Time Protection:

Principled Prevention of Timing Channels

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

- ENTROPY’19, Stockholm, 16 June 2019

https://trustworthy.systems

Threats

=

Speculation

An “unknown unknown” until recently

+

A “known unknown” for decades

Microarchitectural Timing Channel

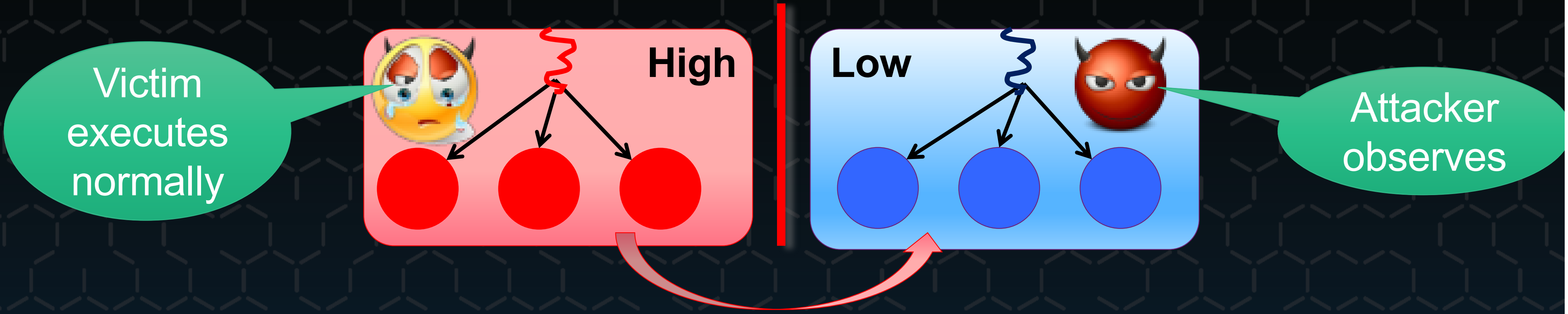
What Are Timing Channels?

Timing Channels

Information leakage through timing of events

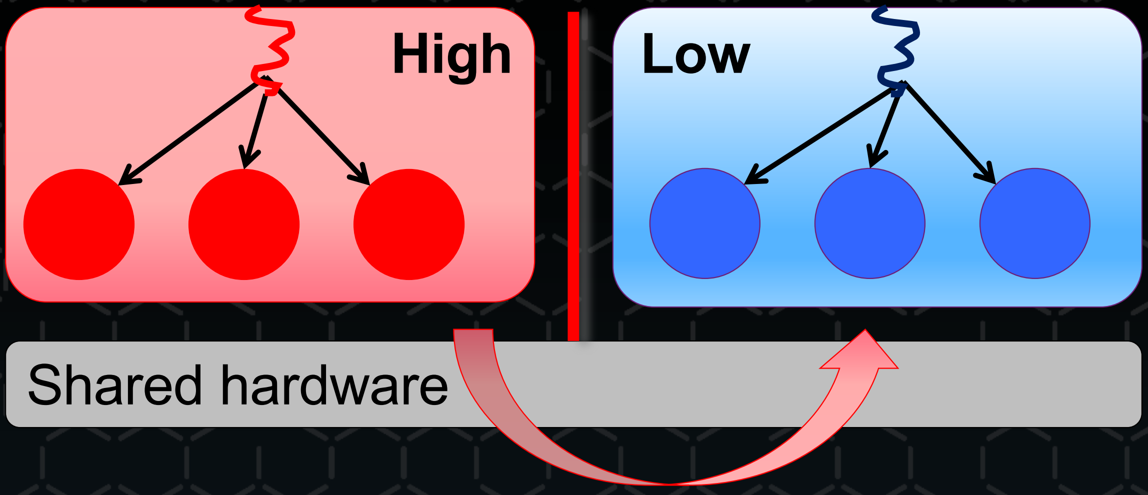
- Typically by observing response latencies or own execution speed

