



Security Need a New Hardware-Software Contract

Gernot Heiser | gernot.heiser@data61.csiro.au | @GernotHeiser

- Presentation to RISC-V Privspec TC, November 2019

<https://trustworthy.systems>



UNSW
SYDNEY



Threats



= +



Speculation

An “unknown unknown” until recently

A “known unknown” for decades



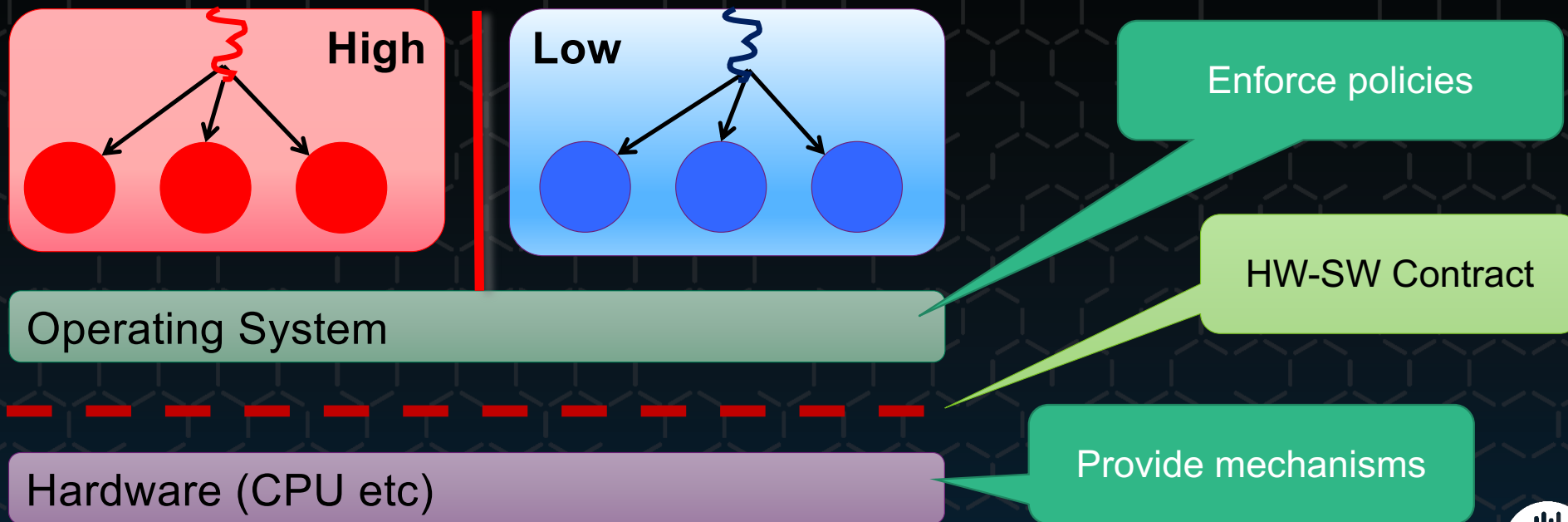
Microarchitectural
Timing Channel

Spatial Isolation: A Solved Problem

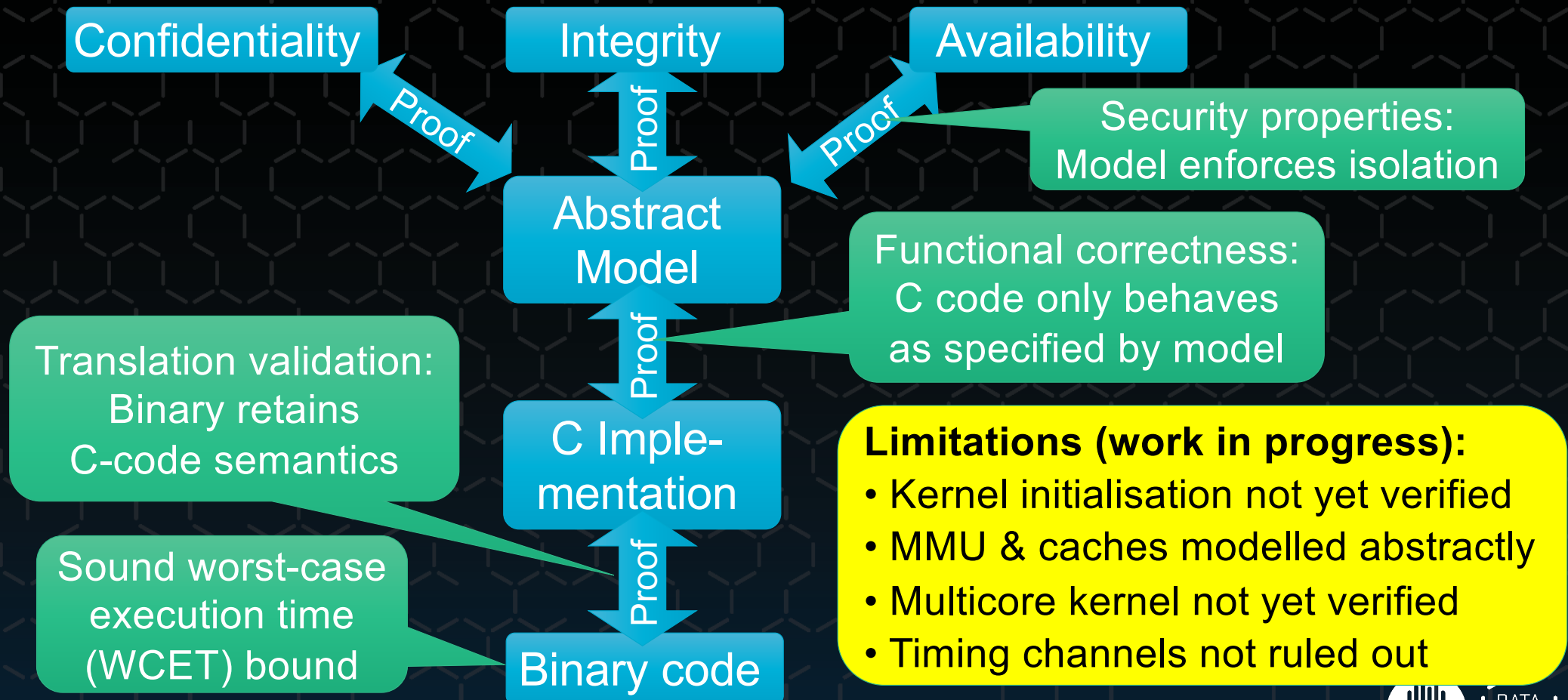


Enforcing Security: The OS's Job

Security enforcement must be **mandatory**, i.e. not dependent on application/user cooperation!



