Instruction-Level Data Isolation for the Kernel on ARM

ARM系统上指令级别的内核数据隔离

DAC 2017

刘文清

目录
CONTENTS

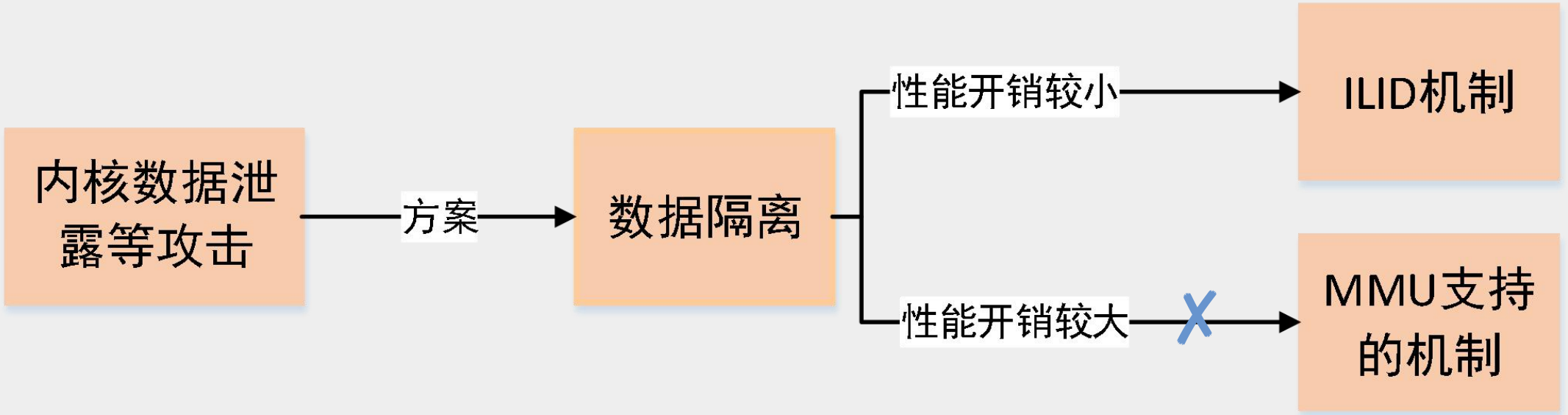
01

Introduction

Introduction

- 许多对内核的攻击导致内核数据的泄露，为了保护内核数据不被恶意篡改等，数据隔离被提出。目的是创建安全机制保护系统敏感数据。
- 数据隔离已成为一种可以通过为敏感内核数据提供强大保护来解决该问题的关键技术。 但是，现有数据隔离机制都会产生不可忽视的性能开销。
- 为了有效地在基于ARM的机器上执行数据隔离，提出基于硬件指令的数据隔离机制（ILDI）。目的是减小系统运行时的性能开销。

Introduction



Introduction

- 数据隔离一般思路（基于MMU）
  - 思想
    - 创建与内核完全隔离的安全区域
    - 仅允许持有专有权限的特定代码访问安全区域来管理敏感数据
  - 缺点
    - 性能开销很大
    - 因每次访问安全区域都需MMU配置（页表切换）和TLB维护的成本

Introduction

- 构造ILDI的基本思路
  - 数据隔离
    - PAN位：可协助实现创建内核的内存指令不可访问的安全区域
    - LSU(Load and Store Unprivileged Instruction)
    - 访问安全区域的唯一方法是通过LSU指令给出的，LSU指令用于访问安全区域
  - SC模块
    - 保证数据隔离安全运行
    - 对PAN和LSU涉及的page tables 和 regs进行保护

解决的问题

- 解决的问题
  - 实现性能开销相对较小的数据隔离方案
  - 能够防御代码重利用攻击、代码注入攻击

02

Related Work

Related Work

- Software Fault Isolation(软件故障隔离)
  - 思想：阻止非认证模块中的代码或者数据扩散到整个内核
  - 方法：阻止非认证的内存指令访问安全区域
  - 缺点：性能随着代码量的增加而降低
- MMU-supported
  - 思想：创建不同访问权限的两个页表，分别访问安全区和普通区，系统会适时让认证代码访问安全区域。比如：TTBR0和TTBR1
  - 缺点：性能开销大，配置切换可以改变区域的保护状态，导致性能降低。（代码段和数据段的属性变化等）

03

Background Theeat

背景知识

PAN

LSU

地址翻译

Background

- PAN (Privileged Access Never)
  - 即“特权执行从不”。阻止特权代码（内核）访问非特权区域（用户区域）。
  - CPSR寄存器可以设置PAN位。SCTLR寄存器可以设置SPAN位。

