CHERI Compartments

Toward Transient-Execution Attack Mitigations on CHERI Compartments

Franz A. Fuchs, Robert N. M. Watson, Simon W. Moore, Peter Sewell, Peter G. Neumann
Hesham Almatary, Jonathan Anderson, Alasdair Armstrong, Peter Blandford-Baker,
John Baldwin, Hadrien Barrel, Thomas Bauereiss, Ruslan Bukin, David Chisnall, Jessica Clarke, Nirav Dave, Brooks Davis,
Lawrence Esswood, Nathaniel W. Filardo, Dapeng Gao, Khilan Gudka, Brett Gutstein, Alexandre Joannou,
Mark Johnston, Robert Kovacsics, Ben Laurie, A. Theo Markettos, J. Edward Maste, Alfredo Mazzinghi,
Alan Mujumdar, Prashanth Mundkur, Steven J. Murdoch, Edward Napierala, George Neville-Neil, Robert Norton-Wright,
Philip Paeps, Lucian Paul-Trifu, Allison Randal, Ivan Ribeiro, Alex Richardson, Michael Roe, Colin Rothwell, Peter Rugg,
Hassen Saidi, Peter Sewell, Thomas Sewell, Stacey Son, Domagoj Stolfa, Andrew Turner, Munraj Vadera,
Konrad Witaszczyk, Jonathan Woodruff, Hongyan Xia, and Bjoern A. Zeeb

University of Cambridge and SRI International

CHERITech

Glasgow, 31 March 2023

Approved for public release; distribution is unlimited. This research is sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL), under contract FA8750-10-C-0237. The views, opinions, and/or findings contained in this article/presentation are those of the author(s)/presenter(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

Approved for public release; distribution is unlimited.

This work was supported by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL), under contract FA8750-10-C-0237 ("CTSRD"), with additional support from FA8750-11-C-0249 ("MRC2"), HR0011-18-C-0016 ("ECATS"), and FA8650-18-C-7809 ("CIFV") as part of the DARPA CRASH, MRC, and SSITH research programs. The views, opinions, and/or findings contained in this report are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

This work was supported in part by the Innovate UK project Digital Security by Design (DSbD) Technology Platform Prototype, 105694.

We also acknowledge the EPSRC REMS Programme Grant (EP/K008528/1), the ERC ELVER Advanced Grant (789108), the Isaac Newton Trust, the UK Higher Education Innovation Fund (HEIF), Thales E-Security, Microsoft Research Cambridge, Arm Limited, Google, Google DeepMind, HP Enterprise, and the Gates Cambridge Trust.

Transient-Execution Attacks

Transient-execution attacks combine:

- Directed speculative execution
- Side-channels, e.g., cache timing

Spectre v1 is most infamous example:

```
if (idx0 < size){
        int idx1 = array0[idx0];
        int idx2 = array1[idx1];
        …
}
```

Cache lines for `array1`

Transient-Execution Attacks

Transient-execution attacks have become numerous:

- Spectre-like: Following control-flow or data-flow misprediction
- Meltdown-like: Following a faulting instruction
- MDS (Microarchitectural Data Sampling): Leaking in-flight data from buffers

Leakage sources and reasons:

- Branch direction prediction
- Indirect jump target prediction
- Return address prediction
- Memory disambiguation
- Speculative load forwarding
- Instruction scheduling
- Out-of-order execution
- Reading from store buffers
- …

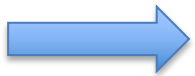
Transient-Execution Attacks on CHERI

Previous research shows that CHERI systems can be vulnerable to transient-execution attacks

- Traditional Spectre attacks mostly work
- CHERI can, but does not need to protect against transient-execution attacks

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

Results on obtained on CHERI-Toooba

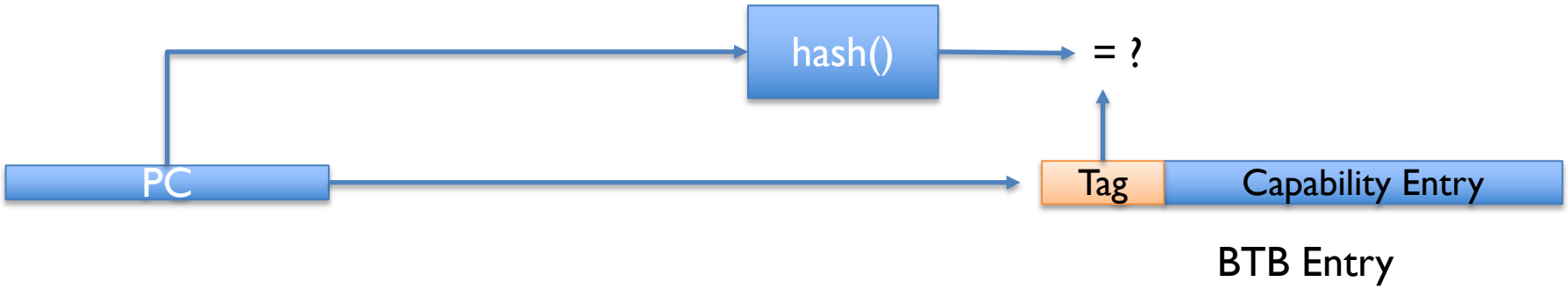


Mitigation of transient-execution attacks is caused by design properties of CHERI-Toooba rather than CHERI