Covert channel: Information flow that bypasses the security policy



Side channel: Covert channel exploitable without insider help

Cause: Competition for Shared HW Resources

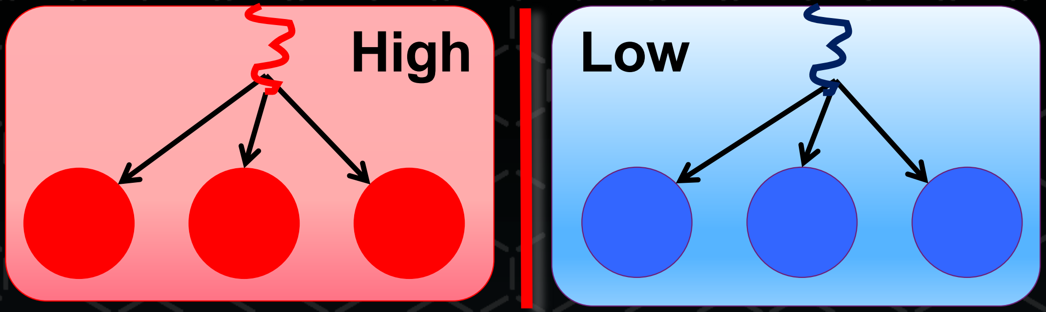


Affect execution speed

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

Preventing Timing Channels

Confidentiality Needs Time Protection

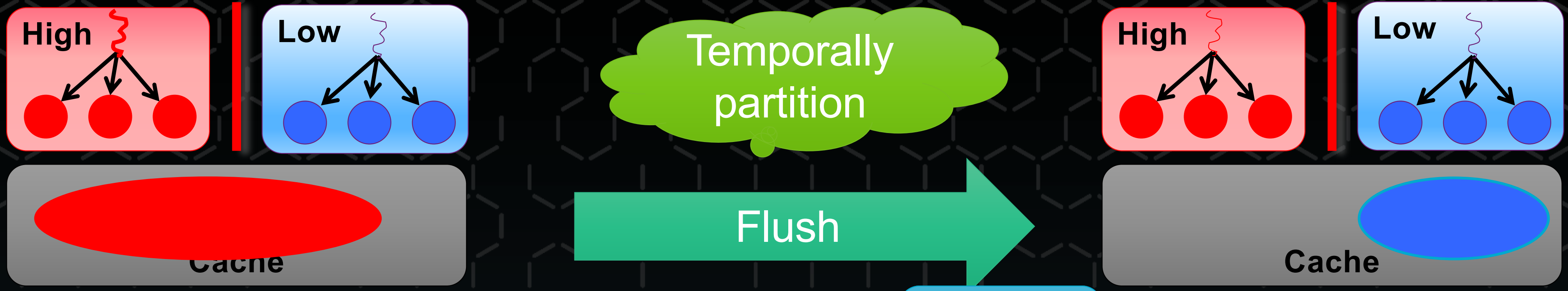


Traditionally OSes enforce security by *memory protection*, i.e. enforcing spatial isolation

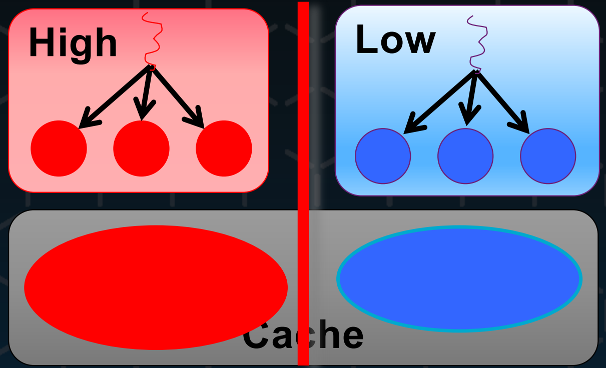
Time protection: A collection of *OS mechanisms* which collectively *prevent interference* between security domains that make execution speed in one domain dependent on the activities of another.

