Time Protection

Principled Prevention of Timing Channels

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

- IFIP WG 10.4, Reggio di Calabria, 2020-01-31

https://trustworthy.systems

Threats

=

Speculation

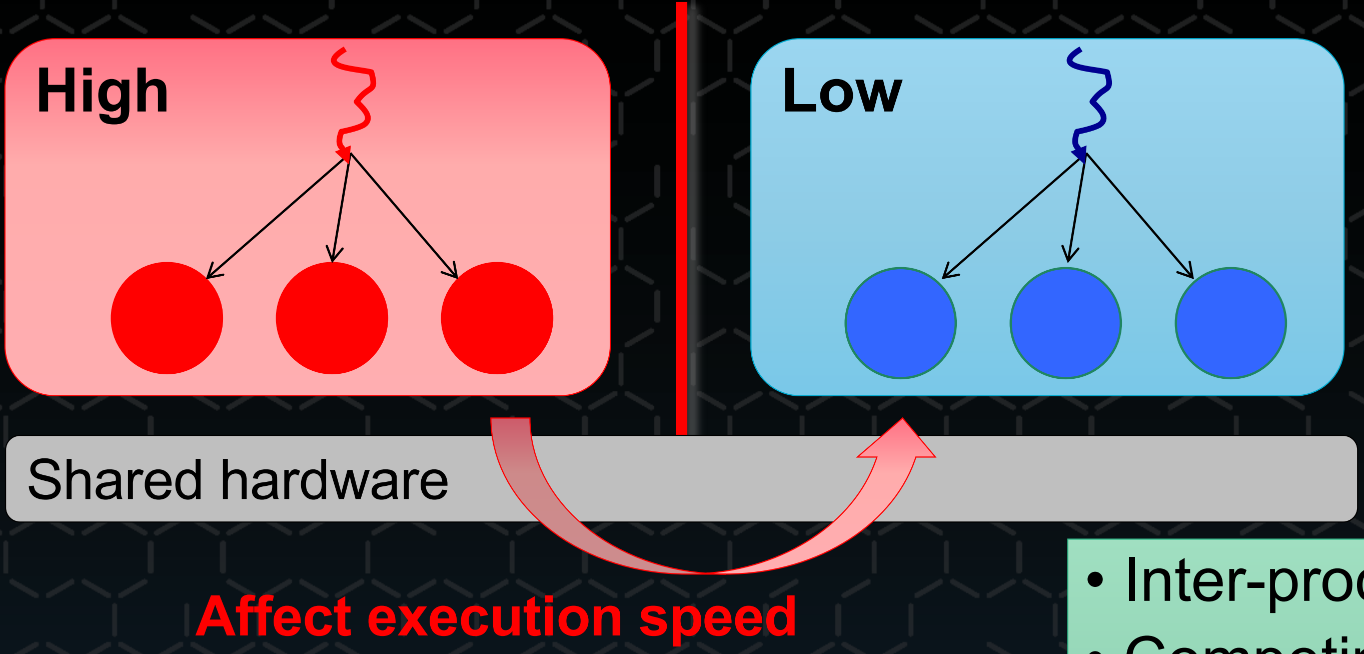
An “unknown unknown” until recently

+

A “known unknown” for decades

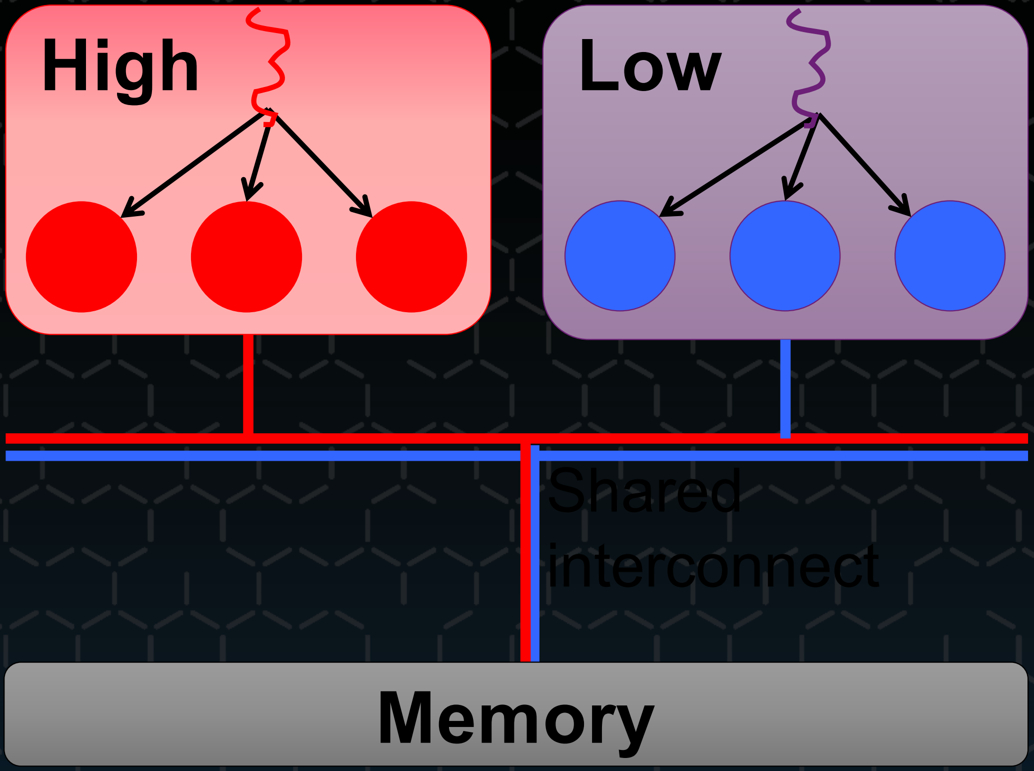
Microarchitectural Timing Channel

Cause: Competition for HW Resources



