Fast CHERI-RISC-V Compartmentalisation

Franz A. Fuchs, Jonathan Woodruff, Peter Rugg, Alexandre Joannou, and Simon W. Moore
Department of Computer Science and Technology, University of Cambridge
franz.fuchs@cl.cam.ac.uk
Project URL: cheri-cpu.org

UNIVERSITY OF CAMBRIDGE
Computer Science & Technology

Background

Transient-Execution Attacks

Speculative execution is used to leak secrets to an unprivileged attacker. Attacks rely on two microarchitectural mechanisms: First, **speculative execution** accesses a secret and encodes it in the microarchitecture. Second, a **microarchitectural side channel** is used to decode the secret.

	CHERI-RISC-V
Spectre-PHT	Safe*
Spectre-BTB	Vulnerable
Spectre-RSB	Vulnerable
Spectre-STL	Vulnerable
Meltdown-US-CHERI	Safe
Meltdown-GP-CHERI	Safe

Table 1. Results obtained on CHERI-Toooba; *when used bypass bounds check, and when running in pure-capability mode.

Conventional Compartmentalisation

Compartmentalisation enables privilege decomposition. A traditional compartmentalisation strategy is to separate one process into multiple smaller processes, which are then referred to as *compartments*. Compartmentalisation decreases the attack surface because malicious code cannot escape its compartment.

Compartmentalisation can be used as a mitigation mechanism against transient-execution attacks. By isolating speculation state within compartments, attackers are not able to conduct cross-compartment attacks.

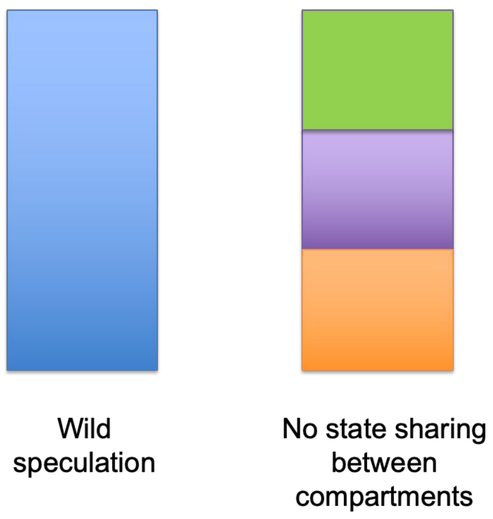

Figure 1. Split one process into multiple compartments. Speculation across compartments is not allowed and thus mitigates cross-compartment transient-execution attacks.

CHERI Capabilities

CHERI adds fine-grained memory capabilities that allow for fine-grained compartmentalisation. Capabilities not only describe a region of memory, but also authorise access to it. A core feature of CHERI are sealed capabilities. Sealed capabilities are immutable and non-dereferenceable.

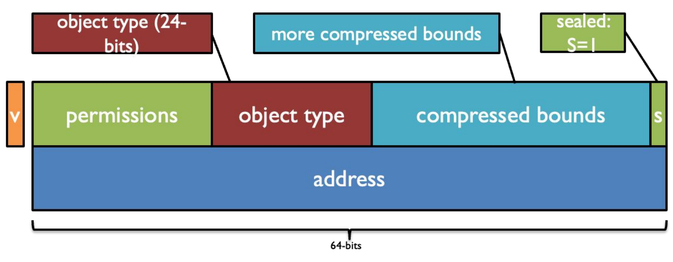

Figure 2. A 129-bit CHERI capability enabling fine-grained memory protection.

What does the roadmap for fast and secure CHERI compartments look like?

Implementing secure compartments on CHERI is a complex process and requires input from multiple parties. We have laid out the main challenges on the road towards secure compartmentalisation on CHERI-RISC-V:

- Defining fast and secure compartmentalisation models with CHERI
- Identifying microarchitectural state and constraining sharing it between compartments
- Defining and evaluating domain-crossing mechanisms

CHERI Compartments

In CHERI, compartments can be built with CHERI capabilities. CHERI compartments enable both **fast** and **fine-grained** compartmentalisation.

In CHERI, compartments can be as small as the programmer decides to narrow down the capability. Thus, CHERI enables different compartmentalisation models ranging down to compartment sizes of a single function. Two or more compartments can be put into the same address space because a mutually exclusive set of capabilities will prevent any access into a compartment without explicitly having been granted with this privilege.

Process-based compartmentalisation using the MMU is heavily applied in today's computing world. Often, no compartmentalisation is used when process-based compartmentalisation is getting too performance costly. CHERI can be seen as an **accelerator for compartmentalisation**.

