Fast and secure compartmentalisation on CHERI

Franz A. Fuchs, Jonathan Woodruff, Peter Rugg, Robert N. M. Watson, 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.

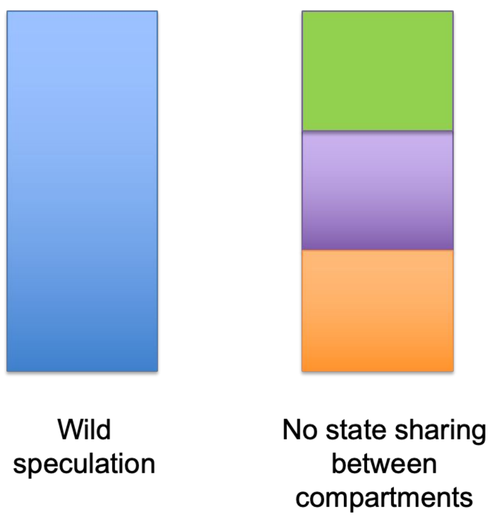

Figure 1. Split one process into multiple compartments. Speculation across compartments is not allowed and thus mitigates cross-compartment transient-execution attacks.

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.

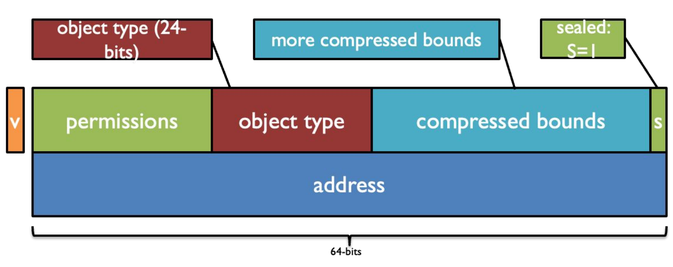

Figure 2. A 129-bit CHERI capability enabling fine-grained memory protection.

What does the roadmap for 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 transient-execution security via CHERI compartments:

- Filling the architectural void to specify speculative execution paths.
- Identifying microarchitectural state and constraining sharing it between compartments.
- Defining and evaluating domain-crossing mechanisms, and making transitions themselves secure.

Architectural Specification Vacuum

Architectures suffer from a specification vacuum with respect to behaviour in speculation. On the one hand, hardware designers can freely operate as long as their designs uphold the architectural abstraction, allowing aggressive speculation to be conducted under the hood. On the other hand, software mitigations are being developed around ad hoc mental models of speculative execution that may bear little relation to the range of speculative behaviours that have been implemented.

Therefore, we have the strong need for architectural guarantees. These guarantees provide strong and simple security primitives for software. This means that software can be verified to be secure against the ISA, including the new speculative isolation guarantees. On the hardware side, the guarantees of isolation of speculation state are mandated to be implemented, but hardware designers are free to operate within the constraints. These designs can then be tested against the ISA.

