 들어, 중요한 작업을 확인하고 에뮬레이션하는 권한 부분은 SelMon이 보호하는 지역에 위치해야 한다. 가장 중요한 것은 가상 및 물리적 주소가 동일하게 매핑된 메모리 영역에 게이트 코드를 할당해야 한다는 것이다. 이는 공격자가 SCTLR을 업데이트하는 명령을 남용하는 것을 방지하기 위한 것이다. MMU를 끄는 데 악용될 수 있다. 이러한 요구 사항에도 불구하고 우리의 접근 방식은 보안 응용 프로그램을 위해 예약할 특정 메모리 주소 (예 : 0x0)를 요구하지 않는다는 점에서 SKEE [17]보다 더 유연하다. 우리의 접근 방식은 앞서 언급 한대로 TEE에서 보안 OS를 보호하여 Trust-Zone의 보안을 강화하는 데 도움이 되지만 일반적으로 TEE의 게이트 키퍼 역할을 하는 모니터 모드와 호환이 안된다. 본질적으로 모니터 모드에서 와치 포인트 예외가 지원되지 않기 때문이다. 앞에서 권한 분리 섹션에서 논의한 바와 같이 이전 접근 방식은 모니터 모드의 보호에도 적합하지 않다. SelMon과 소프트웨어 결함 분리 (SFI) [19, 38, 45]를 혼성화하는 것은이 문제를 해결하기위한 합리적인 접근 방법이 될 수 있다. 예를 들어, 와치 포인트 지원 부족, 비 접근성을 강제하는 것은 읽기 및 쓰기 작업을 계측하여 보상할 수 있으므로 권한 영역에 액세스하지 않도록 강제할 수 있다.

[0293] ARM 마이크로 컨트롤러 (예 : ARMv8-M)의 사양은 와치 포인트를 포함한 디버깅 속성도 정의한다. 그러나 이 저가형 아키텍처로 SelMon을 재현하려면 아키텍처 차이로 인해 현재 디자인을 수정해야 한다.

[0294] 예를 들어 로우 엔드 사양은 MMU 및 DEP를 지원하지 않는다. 따라서 페이지 테이블을 사용하여 각 지역의 구성 요소에 대한 권한을 설정하고 DEP 정책을 동적으로 조정할 수 없다. 또한 리소스 제약 장치를 구축하는 데 드는 비용을 줄이기 위해 일반적으로 디버그 기능을 장치에서 제거해야 한다. 메모리 보호 장치 (MPU)를 컴파일러 기술 [22]과 함께 사용하면 이러한 누락 된 구성 요소를 에뮬레이션할 수 있는 가능한 솔루션이 될 것으로 기대한다. 저사양 장치에서 SelMon을 구현하기 위해 이를 추가로 살펴볼 것이다. 반면 x86은 하드웨어 와치포인트도 제공한다. 안타깝게도 가능한 모니터링 범위의 크기는 제한되어 있다.

[0295] 개별 와치포인트 당 8 바이트까지; 따라서 x86에서 와치포인트(watchpoint)를 사용하여 SelMon을 구현하는 데 적합하지 않다. 대신 메모리 영역을 16 개 부분으로 분할하고 각 영역에 대한 액세스를 선택적으로 비활성화 (및 활성화)하는 메모리 보호 키 [13]는 SelMon을 x86 기반 시스템에 배포하기 위한 실행 가능한 하드웨어 기능

이 될 수 있다.

[0296] 9 결론

[0297] 먼저 권한있는 소프트웨어의 자체 보호를 위한 기존 접근 방식을 분석하고 이러한 접근 방식이 일반적으로 모바일 장치에 채택되기에 충분하지 않은 이유를 보여주었다. 그런 다음 DEP 및 와치 포인트와 같은 일반 하드웨어 기능을 활용하여 ARM 아키텍처에서 권한있는 소프트웨어를 보호하는 메커니즘인 SelMon을 시연했다. 접근 방식의 효과를 보여주기 위해 OS 커널 무결성을 모니터링하는 씬 하이퍼 바이저에 SelMon을 적용했다. 성능 평가에서 SelMon 하이퍼 바이저 기본 작업에서 약 30 %의 오버 헤드를 부과했다. 그러나 대부분의 오버 헤드는 각 테스트 케이스의 전체 런타임에서 하이퍼 바이저 호출의 최소 부분으로 인해 SPEC CPU2006에서 가려졌다. Phoronix를 사용한 OS 성능 평가에서 최대 5 %의 오버 헤드가 관찰되었다.