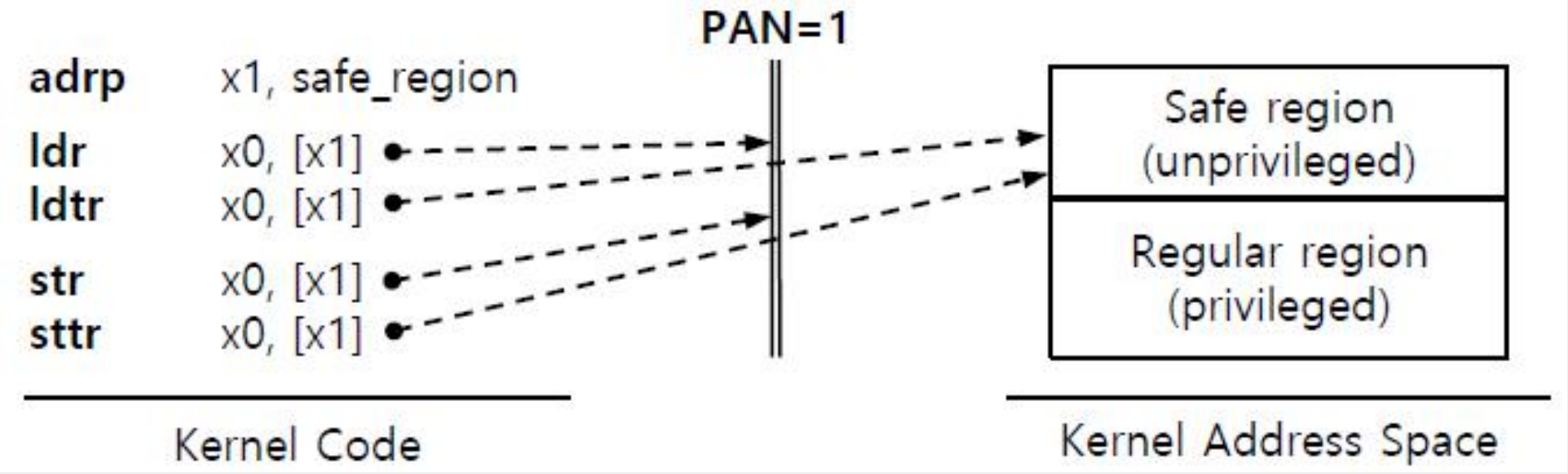
- LSU(Load and Store Unprivileged Instruction)
  - EL1层（特权层 ）和EL0层（非特权层）执行情况一样，访问内存时依赖EL0层的访问权限

Table 1: Load and Store Unprivileged Instructions

Load Unprivileged Insts.	Store Unprivileged Insts.
LDTR, LDTRB, LDTRSB, LDTRH, LDTRSH, LDTRSW	STTR, STTRB, STTRH

Background

- ILDI机制
  - 只有LSU指令被允许访问unprivileged region



Background

- Address Translation
  - 两个翻译页表基寄存器
  - TTBR0 虚拟地址低部分（用户空间）
  - TTBR1 虚拟地址高部分（内核空间）