Proved OS-Enforced Spatial Isolation



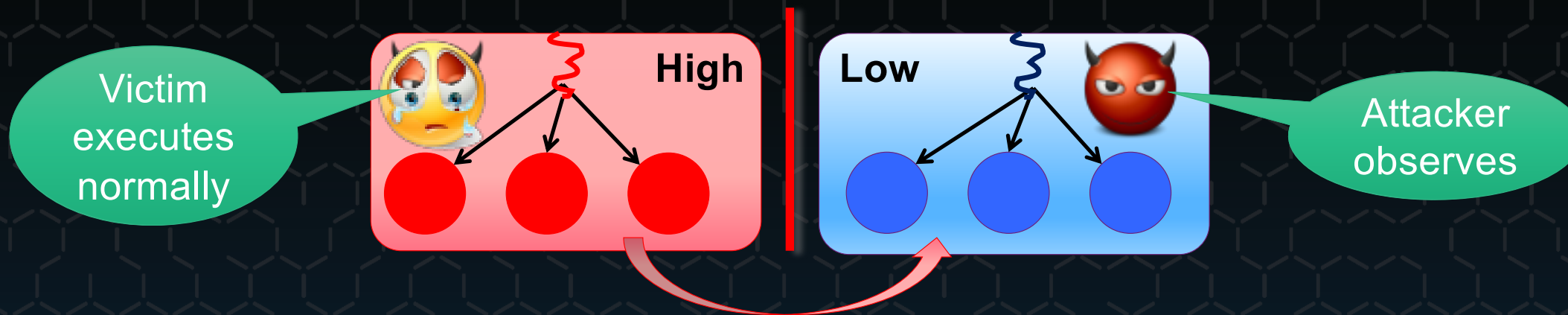
What Are Timing Channels?

Timing Channels

Information leakage through timing of events

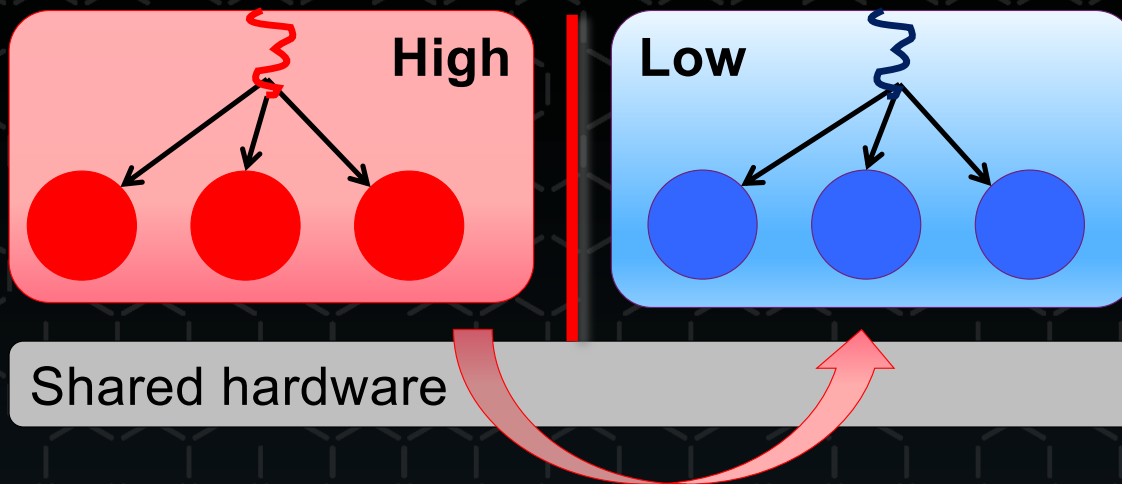
- Typically by observing response latencies or own execution speed

Covert channel: Information flow that bypasses the security policy



Side channel: Covert channel exploitable without insider help

Cause: Competition for Shared HW Resources

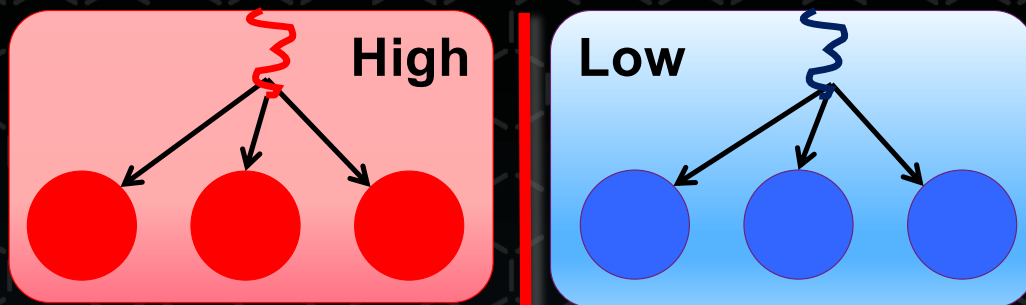


Affect execution speed

- Inter-process interference
- Competing access to micro-architectural features
- **Hidden by the HW-SW contract!**

What Is Needed?