[0298] 하이퍼 바이저에 SelMon을 이식했지만, 접근 방식의 이식성을 강조하는 필수 하드웨어 기능 (예 : 와치포인트)의 가용성으로 인해 보안 및 비보안 커널과 같은 다른 모드에서도 유사하게 채택될 수 있다.

[0299] 이상에서 설명한 본 발명은 전술한 실시예 및 첨부된 도면에 의해 한정되는 것이 아니고, 본 발명의 기술적 사상을 벗어나지 않는 범위 내에서 여러 가지 치환, 변형 및 변경이 가능함은 당업자에게 명백할 것이다.

[0300] [1] 2008. Architecture Reference Manual (ARMv7-A and ARMv7-R edition). ARM DDI C 406 (2008).

[0301] [2] 2017. Android for Work on Samsung KNOX devices. https://kp30.s3.amazonaws.com/0b2e7bf6ffc167b609daed41001d00e9.pdf

[0302] [3] 2017. Full TrustZone exploit for MSM8974. http://bits-please.blogspot.kr/2015/08/full-trustzone-exploit-for-msm8974.html

[0303] [4] 2017. Here Be Dragons: Vulnerabilities in TrustZone. http://atredispartners.blogspot.kr/2014/08/here-be-dragons-vulnerabilities-in.html

[0304] [5] 2018. AMD-V Nested Paging. http://developer.amd.com/wordpress/media/2012/10/NPT-WP-1%201-final-TM.pdf

[0305] [6] 2018. ARM Architecture Reference Manual ARMv8, for ARMv8-A architecture profile. https://developer.arm.com/docs/ddi0487/latest/arm-architecturereference-

[0306] manual-armv8-for-armv8-a-architecture-profile

[0307] [7] 2018. ARM Security Technology - Building a Secure System using TrustZone Technology. http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-

[0308] 009492c/PRD29-GENC-009492Ctrustzonesecuritywhitepaper.pdf

[0309] [8] 2018. Enabling Intel Virtualization Technology Features and Benefits.

[0310] https://www.intel.com/content/dam/www/public/us/en/documents/whitepapers/virtualization-enabling-intel-virtualization-technology-features-andbenefits-paper.pdf

[0311] [9] 2018. GDB: The GNU Project Debugger. https://www.gnu.org/software/gdb/

[0312] [10] 2018. Juno ARM넉 뱓 Development Platform SoC. http://infocenter.arm.com/help/topic/com.arm.doc.ddi0515f/DDI0515Fjunoarmdevelopmentplatformsoctrm.pdf

[0313] [11] 2018. Kgdb. https://elinux.org/Kgdb

[0314] [12] 2018. Lifting the (Hyper) Visor: Bypassing Samsung's Real-Time Kernel Protection. https://googleprojectzero.blogspot.com/2017/02/lifting-hyper-visorbypassing-samsungs.html

[0315] [13] 2018. Memory protection keys. https://lwn.net/Articles/643797/

[0316] [14] 2018. Second-level page table walk. https://developer.arm.com/docs/ddi0333/latest/memory-management-unit/mmu-descriptors/second-level-page-tablewalk

[0317] [15] William Arbaugh, David J Farber, Jonathan M Smith, et al. 1997. A secure and reliable bootstrap

architecture. In Security and Privacy, 1997. Proceedings., 1997 IEEE Symposium on. IEEE, 65??71.

[0318] [16] Ahmed M Azab, Peng Ning, Jitesh Shah, Quan Chen, Rohan Bhutkar, Guruprasad Ganesh, Jia Ma, and Wenbo Shen. 2014. Hypervision across worlds: real-time kernel protection from the ARM trustzone secure world. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. ACM, 90??102.

[0319] [17] Ahmed M. Azab, Kirk Swidowski, Rohan Bhutkar, Jia Ma, Wenbo Shen, Ruowen Wang, and Peng Ning. 2016. SKEE: A lightweight Secure Kernel-level Execution Environment for ARM. In 23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016.

[0320] [18] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. 2003. Xen and the art of virtualization. In ACM SIGOPS operating systems review, Vol. 37. ACM, 164??177.

[0321] [19] Kjell Braden, Lucas Davi, Christopher Liebchen, Ahmad-Reza Sadeghi, Stephen Crane, Michael Franz, and Per Larsen. 2016. Leakage-Resilient Layout Randomization for Mobile Devices.. In NDSS.