[Ge et al. EuroSys'19]

Time Protection: Partition Hardware



Spatially partition



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

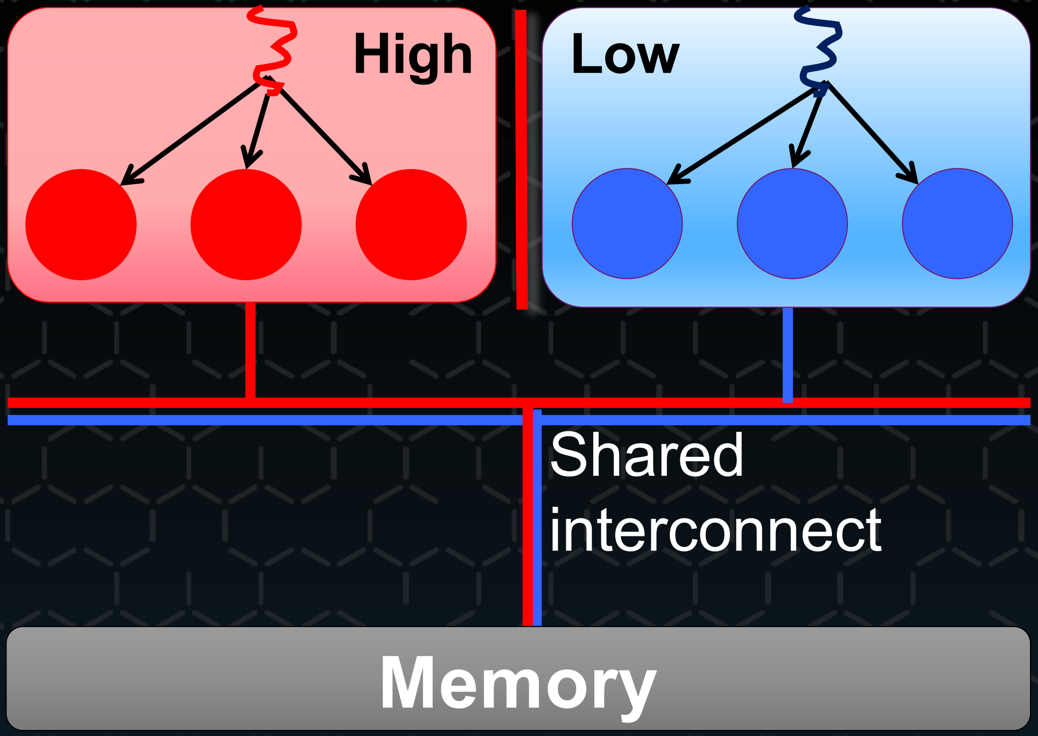
Timing channels can be closed *iff* the OS can

- (spatially) partition or
- reset

all shared hardware

On-core state

Sharing 1: Stateless Interconnect

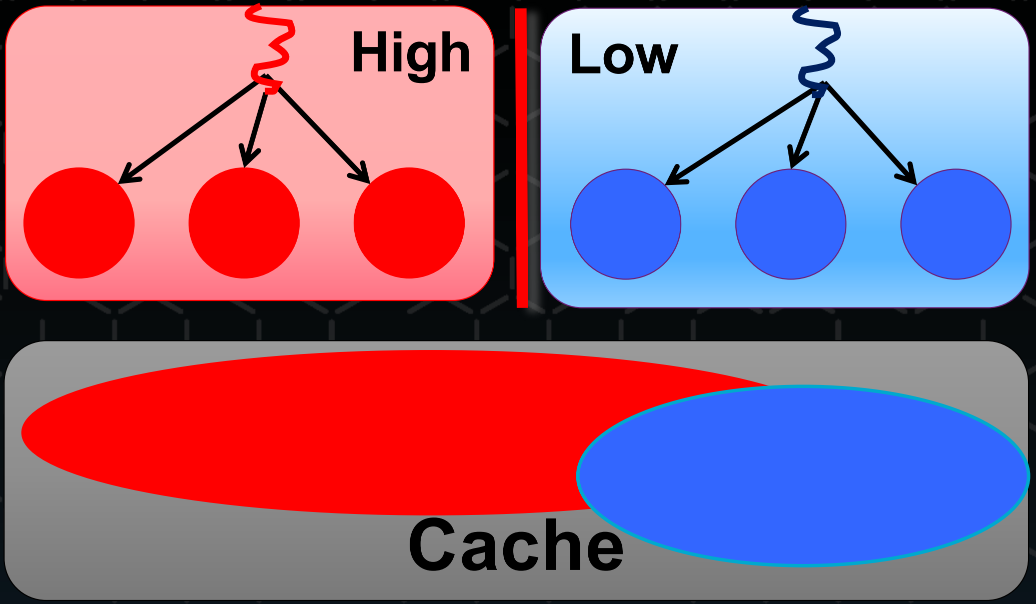


H/W is *bandwidth-limited*

- 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!