Transient-Execution Attacks on CHERI

What are the reasons for the successful attacks on CHERI systems?

"Speculation with capabilities"



Malicious code retrieves a capability and all its permissions during speculative execution!

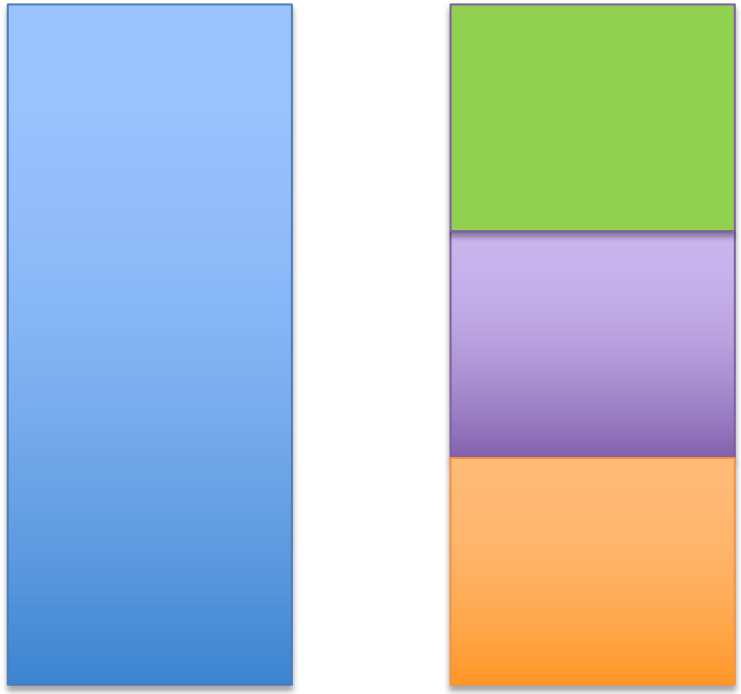
How to mitigate transient-execution attacks?

Compartmentalisation

Conventional compartmentalisation:

- Privilege decomposition
- Traditionally separate one process into multiple compartments with less privileges each
- Decrease the attack surface

→ Malicious code cannot escape their compartment



Big process

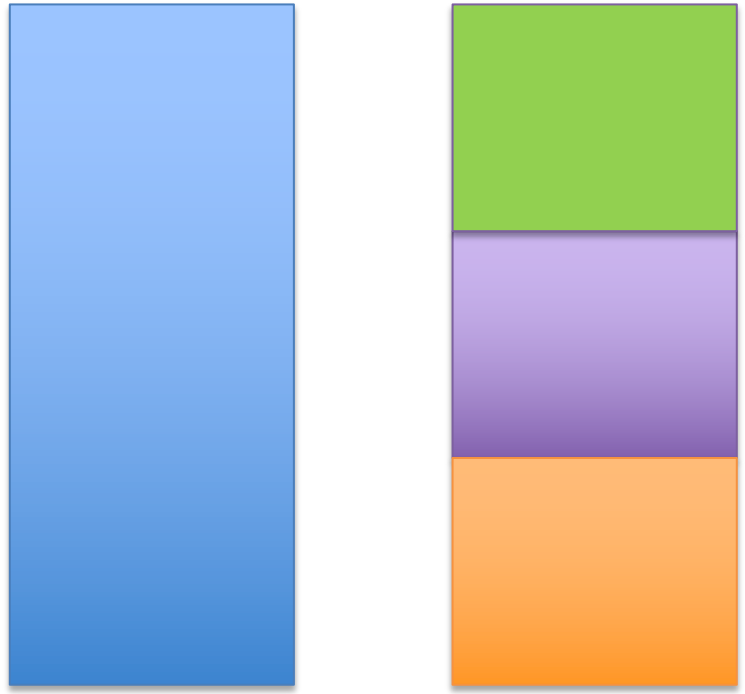
Multiple small compartments

Compartmentalisation against Transient Execution

Effective mitigation approach, but:

- Coarse-grained approach does not allow for fine-grained protection
- High performance cost when process switching

We need to do better!