Confidentiality Needs **Time Protection**

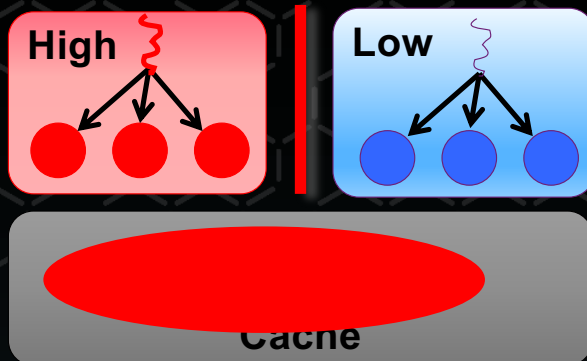


Traditionally OSes enforce security by *memory protection*, i.e. enforcing spatial isolation

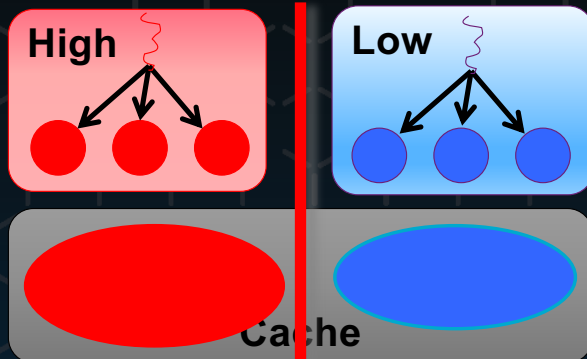
Time protection: A collection of *OS mechanisms* which collectively *prevent interference* between security domains that make execution speed in one domain dependent on the activities of another.

[Ge et al. EuroSys'19]

Time Protection: Partition Hardware

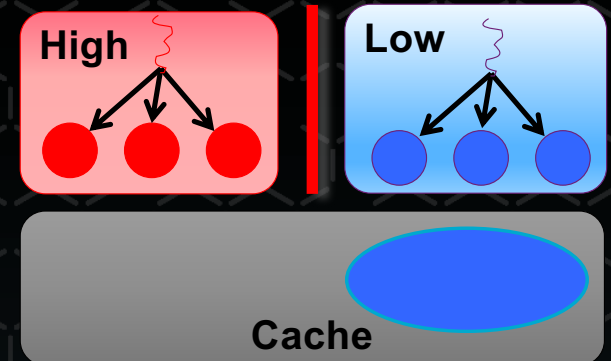


Spatially partition



Temporally partition

Flush



Need both!

Cannot spatially partition on-core caches (L1, TLB, branch predictor, pre-fetchers)

- virtually-indexed
- OS cannot control

Flushing useless for concurrent access

- HW threads
- cores

Requirements for Time Protection

Off-core
state &
stateless HW

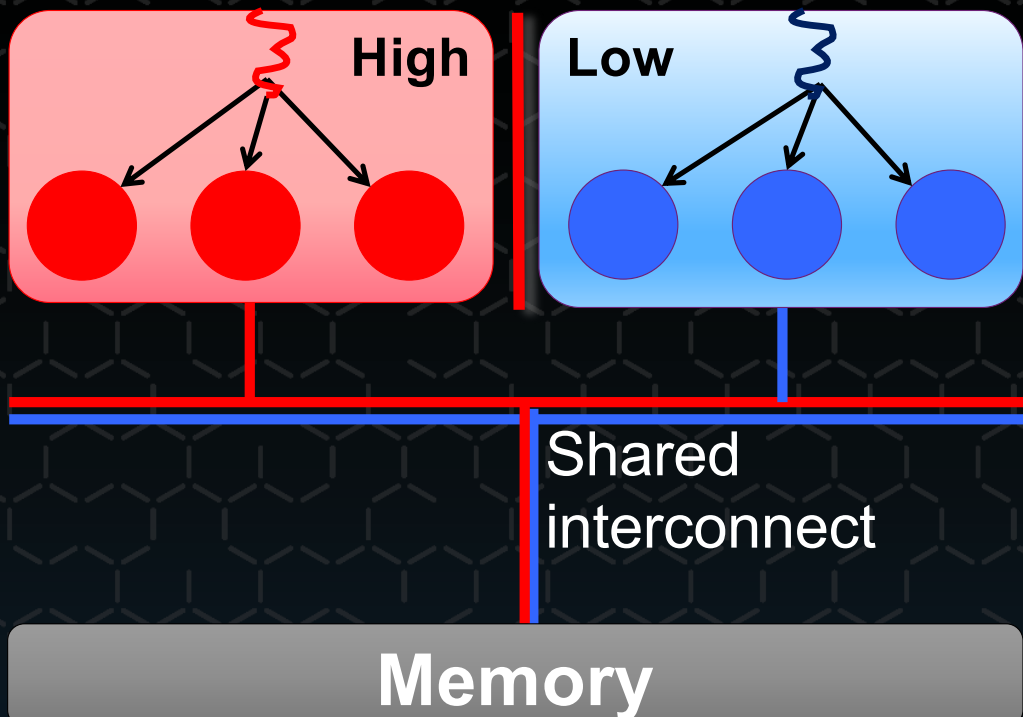
Timing channels can be closed *iff* the OS can