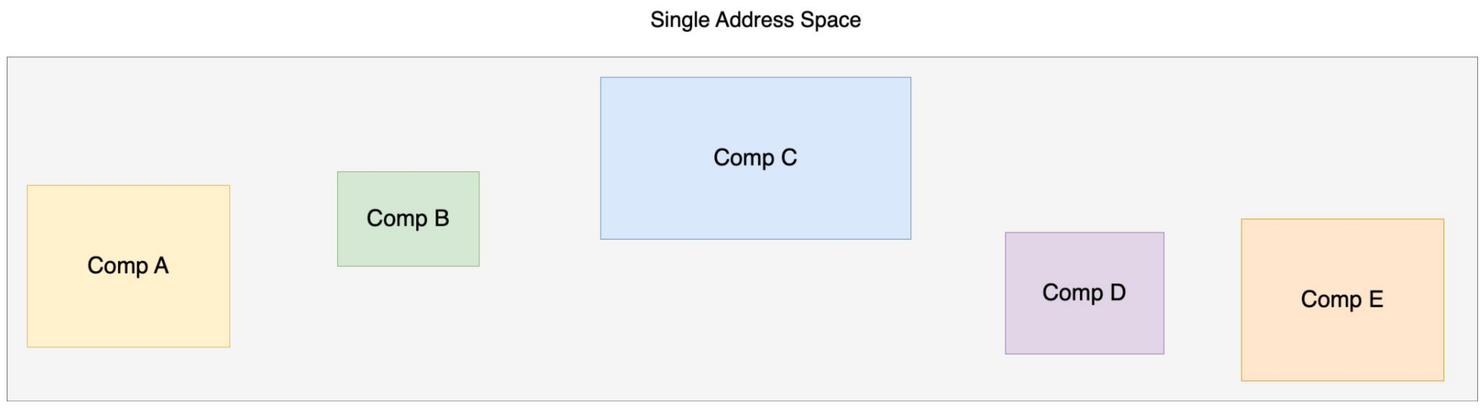

Figure 3. The compartments are separated from each other via sets of mutually exclusive capabilities. Thus, all compartments can live in the same address space rendering costly domain-crossing mechanisms, e.g., kernel calls, unnecessary.

Identifying Microarchitectural State

Microarchitectures have become increasingly complex in the past decades to fuel the ever increasing need for single-core performance. Therefore, microarchitectures employ a great amount of state. In order to secure compartments, we need to identify state all over the entire microarchitecture.

Securing microarchitectural state likely comes at a performance cost. Therefore, we envision the need for software to decide whether it wants to share state with other compartments. A natural way could be **compartment IDs** (CIDs) that can be used by microarchitectures to separate state between compartments