Wild speculation

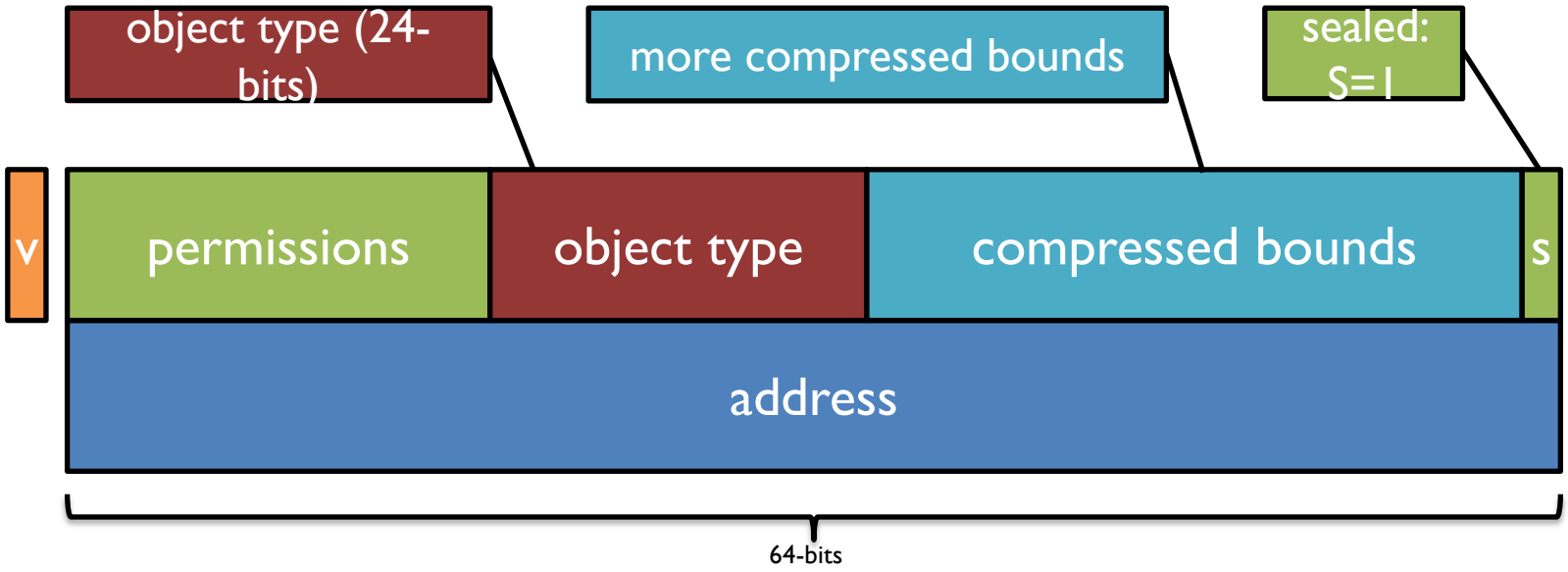
No state sharing between compartments

Compartmentalisation with CHERI

Advantages against conventional compartmentalisation:

- Fine-grained decomposition
- Built-in domain transitions, e.g., through capability sealing (non-dereferencable capabilities)

That sounds like the problem is already solved!



What are the main challenges ahead?

Compartmentalisation Challenges (1)

"The architectural specification vacuum"

Currently, the architecture does not constrain speculative execution

We need architectural guarantees about speculative execution for security

Guarantees needed for:

- Ability to test hardware for security properties
- Build secure software on top of architectural guarantees

Compartmentalisation Challenges (2)

"Understanding what we need to protect"

High-end microarchitectures have more and more state (Apple M1 has 16 billion transistors as an indicator for microarchitectural state growth)

Different forms of state:

- Long-term (i.e., BTB)
- Transient (i.e., scheduling in a pipeline)

Reason for complexity:

Need for ever increasing single-core performance!