- (spatially) partition or
- reset

all shared hardware

On-core
state

Sharing 1: Stateless Interconnect



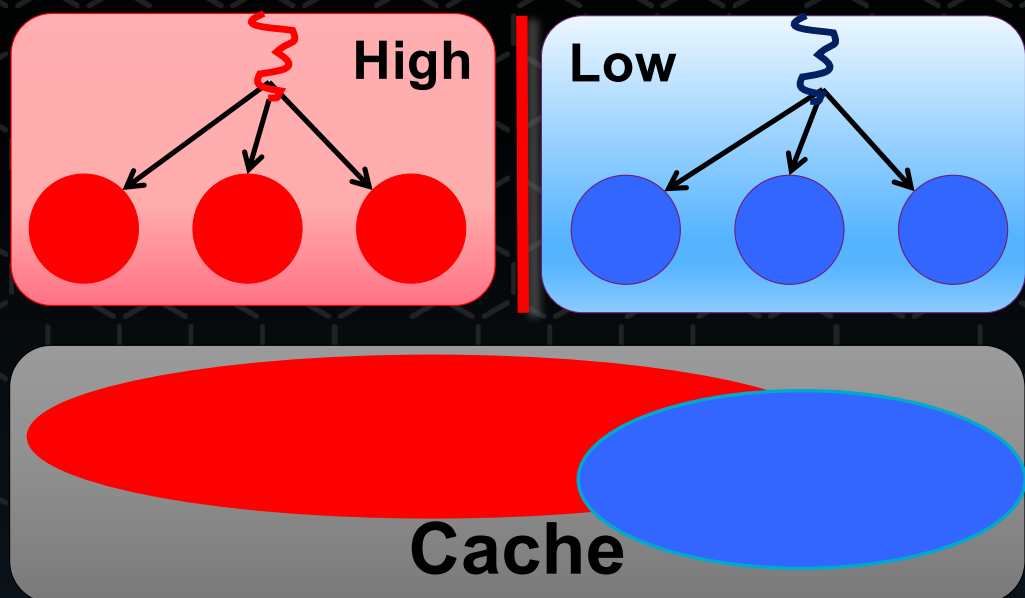
H/W is *bandwidth-limited*

- Interference during concurrent access
- Generally reveals no data or addresses
- Must encode info into access patterns
- *Only usable as covert channel, not side channel*

**No effective defence
with present hardware!**

Outside
Privspec TC
scope

Sharing 2: Stateful Hardware



HW is *capacity-limited*

- Interference during
 - concurrent access
 - time-shared access
- Collisions reveal addresses
- *Usable as side channel*

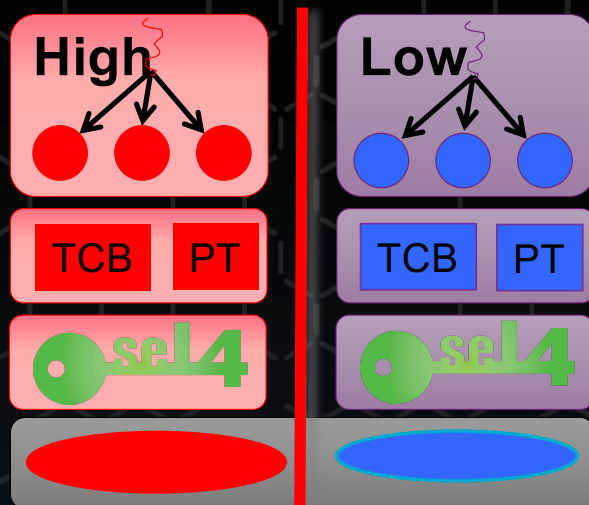
Solvable problem –
focus of this work

Any state-holding microarchitectural feature:

- cache, branch predictor, pre-fetcher state machine

Implementing Time Protection on Stateful Hardware

Spatial Partitioning: Cache Colouring



- Partitions get frames of disjoint colours
- seL4: userland supplies kernel memory
⇒ colouring userland colours dynamic kernel memory
- Per-partition kernel image to colour kernel
[Ge et al. EuroSys'19]



Temporal Partitioning: Flush on Switch

Must remove any history dependence!

1. $T_0 = \text{current_time}()$
2. Switch user context
3. Flush on-core state
4. Touch all shared data needed for return
5. $\text{while } (T_0 + \text{WCET} < \text{current_time}()) ;$
6. Reprogram timer
7. return

Latency depends on prior execution!

Time padding to Remove dependency

Ensure deterministic execution

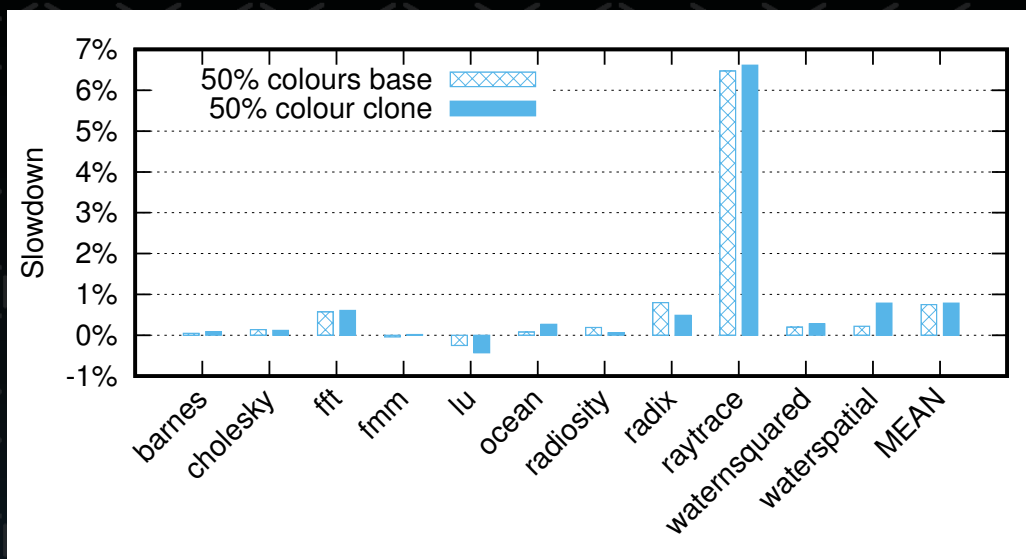
Cost of Reset



- Flushing **on-core state** is not a performance issue:
 - no cost when not used
 - direct flush cost for dirty L1-D in the order of $1\mu\text{s}$
 - direct flush cost for everything else in the order of 100 cycles
 - indirect cost is negligible, if used on security-partition switch
 - eg VM switch, 10–100 Hz rate
 - no hot data in cache after other partition's execution
- **Hardware support (eg targeted L1 flush) is essential!**



Performance Impact of Colouring



- Overhead mostly low
- Not evaluated is cost of not using super pages [Ge et al., EuroSys'19]