Sharing 2: Stateful Hardware



HW is *capacity-limited*

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

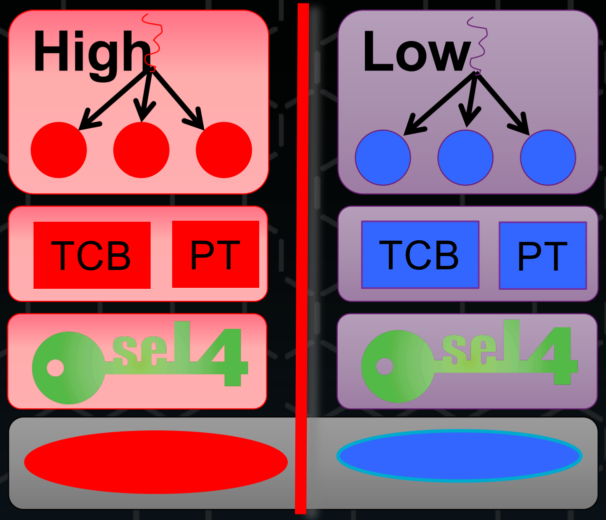
Solvable problem – focus of this work

Any state-holding microarchitectural feature:

- cache, branch predictor, pre-fetcher state machine

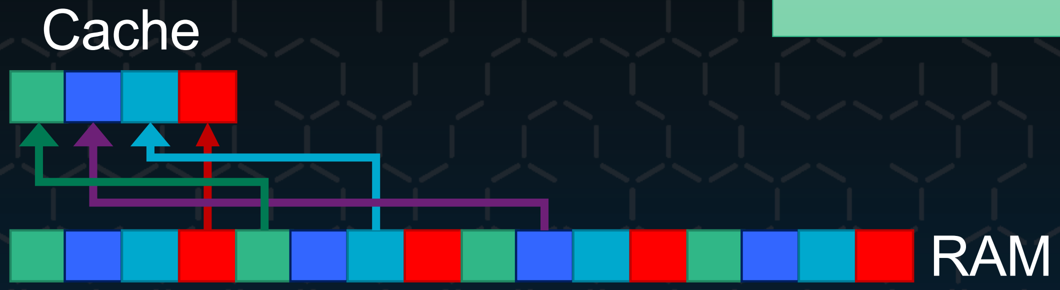
Implementing Time Protection on Stateful Hardware

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

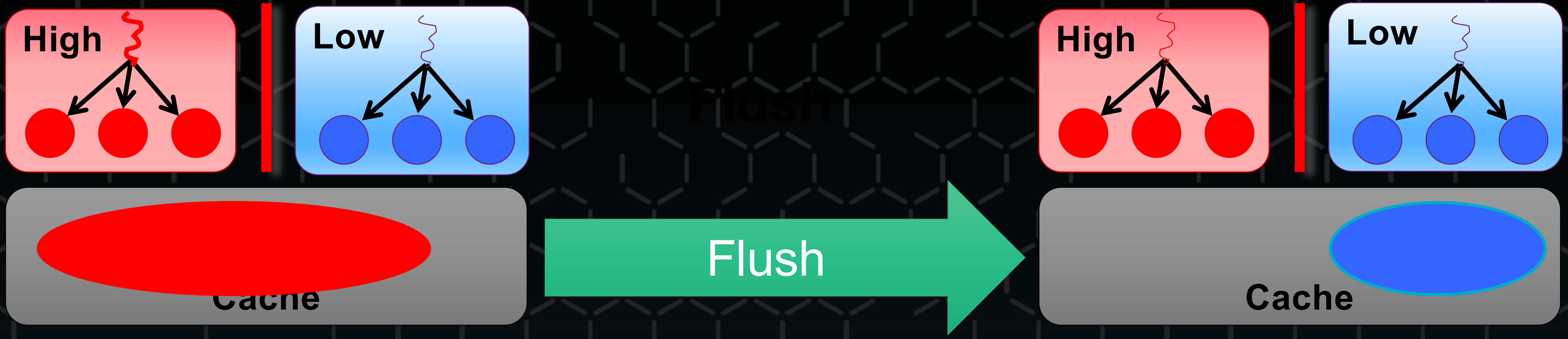
Latency depends on prior execution!