Threat Model

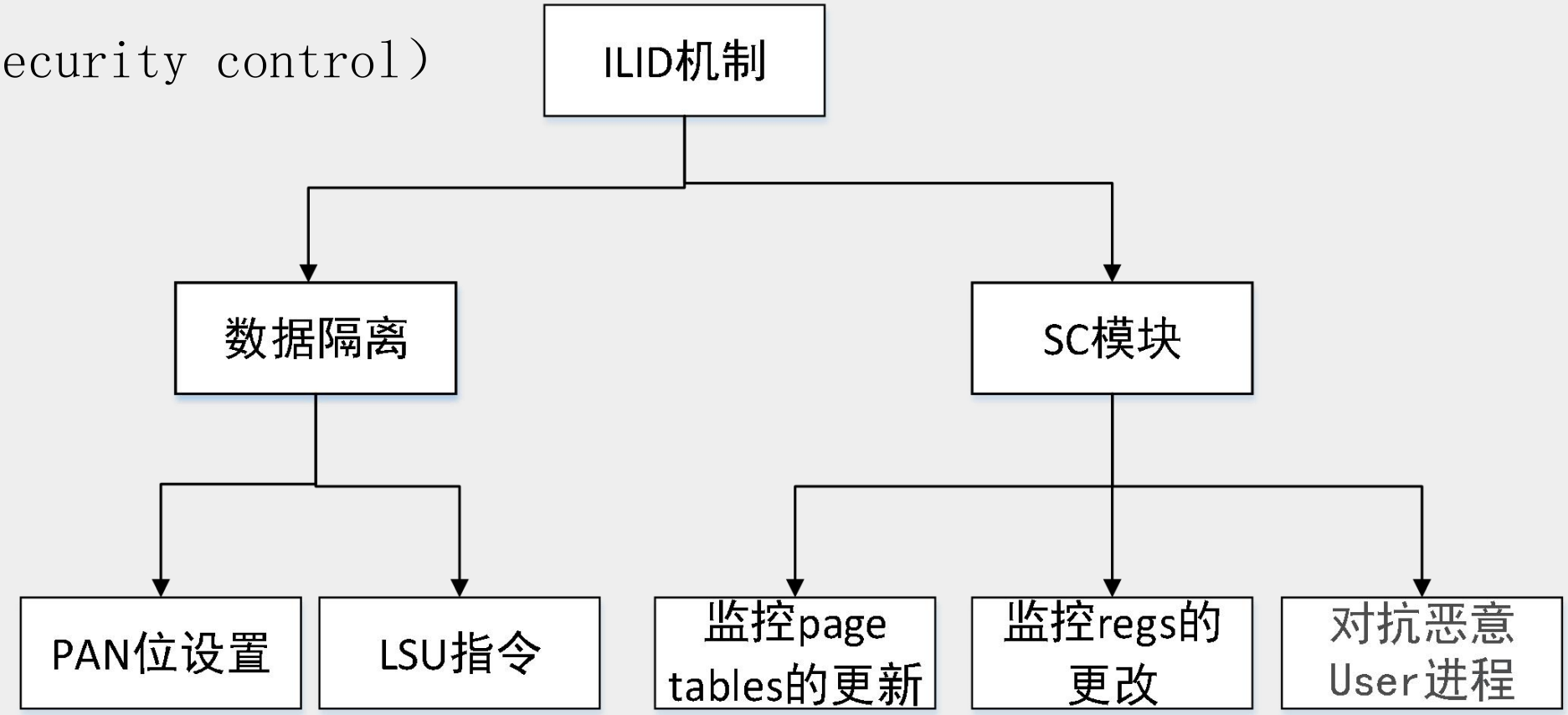
- 威胁模型
  - 攻击者能够利用OS的内存泄漏漏洞（memory corruption vulnerability），恶意地进行读写操作，从而污染内核
- 安全保证
  - ILDI机制创建安全区域，允许内核通过固定的指令访问安全区域
- 假设条件
  - 启动是安全的
  - DoS攻击、侧信道攻击、基于硬件的攻击（冷启动攻击、RowHammer攻击）等不在考虑范围之内