Architecture	x86	Arm
Mean slowdown	3.4%	1.1%

Arch	seL4 clone	Linux fork+exec
x86	79 μ s	257 μ s
Arm	608 μ s	4,300 μ s

Reality Check: Flushing On-Core State



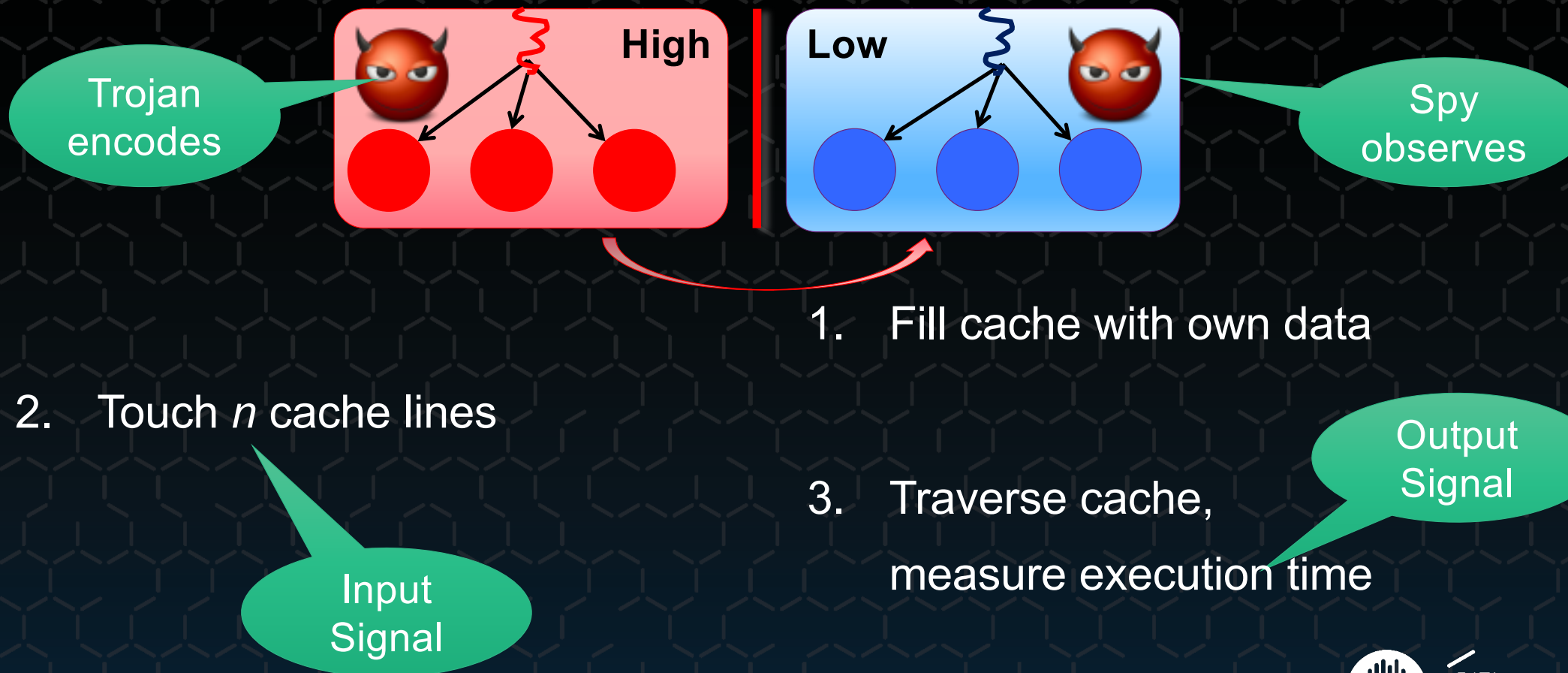
Evaluating Intra-Core Channels



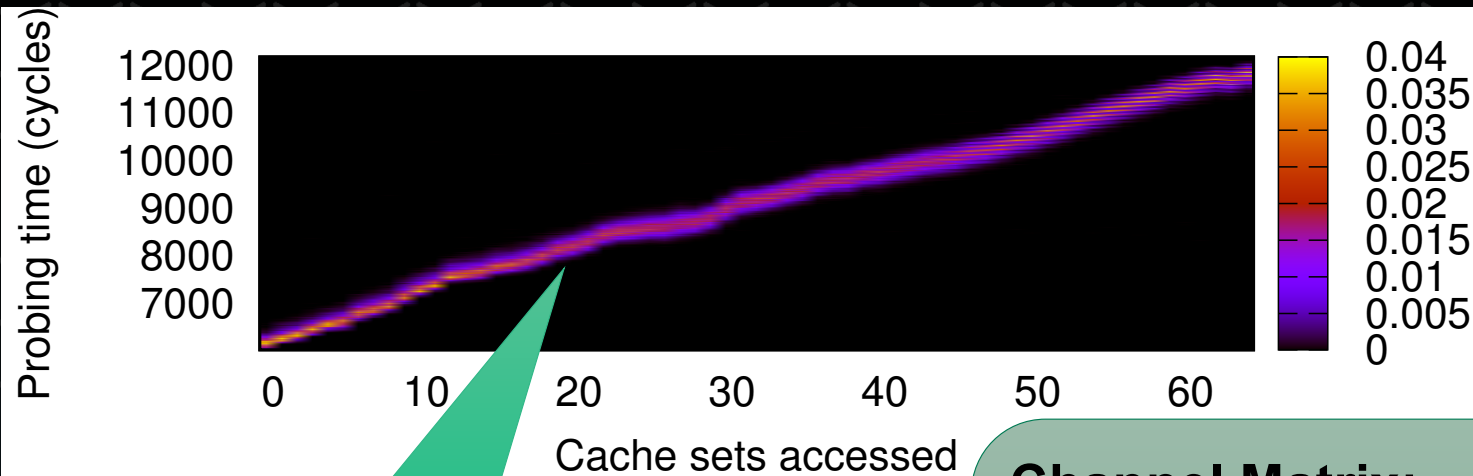
Mitigation on Intel and Arm processors:

- Disable data prefetcher (just to be sure)
- On context switch, perform all architected flush operations:
 - Intel: wbinvd + invpcid (no targeted L1-cache flush supported!)
 - Arm: DCCISW + ICIALLU + TLBIALl + BPIALL

Methodology: Prime and Probe



Methodology: Channel Matrix



Horizontal
variation indicates
channel

Channel Matrix:

- Conditional probability of observing time, t , given input, n .
- Represented as heat map:
 - bright = high probability

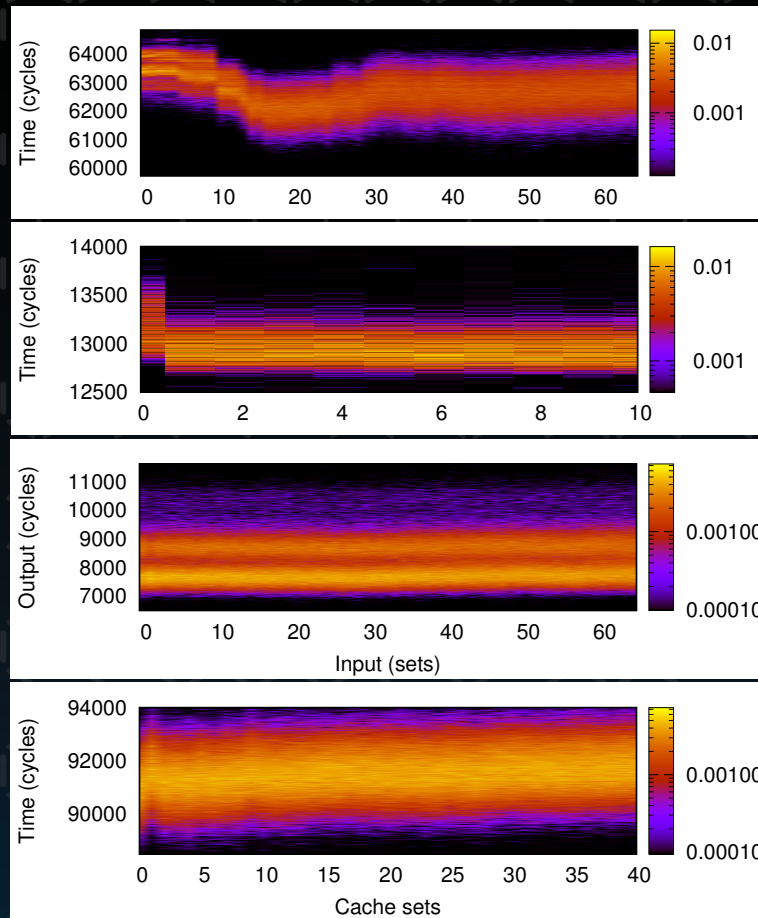
I-Cache Channel With Full State Flush

CHANNEL!

CHANNEL!

No evidence
of channel

SMALL CHANNEL!