Time padding to Remove dependency

Ensure deterministic execution

Reality Check: Flushing On-Core State

Evaluating Intra-Core Channels

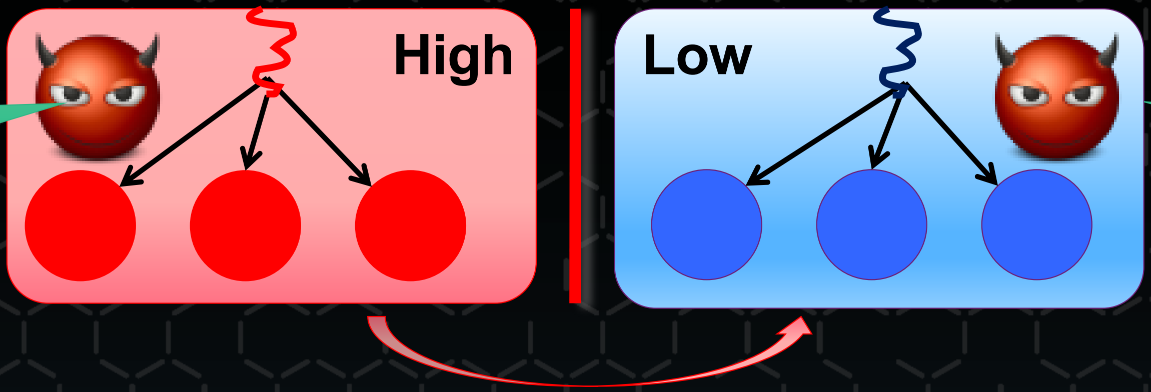


Mitigation on Intel and Arm processors:

- 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

Trojan encodes



Spy observes

1. Fill cache with own data

2. Touch *n* cache lines

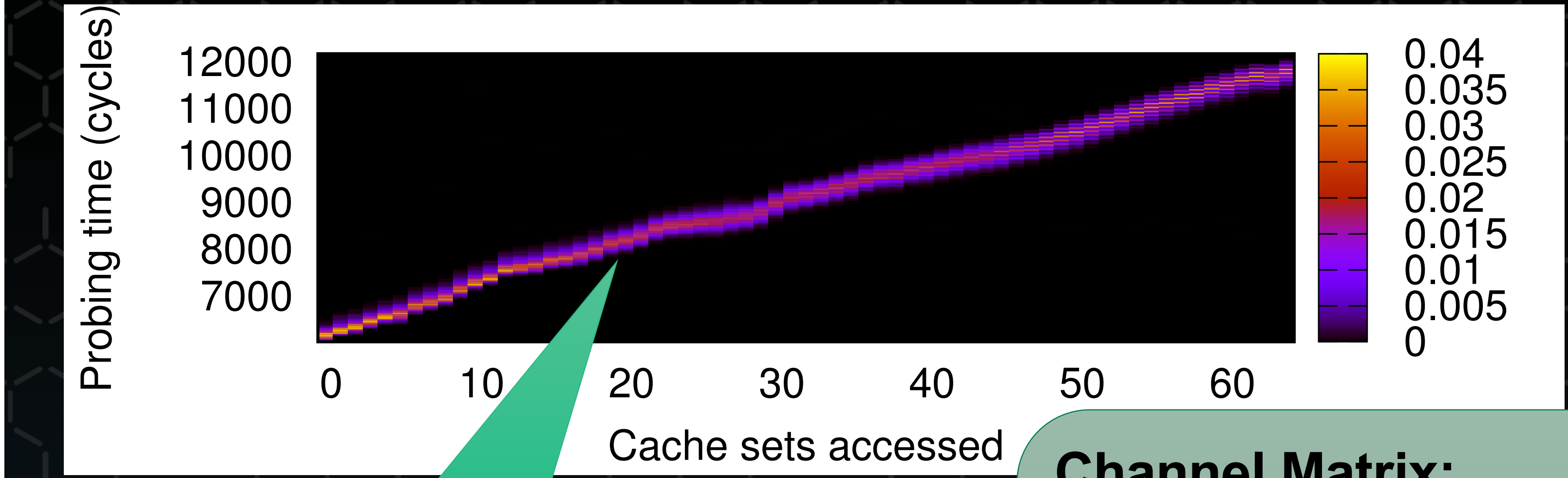
Input Signal

3. Traverse cache,

   measure execution time

Output Signal

Methodology: Channel Matrix



Raw I-cache channel
Intel Sandy Bridge

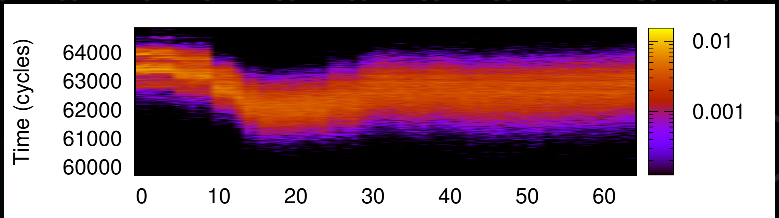
Horizontal variation indicates channel

Channel Matrix:

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

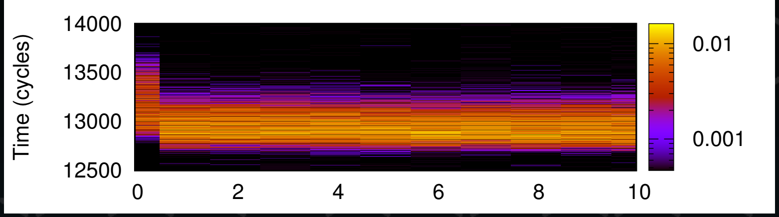
I-Cache Channel With Full State Flush

CHANNEL!



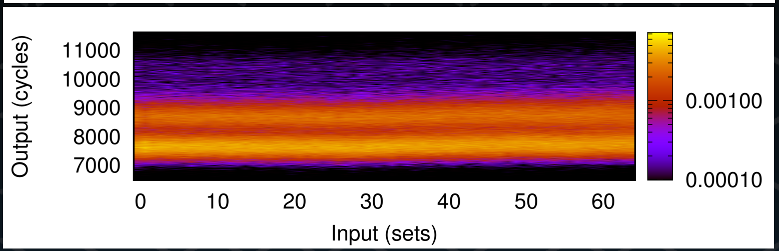
Intel Sandy Bridge

CHANNEL!



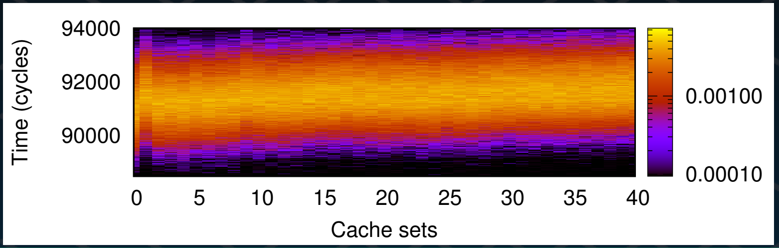
Intel Haswell

No evidence of channel



Intel Skylake

SMALL CHANNEL!