04

Design Implement

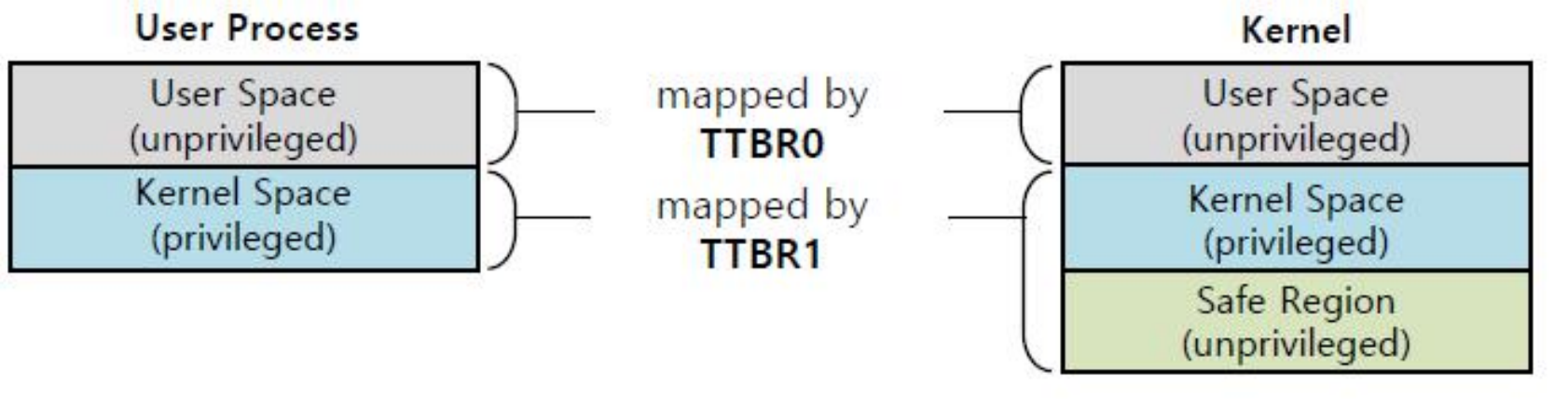
Design

- ILDI组成
  - 数据隔离
  - SC模块（security control）



Design

- 数据隔离
  - 目的
    - 使用PAN位和LSU指令实现对safe region的访问，以减小性能开销
  - 实现
    - 在内核创建safe region（非特权级别），设置PAN为1，于是kernel无法访问处于unprivileged的safe region，只有使用LSU指令进行load和store 操作



Design

- SC模块（security control）
  - 设计
    - hyp mode层，隔离于kernel，kernel通过固定的entry exit 门与SC模块进行交互，控制被污染kernel对page tables和regs的恶意访问
  - 功能
    - 对数据隔离的安全控制
    - 对page tables 的访问，更新操作放到SC中
    - 对regs (SCTLR、TTBR)修改操作的监控放到SC中
    - 对抗恶意User Process

TTBR ： 存放safe region的page table 的基地址
SCTLR： 设置PAN位

SC模块

- Regs的控制
  - 目的
    - 保护PAN相关的寄存器不被恶意操作
    - PAN位在系统运行过程中必须保持为1，safe region才能一直为unprivileged状态，需要控制能操作PAN位的指令或者寄存器。
  - 操作
    - 移除MSR指令（可清除PAN位）
    - SC模块监控SCTLR reg （控制PAN位）的更改操作

SC模块

- Page Tables的控制
  - 操作
    - SC模块监控对TTBR和MMU的更改操作
  - 目的
    - safe region的地址映映射在page tables 中，监控page tables的相关操作
    - 检测和阻止双映射(double mapping)，防止不同访问权限的虚拟地址映射相同的物理地址（safe region 必须保持unprivileged）

SC模块

- 对抗恶意User Process
  - 目的
    - safe region持续为unprivileged状态，为了防止恶意User进程访问safe region
  - 措施
    - 创建2套页表，页表1不包含safe region 的地址映射。页表2含所有映射。
    - User process 运行，开启页表1，否则开启页表2.
    - Safe region 的地址映射残留在TLB中，在ASID的协助下刷新TLB，减小性能开销

05

Security Analysis

Security analysis

- 代码注入攻击
  - 攻击
    - 注入LSU指令或者特权代码来操作系统寄存器
    - DMA攻击
  - 对策
    - 使用W⊕X机制，对page tables 和 regs监控，监控对DAM的操作
- 代码重使用攻击
  - 攻击
    - 更改内核的控制流，用精心构造的参数重新执行LSU指令
  - 对策
    - 代码指针分离、影子栈

06

Evaluation

Evaluation

- MMU-supported ILDI对比
  - Primitive Operation
    - 主要操作：与safe region 的加载保存敏感数据（Load Store data）
  - Results
    - 本方案的性能开销更小
    - MMU-supported 方案的速度是ILDI机制的1/188

Table 2: CPU Cycles of Primitive Operations

	Load	Store
Normal Memory Insts.	1	1
Our Work	1	1
MMU-supported	188	188

Evaluation

- 实验环境
  - 多功能高速V2M-Juno r1平台(Cortex-A57 1.15 GHz双核处理器和Cortex-A53 650 MHz四核处理器，2G DRAM)
- 测试
  - Linaro Android版本15.06（Android版本为5.1.1 4，Linux内核3.10）
  - Benchmarks for base system
    - Lmbench(for kernel)
    - GeekBench 3.4.1、Basemark OS II 、Vllamo 3.2、Antutu 6.0.1(for system)

Evaluation Results

Table 4: Performance Overhead of Synthetic Benchmarks in comparison with Native

Test	Base	Our Work (w/ CFI)	MMU-sup. (w/ CFI)
GeekBench 3.4.1	0.0 %	0.8 %	2.2 %
Basemark OS II	0.0 %	0.1 %	15.0 %
Vellamo 3.2			
browser	0.2 %	1.5 %	4.6 %
metal	0.1 %	0.2 %	0.7 %
Antutu 6.0.1	0.2 %	1.2 %	3.0 %

- 由结果知，SC模块监控page tables 和 regs ，对系统造成的性能开销较低