[0322] [20] Yaohui Chen, Dongli Zhang, Ruowen Wang, Rui Qiao, Ahmed M Azab, Long Lu, Hayawardh Vijayakumar, and Wenbo Shen. 2017. NORAX: Enabling executeonly memory for COTS binaries on AArch64. In Security and Privacy (SP), 2017 IEEE Symposium on. IEEE, 304??319.

[0323] [21] Yeongpil Cho, Donghyun Kwon, Hayoon Yi, and Yunheung Paek. 2017. Dynamic Virtual Address Range Adjustment for Intra-Level Privilege Separation on ARM. In 24th Annual Network and Distributed System Security Symposium, NDSS 2017, San Diego, California, USA, February 26 - March 1, 2017.

[0324] [22] Abraham A Clements, Naif Saleh Almakhdhub, Khaled S Saab, Prashast Srivastava, Jinkyu Koo, Saurabh Bagchi, and Mathias Payer. 2017. Protecting bare-metal embedded systems with privilege overlays. In 2017 IEEE Symposium on Security and Privacy (SP). IEEE, 289??303.

[0325] [23] Patrick Colp, Jiawen Zhang, James Gleeson, Sahil Suneja, Eyal de Lara, Himanshu Raj, Stefan Saroiu, and Alec Wolman. 2015. Protecting Data on Smartphones and Tablets from Memory Attacks. In Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '15). ACM, 177??189.

[0326] [24] John Criswell, Nathan Dautenhahn, and Vikram Adve. 2014. KCoFI: Complete control-flow integrity for commodity operating system kernels. In 2014 IEEE Symposium on Security and Privacy. IEEE, 292??307.

[0327] [25] John Criswell, Nathan Dautenhahn, and Vikram Adve. 2014. Virtual ghost: Protecting applications from hostile operating systems. In ACM SIGPLAN Notices, Vol. 49. ACM, 81??96.

[0328] [26] John Criswell, Andrew Lenharth, Dinakar Dhurjati, and Vikram Adve. 2007. Secure virtual architecture: A safe execution environment for commodity operating systems. ACM SIGOPS Operating Systems Review 41, 6 (2007), 351??366.

[0329] [27] Nathan Dautenhahn, Theodoros Kasampalis, Will Dietz, John Criswell, and Vikram Adve. 2015. Nested kernel: An operating system architecture for intrakernel privilege separation. In ACM SIGPLAN Notices, Vol. 50. ACM, 191??206.

[0330] [28] Xiaowan Dong, Zhuojia Shen, John Criswell, Alan L Cox, and Sandhya Dwarkadas. 2018. Shielding software from privileged side-channel attacks. In 27th {USENIX} Security Symposium ({USENIX} Security 18). 1441??1458.

[0331] [29] Andrew Ferraiuolo, Andrew Baumann, Chris Hawblitzel, and Bryan Parno. 2017. Komodo: Using verification to disentangle secure-enclave hardware from software. In Proceedings of the 26th Symposium on Operating Systems Principles. ACM, 287??305.

[0332] [30] Xinyang Ge, Hayawardh Vijayakumar, and Trent Jaeger. 2014. Sprobes: Enforcing Kernel Code Integrity on the TrustZone Architecture. Proceedings of the Third Workshop on Mobile Security Technologies (MoST) (2014).

[0333] [31] Le Guan, Peng Liu, Xinyu Xing, Xinyang Ge, Shengzhi Zhang, Meng Yu, and Trent Jaeger. 2017. TrustShadow: Secure execution of unmodified applications with ARM trustzone. In Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services. ACM, 488??501.

[0334] [32] Zhichao Hua, Jinyu Gu, Yubin Xia, Haibo Chen, Binyu Zang, and Haibing Guan. 2017. vTZ: Virtualizing ARM trustzone. In In Proc. of the 26th USENIX Security Symposium.

[0335] [33] Kari Kostiainen, Jan-Erik Ekberg, N Asokan, and Aarne Rantala. 2009. Onboard credentials with open provisioning. In Proceedings of the 4th International Symposium on Information, Computer, and Communications Security. ACM, 104??115.

[0336] [34] Shih-Wei Li, John S Koh, and Jason Nieh. 2019. Protecting Cloud Virtual Machines from Hypervisor and Host Operating System Exploits. In 28th {USENIX} Security Symposium ({USENIX} Security 19). 1357??1374.