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

Sharing 1: Stateless Interconnect

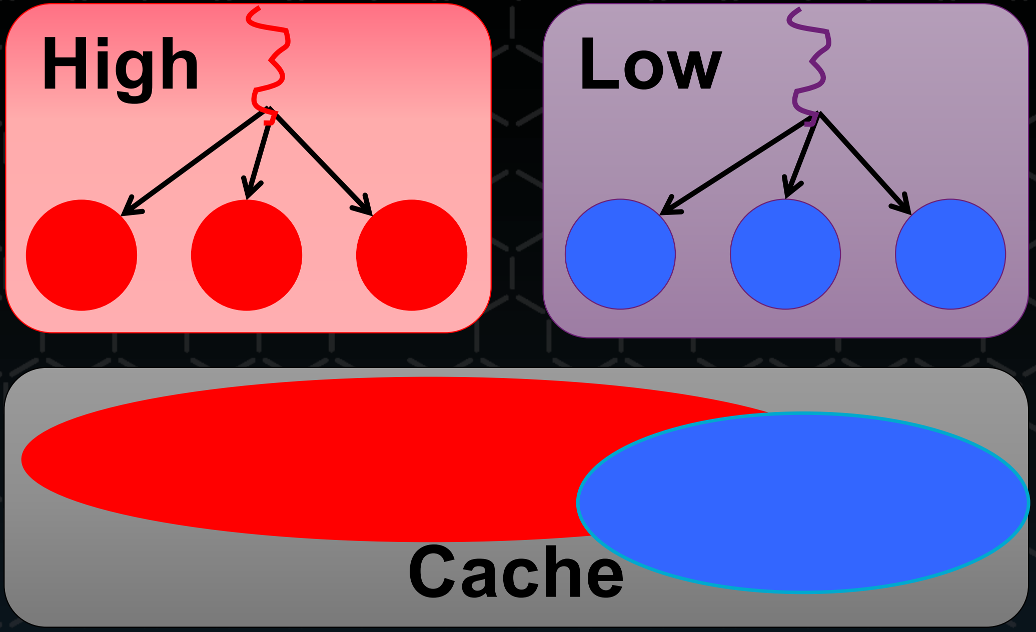


H/W is *bandwidth-limited*

- 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: Stateful Hardware



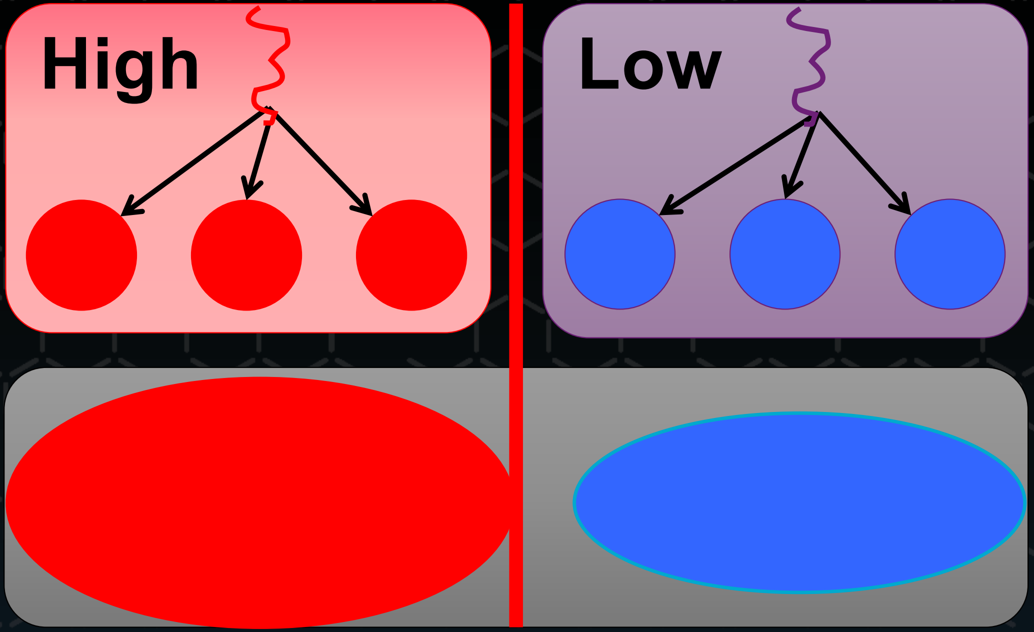
HW is *capacity-limited*

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

Any state-holding microarchitectural feature:

- cache, branch predictor, pre-fetcher state machine

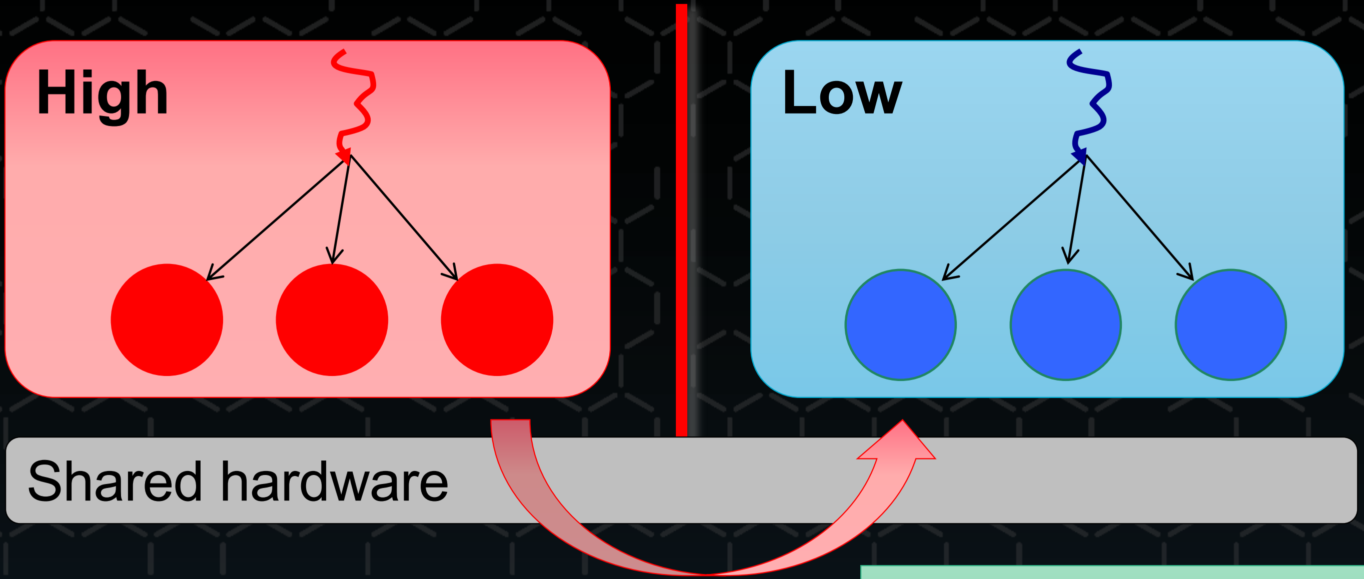
Systematic Defence: Time Protection



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

[Ge et al. EuroSys'19]