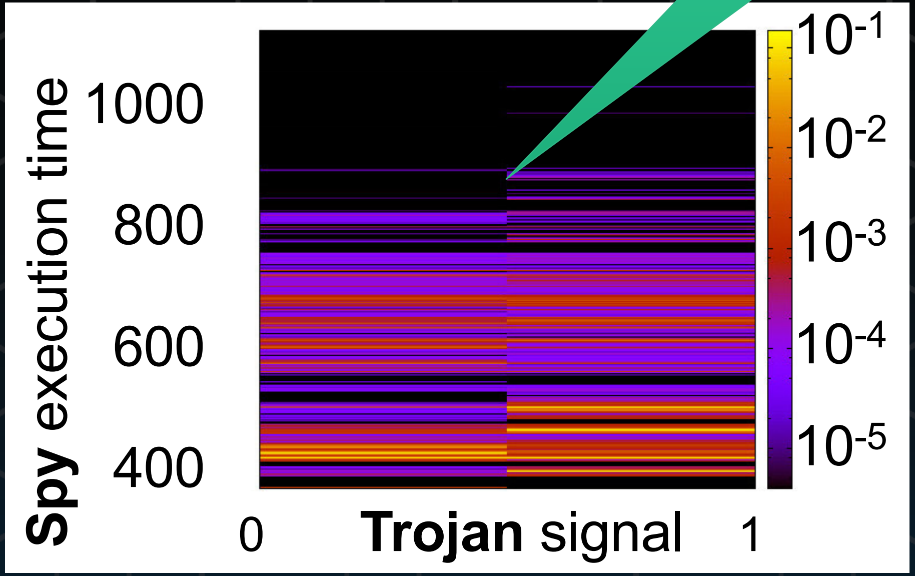


HiSilicon A53

HiSilicon A53 Branch History Buffer

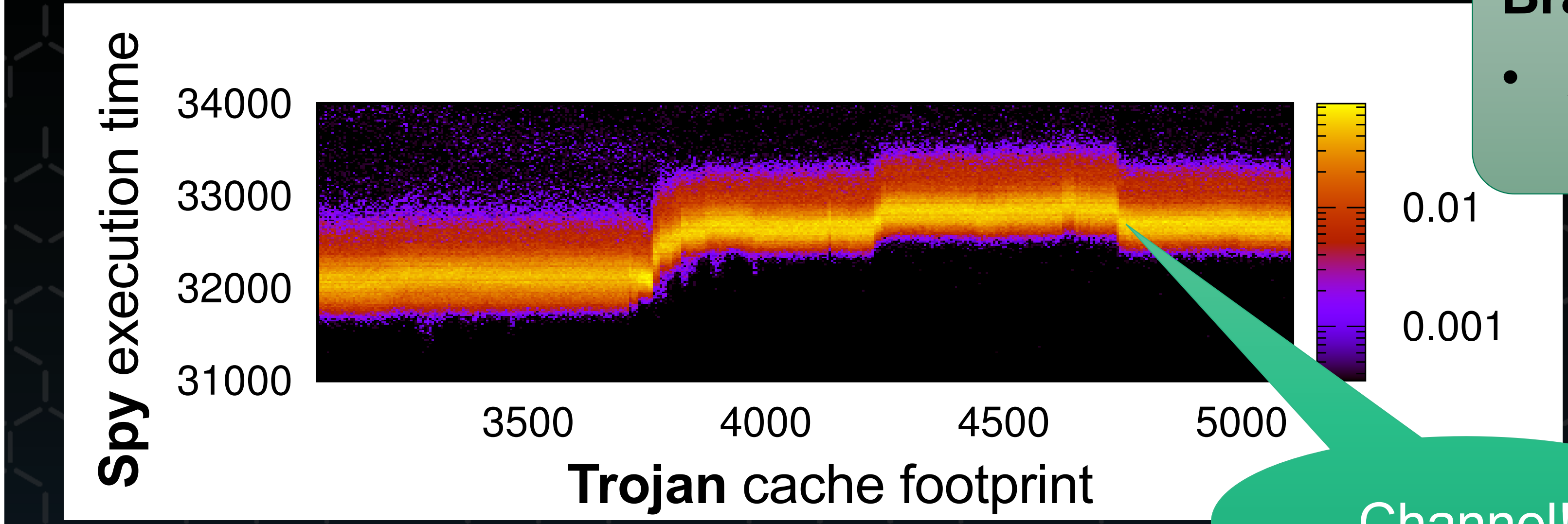
Branch history buffer (BHB)

- One-bit channel
- All reset operations applied



Channel!