Intel Sandy Bridge

Intel Haswell

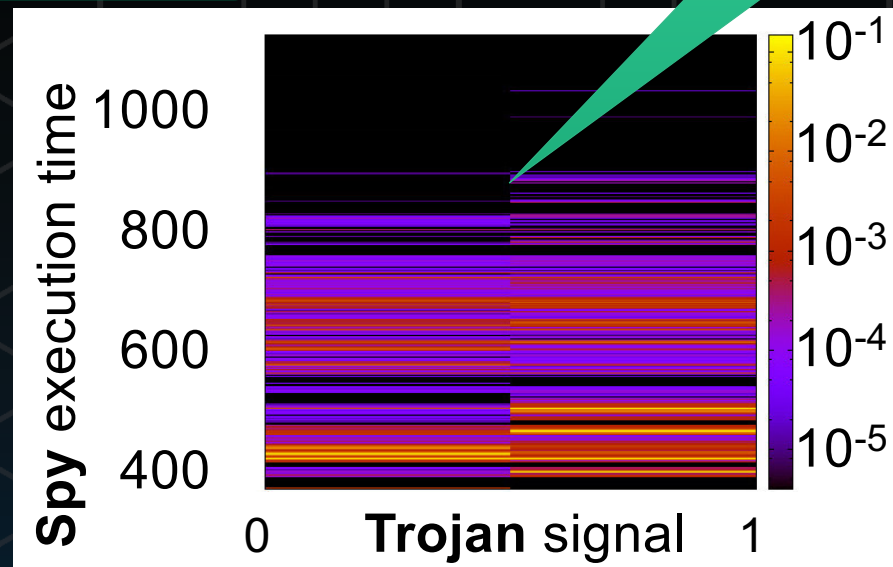
Intel Skylake

HiSilicon A53

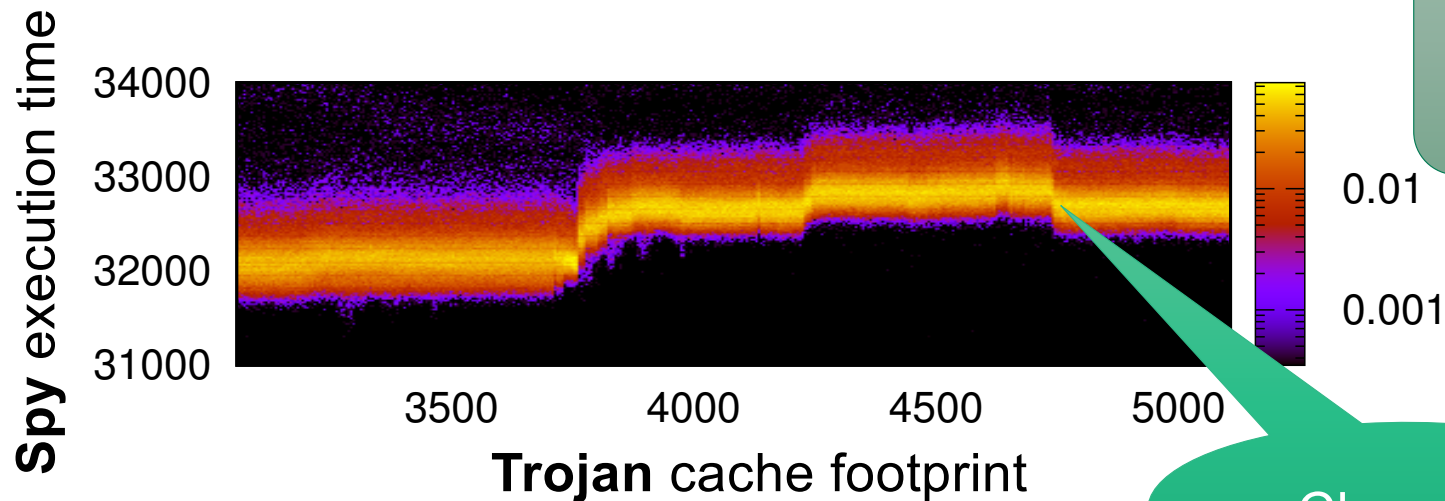
HiSilicon A53 Branch History Buffer

Branch history buffer (BHB)

- One-bit channel
- All reset operations applied



Intel Haswell Branch Target Buffer



Branch target buffer

- All reset operations applied

**Found residual channels
in all recent Intel and ARM
processors examined!**

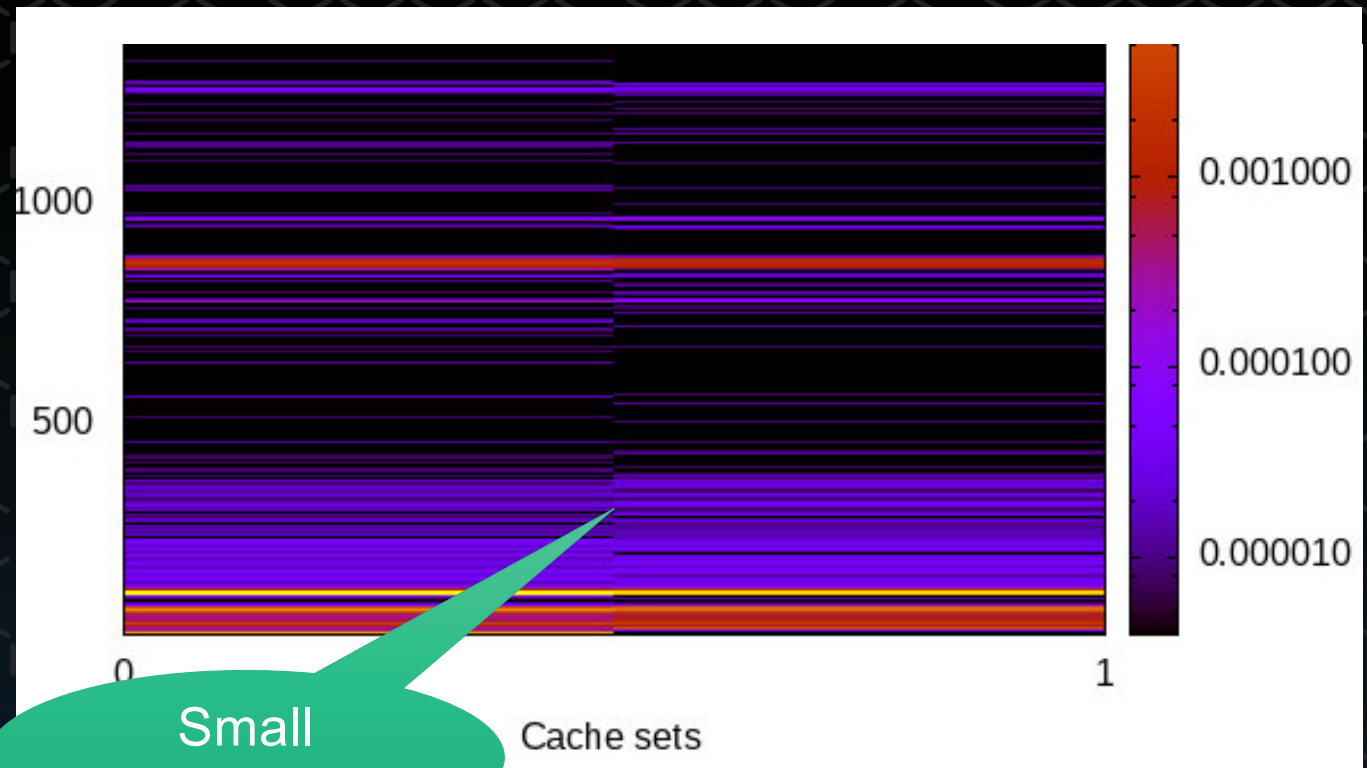
Channel!

Intel Spectre Defences

Intel added *indirect branch control* (IBC) feature, which closes most channels, but...