Time Protection: Prevent Interference

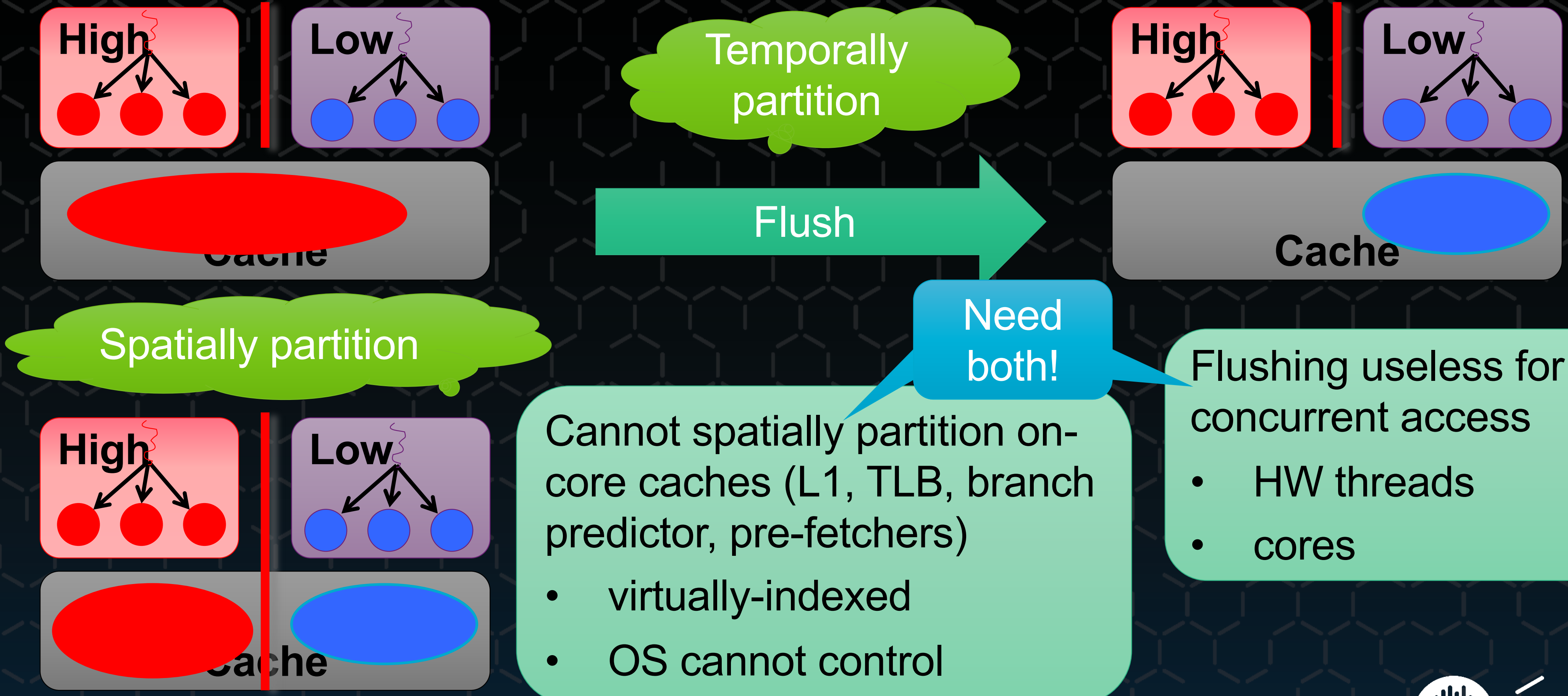


Affect execution speed

Interference results from sharing
⇒ Partition hardware:

- spatially
- temporally (time shared)

Time Protection: Partition Hardware