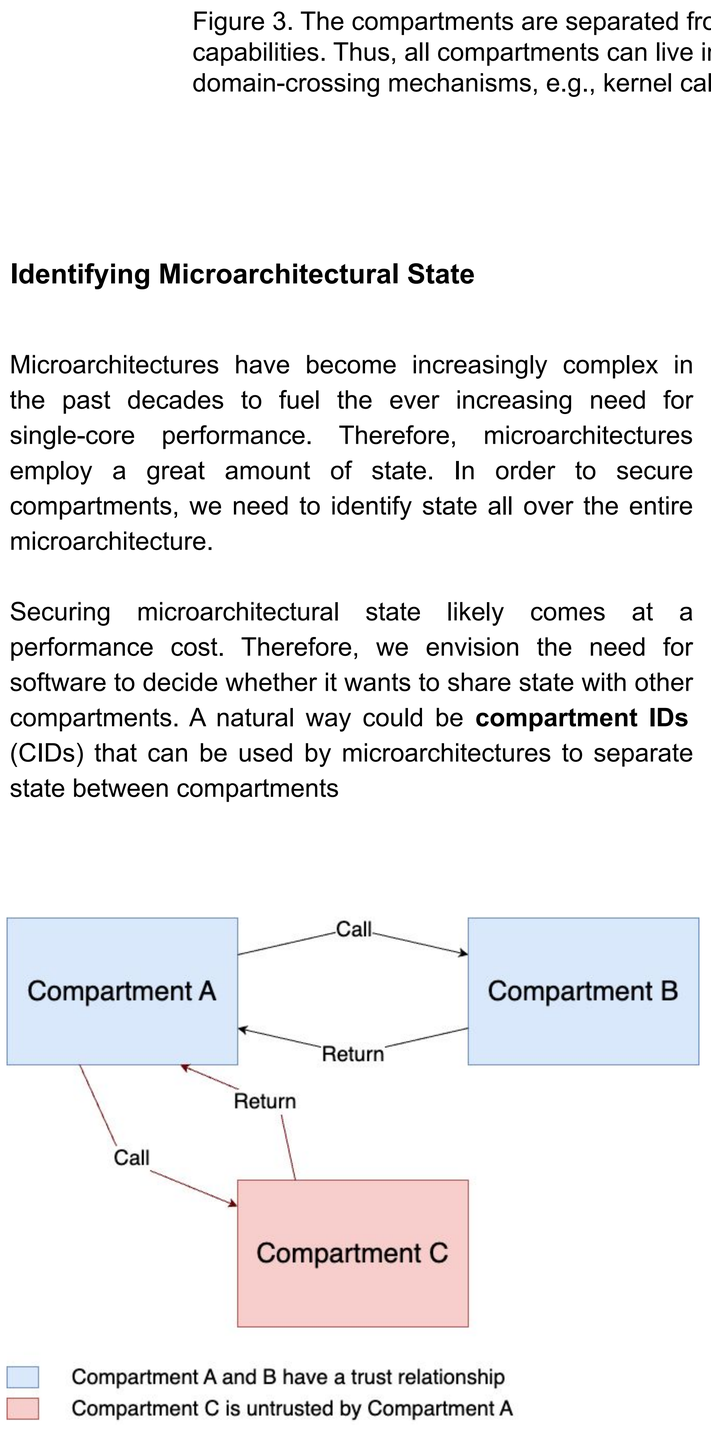

Figure 4. Compartment A and B can express their trust relationship through their respective CIDs.

Domain-Crossing Mechanisms

Transitioning from one compartment to another needs to be both **fast** and **secure**. More fine-grained compartments will lead to an increased number of domain transitions.

We propose to extend the current RISC-V CHERI draft with a minimal set of architectural compartmentalisation primitives:

- Immutable entry points: Trusted code, e.g., a trampoline, can restrict at which point it is entered. In CHERI, these are called **sentry** (sealed entry) capabilities.
- Reliable source of trusted data capability: Needed to save information that cannot leak to any other compartments. We propose a **thread ID capability** (xTIDC) per privilege mode.
- Nestability: Multiple compartmentalisation models need to be nestable in order to guarantee protection at different levels. We are currently evaluating to build **capability abstractions** over xTIDC.
- ...

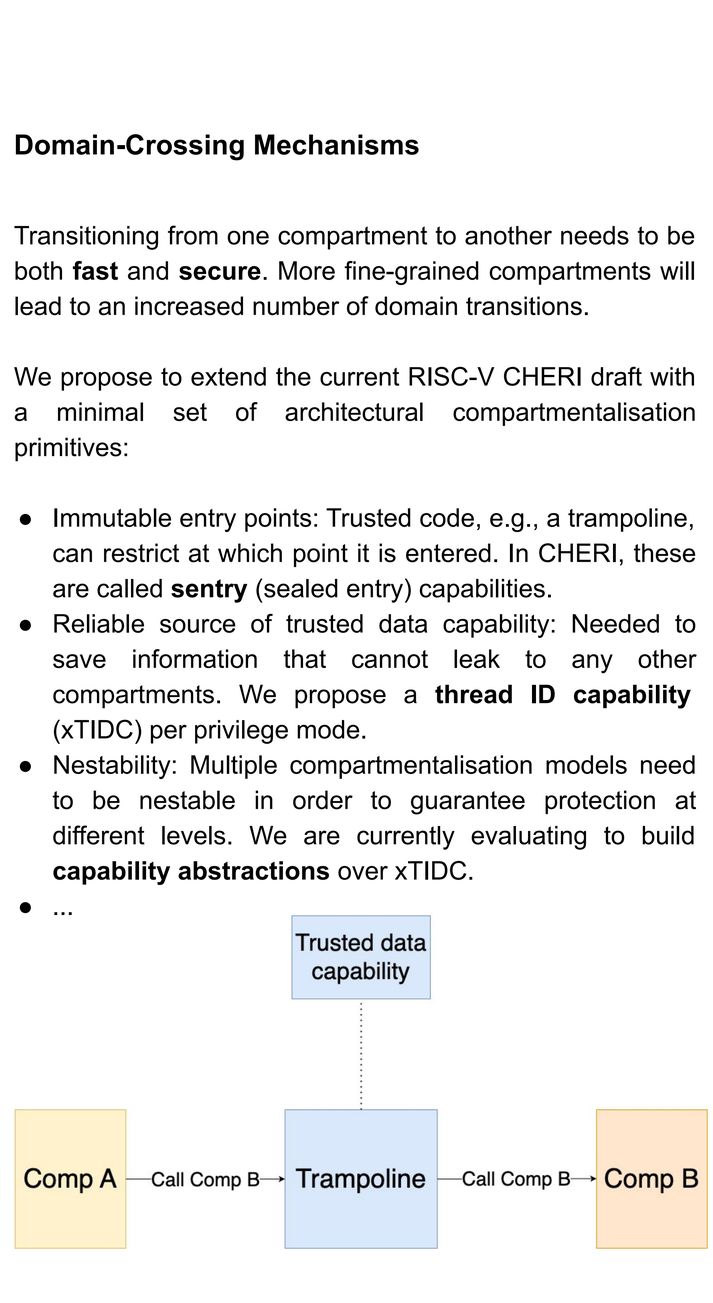

Figure 5. The trampoline needs access to a reliable data capability to separate Comp A and Comp B.

In collaboration with SRI International

This work was supported by the Engineering and Physical Sciences Research Council EP/S030867/1.

CHERI