1주차(hardware-assisted instruction-level isolation on ARM and its applications)

ARM – 스마트폰 etc… 보안성 향상방법!

컴퓨터 시스템을 안전하게! 어떤 시큐리티 메커니즘?

Security mechanism(software- only)

1. 장점 : 적용용이
2. 단점 : performance overhead 심각 -> 그래서 시큐리티 gurantee를 줄인다!

하드웨어의 도움을 받자!

Application 보안성, operating system 보안성, ARM Trustzone

Isolation : 공격을 당할 수 있는 code size 줄인다!(작은 sw entity를 따로 띄어놓아 공격줄임)

Isolation 효과 : 원래는 L+S개의 공격이지만, isolation 하면 small sw entity는 s개의 공격 받음

Domain-level Isolation종류:(1)Intel SGX: enclave, (2)ARM trustzone: normal world/secure world분리

Domain-level isolation : un-authorized code는 sensitive data접근금지

Domain-level isolation 장점 : 드문드문 access되는 곳에 유용! access빈번하면(heap object의 function pointers, return addresses in the stack, etc…)

Shadow stack 이용! : shadow stack에는 return address만 넣는다!

Instruction-level isolation : unauthorized instruction은 sensitive data 접근금지

Instruction-level isolation 실행 : (1)segment(어떤 명령어는 ds에 있는곳은 access불가!)

(2) SFI : check 명령!

(3) information hiding : sensitive data를 임의의 장소에 배치시킨 후, 명령어에 레지스터값넣기

ARM 구조에서의 instruction-level isolation : Load 와 Store unprivileged(LSU)

Isolating kernel Data on latest Cortex-A processors :

커널 보안의 중요성 : 커널은 크고 복잡한 코드를 가짐 -> 메모리 corruption 취약성

Isolation을 이용해서 커널을 보호한다. : page table switching -> penalty: TLB management configure system registers -> privileged Access Never[PAN] bit, LSU 사용

메커니즘 강화 : system control 모듈

Cortex-m processors에서 execute-only memory 실행

Execute – only memory [XOM]

보통 메모리는 read,write,execute지만, 이 메모리는 execute만함

목적 : 알려진 공격 좌절시킴. : intellectual properties 막음, CRA gadgets 숨김

uXOM : compiler-based technique to enforce the execute-only memory permission on code memory.

Basic design 그림보기

Xom의 실제 적용 challenges 발생 :

1. ppb,Exclusive memory instruction 에는 privilege만

* unconvertible memory instructions을 찾음 (unprivileged 바꿈 x)

1. 공격자가 control flow를 바꿔서 normal instruction을 직접 할 수 있으면 문제
2. Return address를 조작해버리면 큰일남.

* 2,3해결책 : 임시 레지스터 사용, 하지만 레지스터를 하나 더 사용한다는 부작용
* Stack pointer를 사용해버린다!

1. 데이터 조작

* Atomic verification Technique: Verification routine을 추가하여, 그리고 interrupt 제어하는 명령어 추가

1. instruction시작 비트를 바꿔버림! -> unintended instructions. Unprivileged instruction을 싷행해버릴수도있음

* handling unintended instructions (original code사용 x)