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

What is seL4?

seL4: Security, Safety, Performance

The world's **first** operating-system kernel with **provable** security enforcement

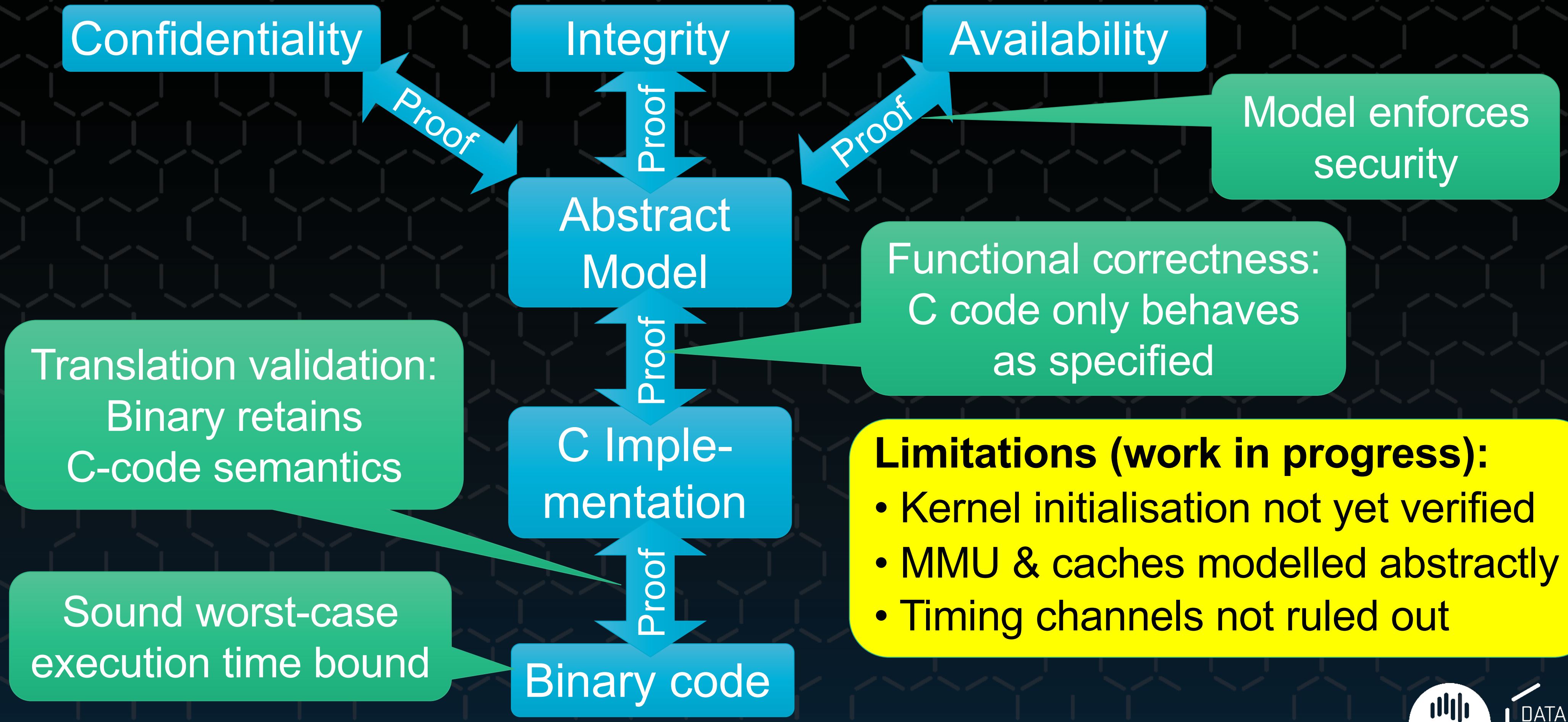
World's most advanced mixed-criticality OS