[0337] [35] Jonathan M McCune, Yanlin Li, Ning Qu, Zongwei Zhou, Anupam Datta, Virgil Gligor, and Adrian Perrig. 2010. TrustVisor: Efficient TCB reduction and attestation. In Security and Privacy (SP), 2010 IEEE Symposium on. IEEE, 143??158.

[0338] [36] Bryan D Payne, Martim Carbone, Monirul Sharif, and Wenke Lee. 2008. Lares: An architecture for secure active monitoring using virtualization. In Security and Privacy, 2008. SP 2008. IEEE Symposium on. IEEE, 233??247.

[0339] [37] Nuno Santos, Himanshu Raj, Stefan Saroiu, and Alec Wolman. 2014. Using ARM trustzone to build a trusted language runtime for mobile applications. In Proceedings of the 19th international conference on Architectural support for programming languages and operating systems. ACM, 67??80.

[0340] [38] David Sehr, Robert Muth, Cliff Biffle, Victor Khimenko, Egor Pasko, Karl Schimpf, Bennet Yee, and Brad Chen. 2010. Adapting Software Fault Isolation to Contemporary CPU Architectures.. In USENIX Security Symposium. 1??12.

[0341] [39] Arvind Seshadri, Mark Luk, Ning Qu, and Adrian Perrig. 2007. SecVisor: A tiny hypervisor to provide lifetime kernel code integrity for commodity OSes. In ACM SIGOPS Operating Systems Review, Vol. 41. ACM, 335??350.

[0342] [40] Monirul I Sharif, Wenke Lee, Weidong Cui, and Andrea Lanzi. 2009. Secure in-vm monitoring using hardware virtualization. In Proceedings of the 16th ACM conference on Computer and communications security. ACM, 477??487.

[0343] [41] Lei Shi, Yuming Wu, Yubin Xia, Nathan Dautenhahn, Haibo Chen, Binyu Zang, and Jinming Li. 2017. Deconstructing Xen.. In NDSS.

[0344] [42] He Sun, Kun Sun, Yuewu Wang, and Jiwu Jing. 2015. TrustOTP: Transforming Smartphones into Secure One-Time Password Tokens. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. ACM, 976??988.

[0345] [43] Zhi Wang and Xuxian Jiang. 2010. Hypersafe: A lightweight approach to provide lifetime hypervisor control-flow integrity. In 2010 IEEE Symposium on Security and Privacy. IEEE, 380??395.

[0346] [44] Zhi Wang, Xuxian Jiang, Weidong Cui, and Peng Ning. 2009. Countering kernel rootkits with lightweight hook protection. In Proceedings of the 16th ACM conference on Computer and communications security. ACM, 545??554.

[0347] [45] Bennet Yee, David Sehr, Gregory Dardyk, J Bradley Chen, Robert Muth, Tavis Ormandy, Shiki Okasaka, Neha Narula, and Nicholas Fullagar. 2009. Native client: A sandbox for portable, untrusted

x86 native code. In Security and Privacy, 2009 30th IEEE Symposium on. IEEE, 79??93.

[0348] [46] Ning Zhang, Kun Sun, Wenjing Lou, and Y Thomas Hou. 2016. CaSE: Cache-Assisted Secure Execution on ARM Processors. In Security and Privacy, 2016. SP 2016. IEEE Symposium on. IEEE.

[0349] [47] Yajin Zhou, Xiaoguang Wang, Yue Chen, and Zhi Wang. 2014. Armlock: Hardware-based fault isolation for arm. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. ACM, 558??569.

