Toward Fine-Grain and Scalable Hardware Isolation Primitives

By Jules Drean, Srini Devadas, Daniel Sanchez and Mengjia Yan

INTRODUCTION

Current hardware isolation primitives do not protect against micro-architectural side-channels efficiently. They are coarse grain, not scalable and not dynamically adaptable.

We propose to :

- Create new hardware isolation primitives
- Defining and characterizing isolation scenarios
- Build a framework to tune the isolation primitives to different isolation scenarios
- Real-world case study with Google Chromium

I. BACKGROUND

What are micro-architectural side-channels ?

Side Channels that exploit shared hardware resources and micro-architectural state to exfiltrate secret information.

What is strong isolation ?

Two Security Domains (**SDs**) are *Strongly Isolated* if the timing of the micro-architectural events of one is independent from the timing of the micro-architectural event of the other.

Insight: Isolation Must be Enforced For **Every Micro-Architectural State**

Memory, Caches, Branch Predictor, Memory Buses...

Limitations of Current Isolation Primitives:

a) Do not cover every micro-architectural states
b) Too coarse grained
c) Not scalable
d) Not dynamically adaptable

What we need: *Address all the limitations above*

For example:
- Support more than a thousand security domains
- Support high variety of resources and security domains

II. PROPOSAL

- Build new hardware isolation primitives.
- Define and characterize isolation scenario for each type of micro-architectural structures and security domain transition patterns
- Design a unified framework to tune and dynamically manage isolation strategies and resources

III. TOOLBOX TO BUILD NEW HARDWARE ISOLATION PRIMITIVES

Partitioning

Allocating part of the ressource for a given security domain

Sanitizing

The ressource micro-architecture is reset to a public state

Invisibility

The micro-architectural state is not modified, making the security domain invisible to others

State Restoration

The attacker's micro-architectural state is restored when context-switching back, making the victim's micro-architectural state not observable.

Isolation strategy might need to be combined to enforce security, performance or scalability.

Example: While in State Restoration mode, if the storage space required to restore the state exceed a given side, a state sanitation is programmed for the next context switch.

IV. DIVERSITY IN RESOURCES AND SECURITY DOMAINS: DEFINING ISOLATION SCENARIOS

The optimal isolation strategy depends on:

Ressource Type	Transient or Stateful / Size / Part of the Memory Hierarchy
Sharing Type	Sequential or Simultaneous / Private or Not
SD Type / Interaction	Long or Short-Lasting / Long or Short Interruptions / **Type of Interactions**

Security Domains May Interact Using Different Patterns

Application Partitioning

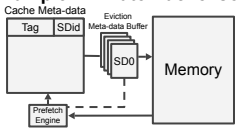
Sensitive Function Isolation

Short Function Calls

Reusable Execution Environments

How to characterize interactions between Security Domains for a given shared resource?

Example : Private Cache Isolation Using State Restoration



If recording state modifications is too expensive -> switch to a sanitize strategy

Switch from SD_0 to the outside
Record evicted addresses from SD_0 in the allocated Eviction Metadata Buffer
If the buffer overflows is too small, program a cache flush for next re-entry

Switch from the outside to SD_A
If cache flush programmed -> flush
Else, restore cache state by prefetching evicted addresses back

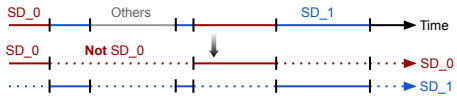
V. A REAL WORLD CASE STUDY: GOOGLE CHROMIUM

Experiment Description :

Instrument Chromium to observe execution traces while launching the program and loading one page.

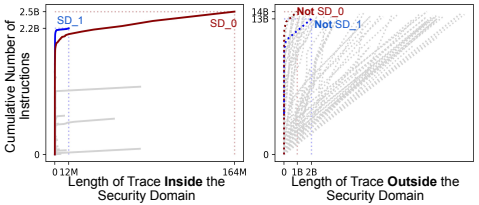
We define Security Domains using thread boundaries (loading one page creates ~72 threads)

We record the length of traces inside and outside of each Security Domain



Then, for each SD, we plot the

Cumulative Number of Instruction as a Function of the Trace Length



Result Description : *Security Domains Have Different Behaviours*

We highlighted in color the two hottest security domains.

	SD_0	SD_1
Number Instructions	HIGH	HIGH
Context Switch Frequency	LOW	MEDIUM

We can tune isolation primitives for different security domains based on their behaviors:

SD_0 ➡cache partition
SD_1 ➡?
Grey SDs ➡invisibility / sanitizing