Open Source

The world's **only** protected-mode OS with complete, sound timeliness analysis

The world's **fastest** microkernel, designed for **real-world** use

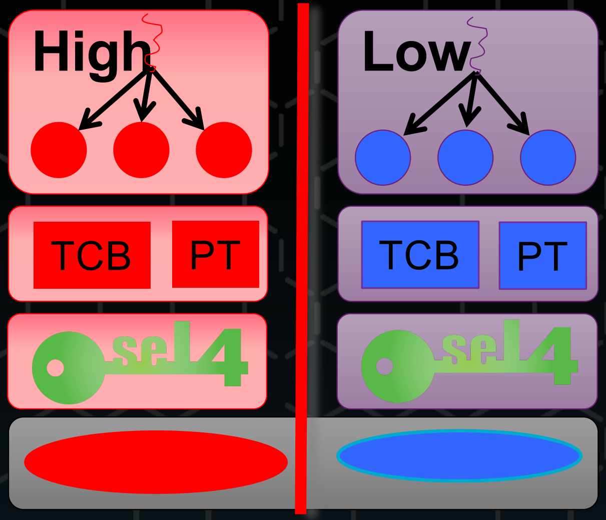
World’s Most Secure OS

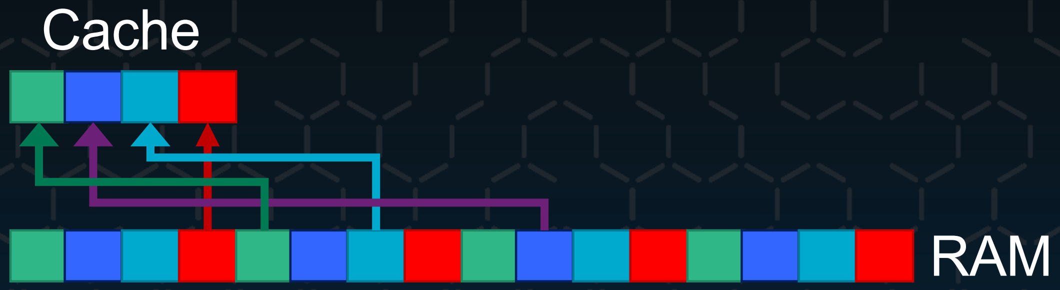

Confidentiality
Integrity
Availability
Proof
Proof
Proof
Model enforces security
Abstract Model
Proof
Functional correctness: C code only behaves as specified
Translation validation: Binary retains C-code semantics
C Imple-mentation
Proof
Sound worst-case execution time bound
Binary code

Limitations (work in progress):
- Kernel initialisation not yet verified
- MMU & caches modelled abstractly
- Timing channels not ruled out

Implementing Time Protection

Spatially Partition: 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]

Kernel remains policy-free, partitioning done at user level

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!