Intel Haswell Branch Target Buffer



Channel!

Branch target buffer

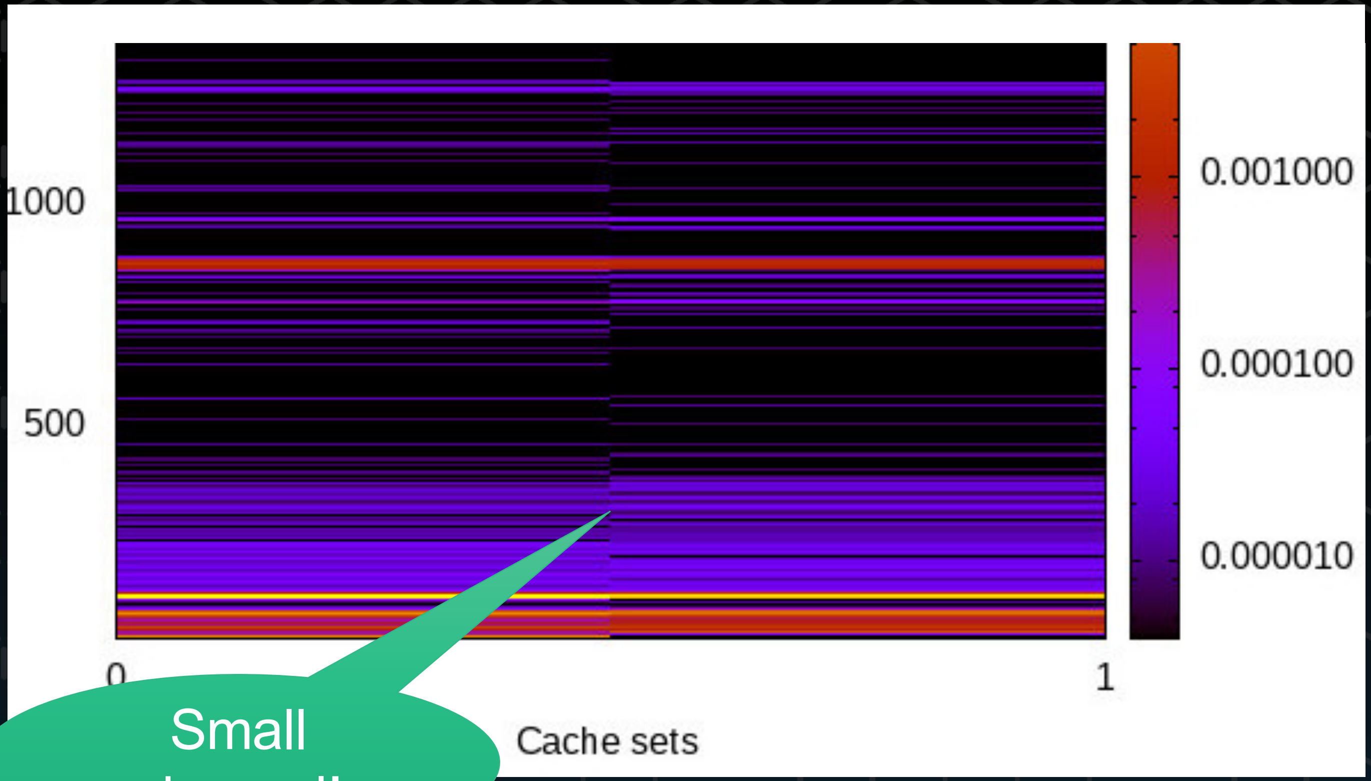
- All reset operations applied

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

Intel Spectre Defences

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

Intel Skylake
Branch history buffer



Small channel!

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

Requirements on Hardware

New HW/SW Contract: aISA

Augmented ISA supporting time protection

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

THANK YOU

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

https://trustworthy.systems