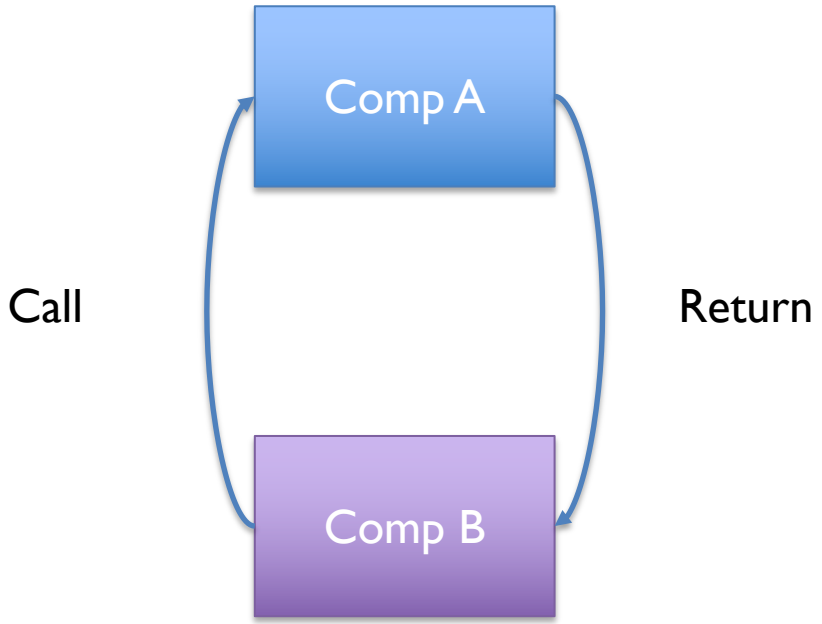
We need a full microarchitectural audit!

Compartmentalisation Challenges (3)

"How to implement a compartment?"

Hardware design challenges:

- Large increase in compartment switches due to fine-grained compartments
- Every compartment switch is more expensive than a function call
- State sharing can benefit performance
- Dedicated additional microarchitectural state for compartments needed



Calls and returns need to be inexpensive

Compartmentalisation Challenges (4)

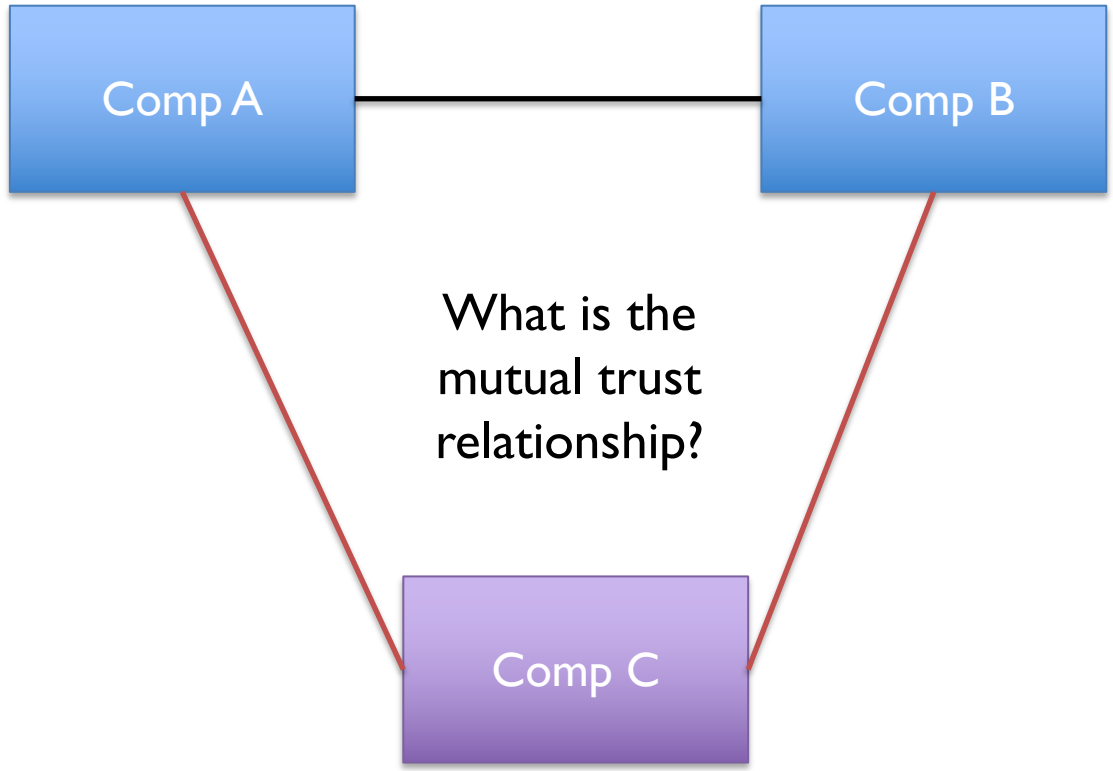
"How to implement security?"

Security challenges:

- Multiple different security models
- Hierarchy of compartments
- Trust relations between compartments

The two compartments have the same code

The two compartments have different code



Conclusions

- Compartmentalisation is needed to mitigate transient-execution attacks
- Industry solutions are performance expensive and coarse grained
- CHERI allows fine-grained compartmentalisation
- Research in progress:
  - Architectural specification necessary
  - Identifying relevant microarchitectural state
  - Implementing microarchitectural solutions