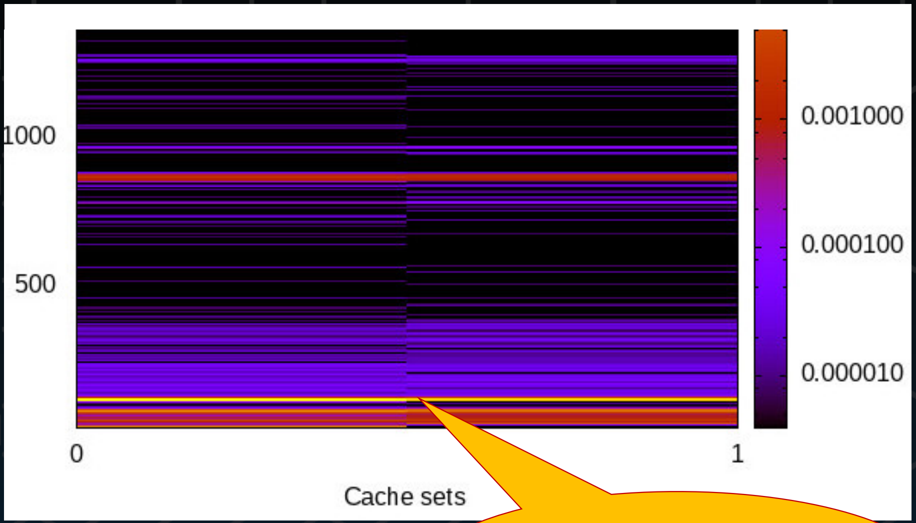
Time padding to Remove dependency

Ensure deterministic execution

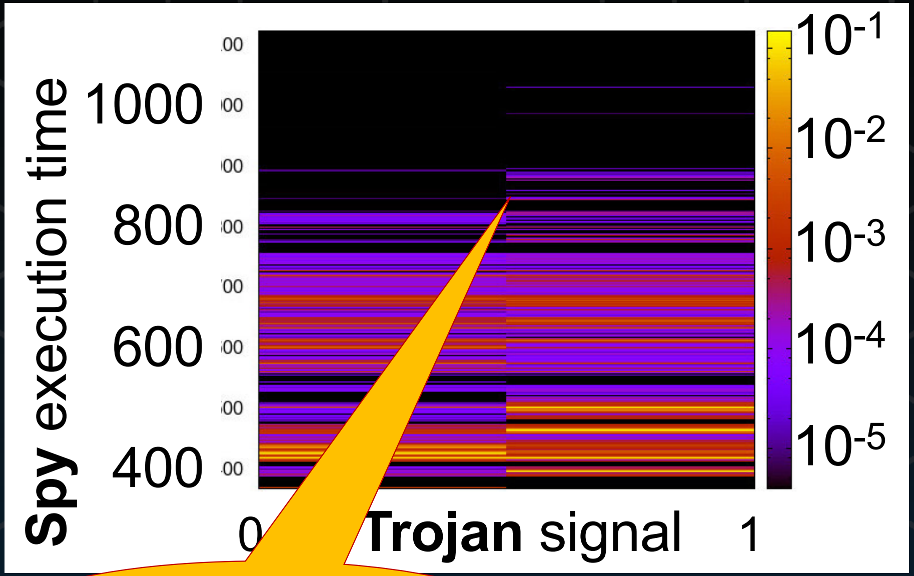
Challenge: Broken Hardware

- Systematic study of COTS hardware (Intel and Arm) [Ge et al, APSys'18]:
  - contemporary processors hold state that cannot be reset

Intel branch history buffer



HiSilicon A53 branch history buffer



Way Out: New HW-SW Contract!

ISA is purely functional contract, abstracts too much away

New contract (augmented ISA):

All shared HW resources must be spatially or temporally partitionable by OS

[Ge et al, APSys'18]

RISC-V to the rescue:
Strong commitment to making it happen!