Intel Skylake
Branch history buffer

**Also residual state
in pre-fetchers**



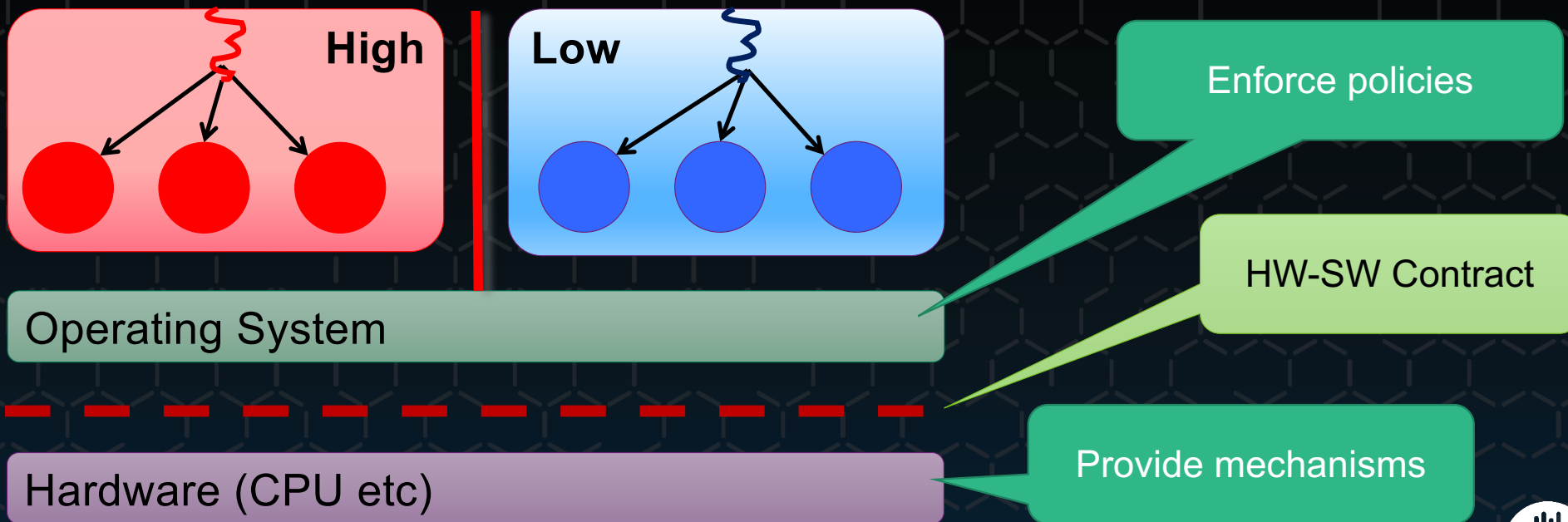
<https://ts.data61.csiro.au/projects/TS/timingchannels/arch-mitigation.pml>

Security: A HW-SW Codesign Issue



Remember: Security Enforcement

Security enforcement must be **mandatory**, i.e. not dependent on application/user cooperation!



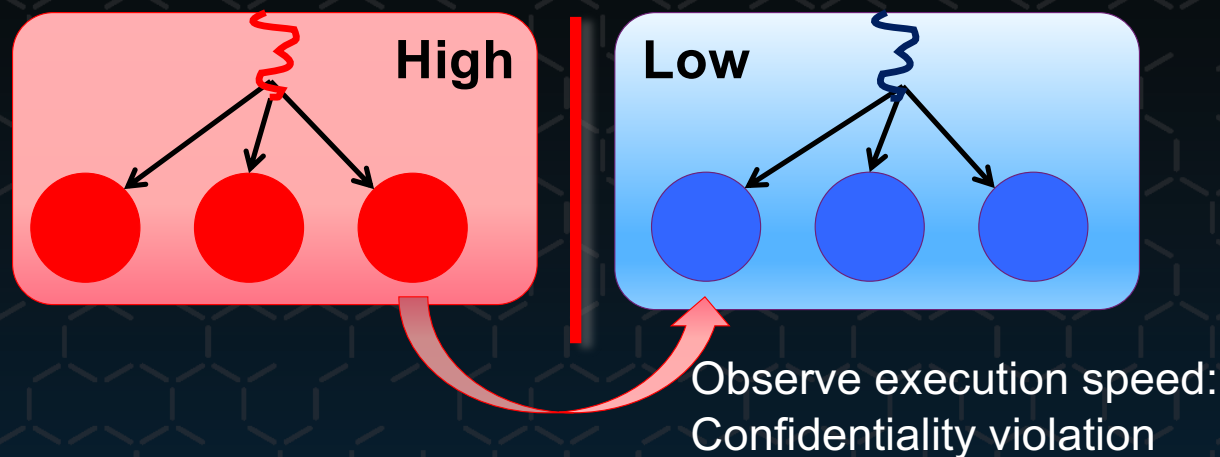
Why Hardware Cannot Do Security Alone

- Security policies are high-level
 - Course-grain: “applications” are sets of cooperating processes
- Hardware mechanisms are fine-grain: instructions, pages, address spaces
 - Much semantics lost in mapping to hardware level
- Security policies are complex: “Can A talk to B?” is too simple
 - maybe one-way communication is allowed
 - maybe communication is allowed under certain conditions
 - maybe low-bandwidth leakage doesn’t matter
 - maybe secrets only matter for a short time
 - maybe only subset of {confidentiality, integrity, availability} is important

Why the ISA is an Insufficient Contract

- The ISA is a purely operational contract
 - Sufficient for ensuring functional correctness
 - Insufficient for ensuring confidentiality or availability

The ISA intentionally abstracts time away



New HW/SW Contract: aISA

Augmented ISA supporting time protection

Security Standing
Committee agrees



For all shared microarchitectural resources:

1. Resource must be spatially partitionable or flushable
2. Concurrently shared resources must be spatially partitioned
3. Resource accessed solely by virtual address must be flushed and not concurrently accessed
 - Implies cannot share HW threads across security domains!
4. Mechanisms must be sufficiently specified for OS to partition or reset
5. Mechanisms must be constant time, or of specified, bounded latency
6. Desirable: OS should know if resettable state is derived from data, instructions, data addresses or instruction addresses
7. Desirable: Flush only affects state that *must* be flushed



What's Needed from Privspec?

- Requirement that all on-chip state satisfies aISA
- Mechanisms for resetting temporally-partitioned state:
 - L1-I, L1-D cache: Sect 1.8 looks good
 - Need similar for TLB, branch predictor, prefetchers, ...
 - should not require any write-back?
 - ok to pump into single abstraction
- Latency bound requirement?
- Support for efficiently co-scheduling harts?



THANK YOU

Gernot Heiser | gernot@unsw.edu.au | @GernotHeiser

<https://trustworthy.systems>