부호의 설명

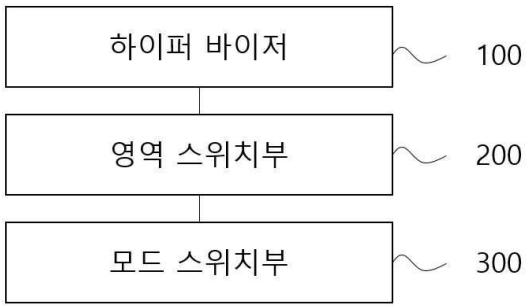
[0350] 100 : 하이퍼 바이저

200 : 영역 스위치부

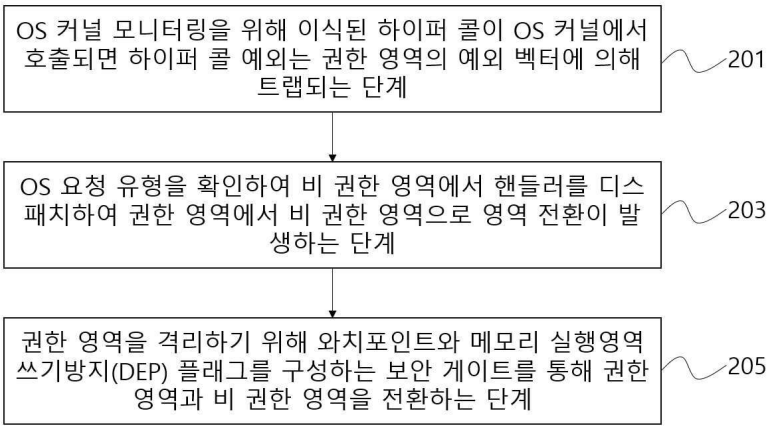
300 : 모드 스위치부

도면

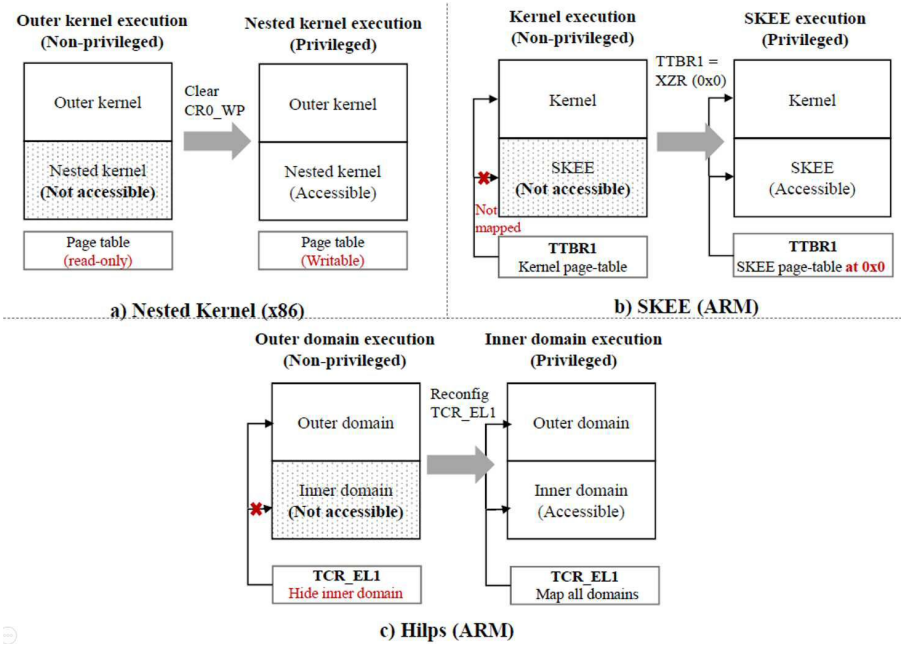
도면1



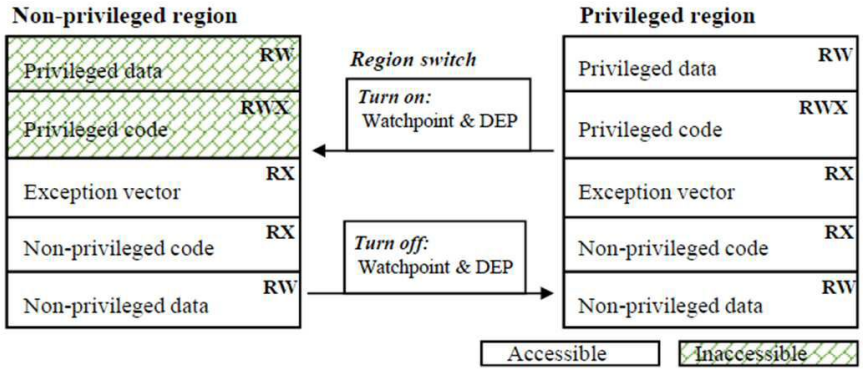
도면2



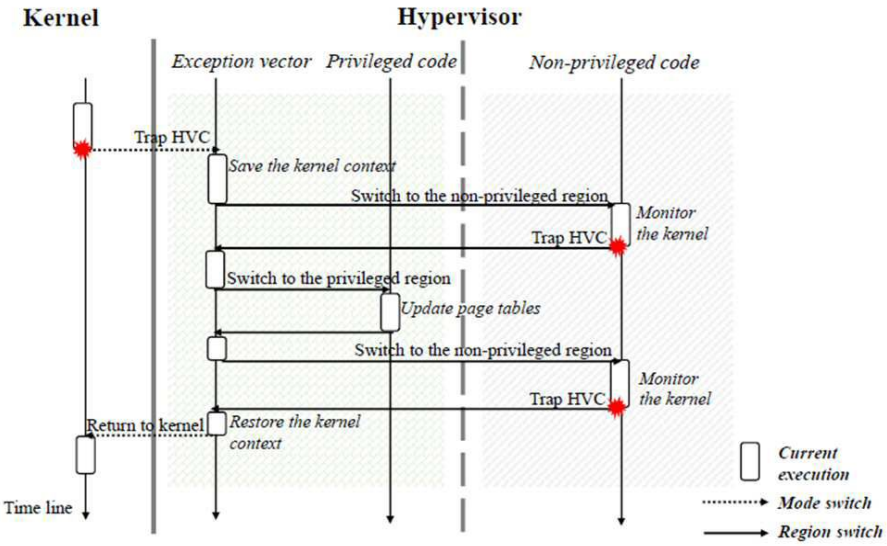
도면3


Outer kernel execution
(Non-privileged)
Nested kernel execution
(Privileged)
Outer kernel
Clear
CR0_WP
Outer kernel
Nested kernel
(Not accessible)
Nested kernel
(Accessible)
Page table
(read-only)
Page table
(Writable)
a) Nested Kernel (x86)
Kernel execution
(Non-privileged)
SKEE execution
(Privileged)
TTBR1 =
XZR (0x0)
Kernel
SKEE
(Not accessible)
Not
mapped
TTBR1
Kernel page-table
Kernel
SKEE
(Accessible)
TTBR1
SKEE page-table at 0x0
b) SKEE (ARM)
Outer domain execution