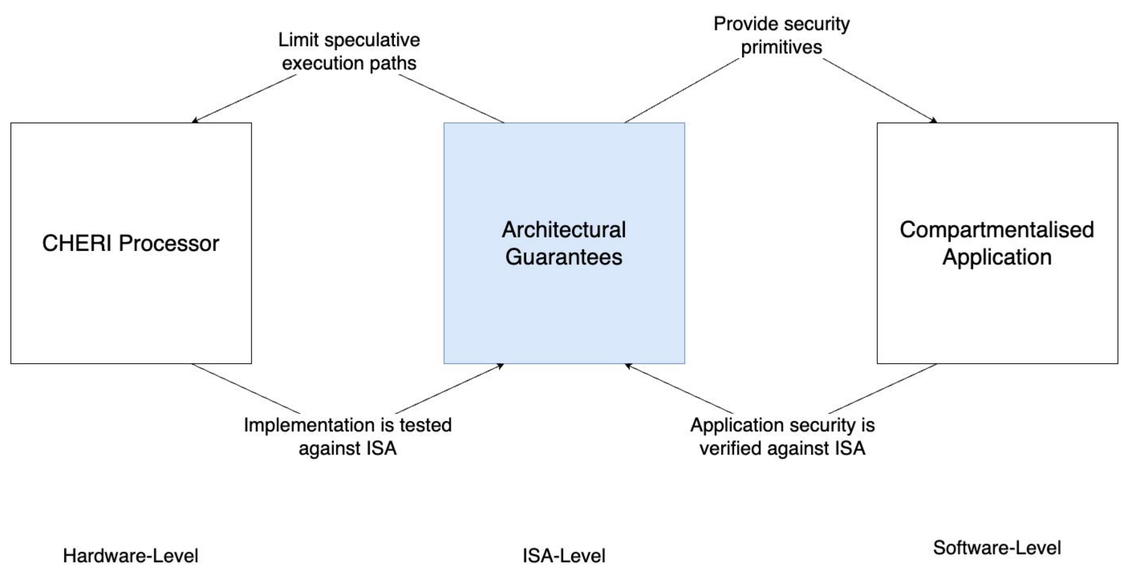

Figure 3. Architectural guarantees play an integral role for transient-execution attack mitigations for compartmentalised applications.

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.

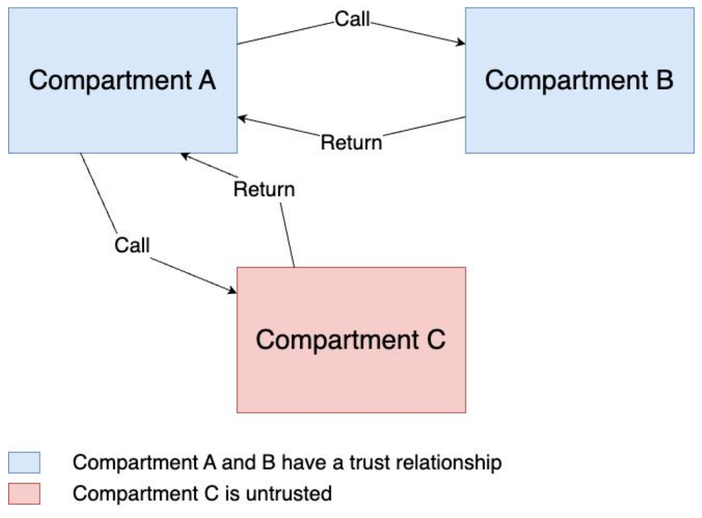

Figure 4. Compartment A and B can express their trust relationship through their respective CIDs.

Domain-Crossing Mechanisms

The CHERI v9 specification currently defines two mechanism for domain-crossing:

- Sealed entry capabilities.
- Use otypes to link a pair of capabilities.

We are evaluating the mechanisms above and propose to research additional mechanisms to suffice the requirements of software compartmentalisation models:

- Immutable entry points: Trusted code can only be called through.
- Reliable source of trusted data capability: Needed to save information that cannot leak to any other compartments.
- Nestability: Multiple compartmentalisation models need to be nestable in order to guarantee different levels of protection.
- ...

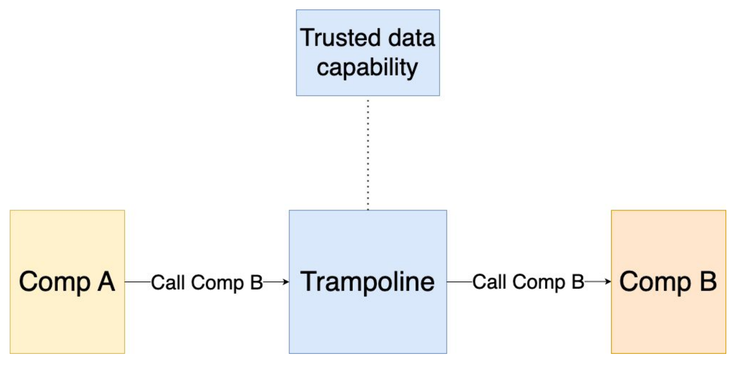

Figure 5. The trampoline needs access to a reliable data capability to separate Comp A and Comp B.

In collaboration with SRI International®

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

CHERI