(Non-privileged)
Inner domain execution
(Privileged)
Reconfig
TCR_EL1
Outer domain
Inner domain
(Not accessible)
TCR_EL1
Hide inner domain
Outer domain
Inner domain
(Accessible)
TCR_EL1
Map all domains
c) Hilps (ARM)

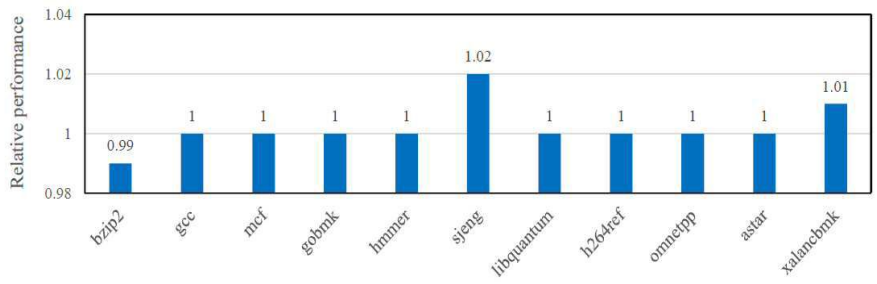
도면4


Non-privileged region
Privileged region
Privileged data
RW
Privileged code
RWX
Exception vector
RX
Non-privileged code
RX
Non-privileged data
RW
Region switch
Turn on:
Watchpoint & DEP
Turn off:
Watchpoint & DEP
Privileged data
RW
Privileged code
RWX
Exception vector
RX
Non-privileged code
RX
Non-privileged data
RW
Accessible
Inaccessible

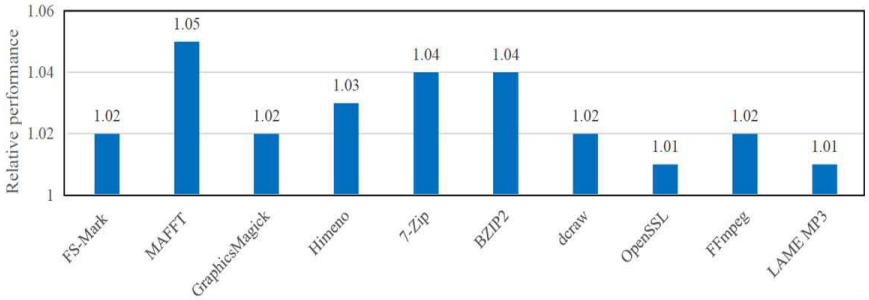
도면5



도면6



도면7



Avr	3.17	27.74	40	42.37	2149	85	1035	80.7	114	475
σ	0.584	1.114	1	2.162	0.468	3.266	1.802	1.133	1	0.346