Safe Speculation for CHERI

Franz A. Fuchs, Jonathan Woodruff, Peter Rugg, Alexandre Joannou, and Simon W. Moore

Department of Computer Science and Technology, University of Cambridge

Abstract

Transient-execution attacks continue to plague the computer hardware industry. Recent attacks show that they can leak sensitive information on many processors of Apple's M chip series. These attacks cannot only target conventional systems, but also secure architectures, e.g., CHERI-enhanced processors. The CHERI capability instruction-set extension promises proven architectural guarantees for memory safety and pointer provenance. However, superscalar and out-of-order CHERI implementations will need to contend with microarchitectural transient-execution side-channel attacks. To ensure the safety of all CHERI implementations, we articulate CSC: a universal architectural speculation contract for the CHERI architecture that maintains key capability invariants in speculation. We then develop tests against sub-classes of CSC, and discover violations in CHERI-Toooba that lead to a new class of transient-execution attacks. We then develop strategies to fully enforce CSC in CHERI-Toooba. We find that simplistic, strong enforcement incurs a low performance overhead of only 3.43% in SPECint2006 benchmarks, with promise for more optimal designs in the future.

Introduction

Despite transient-execution attacks being identified on Intel, AMD, and some Arm cores in 2018, the majority of the recent Apple M series processors are vulnerable to a recently discovered set of transient-execution attacks [1]. The speculation features causing these attacks are not specific to Apple processors, but can be commonly found on industry processors including RISC-V implementations. Secure architectures are an emerging field and see increasing interest within RISC-V. A prominent example is CHERI, which is currently on the path to be ratified[1] as an extension in RISC-V.

CHERI (Capability Hardware Enhanced RISC Instructions) extends instruction sets with unforgeable, bounded pointers [2]. CHERI instruction sets constrain each memory access to the intended object. Since 2017, researchers have repeatedly demonstrated transient-execution attacks bypassing traditional architectural security mechanisms such as privilege rings and address space separation, and also specialized security architectures such as pointer authentication and enclaves [3]. It is abundantly clear that superscalar, out-of-order CHERI implementations will need to consider how to maintain CHERI guarantees in the face of transient-execution attacks.

A Capability Speculation Contract

To ensure safe, high-performance CHERI implementations, we must define an architectural contract for safe speculation with CHERI capabilities. In CHERI, bounds and permissions checks must not only be safe in non-faulting, in-order execution, but also in transient, faulting execution. CHERI invariants include:

- CHERI capabilities are unforgeable; capabilities are derived only from capabilities of greater or equal privilege.
- Memory can be addressed only through a capability describing and authorizing access to that address.

The first requirement is naturally enforceable in speculation, as pipelines generally forward values that are legitimately calculated from register state. The second requirement is also naturally enforceable, as capability metadata is bundled with the address and can be verified before issuing requests to memory. These two requirements together give rise to a powerful emergent property we call the *Capability Speculation Contract* (CSC):

Capability Speculation Contract (CSC)

All instruction and data-memory accesses issued in speculation must be authorized by capabilities either:

1. in the committed register file;
2. in memory transitively reachable through 1.

In other words, a CHERI processor should act – even in speculation – only with rights transitively reachable from its architecturally committed register file. CSC does *not* forbid speculation on capabilities, but it does forbid using speculatively manipulated capabilities that cannot be found in the architectural register file

[1] Specification draft: `https://github.com/riscv/riscv-cheri`

and its transitive closure. CSC obviates side-channel concerns by preventing memory accesses to illegal addresses from being issued. This approach prevents illegal data from entering the core before it might be exfiltrated by a side-channel.

Hardware Verification

In order to test RISC-V CHERI processors, we have developed a test generator that targets CSC violations. This generator arranges for the data-cache miss counter to indicate accesses not authorized by capabilities in the committed register file. Each sequence starts with a full reset, which clears all caches. A prelude then prepares a capability granting access to a single word of memory, and loads that word. Henceforth, any cache misses will indicate a memory access not allowed by this capability. A random stream of capability instructions is then fed to the processor, followed by a read of the data-cache miss counter. If the counter shows unexpected misses, a violation is reported. The generator finds multiple different counterexamples of which the following is most egregious:

```
lb  r11 ,  c10[0]
cbld  c10 ,  c12 ,  c13
lb  r11 ,  c10[0]
```

While the first `lb` is waiting to commit, *cbld* transiently constructs a valid capability from an untagged value *and loads through that capability*. The load through the forged capability will miss the L1 data cache and thus increase the miss counter indicating a violation.

Mitigation Techniques

Apart from the violations discussed above, our generators also found violations caused by speculative PCCs (Program Counter Capabilities), which lead to dangerous attack scenarios. If we allow only those instructions to execute that lie within the PCC written by the latest-executed branch, then an implementation will comply with CSC. We call this implementation strategy *SinglePCC*. We removed bounds from all program-counter state everywhere in the pipeline and replaced them with a single PCC register. Any logic in the pipeline that needs the complete PCC simply appends the bounds from the PCC register to the current instruction address, thus speculating that the bounds of PCC have not changed. Any jump to a capability with different bounds will trigger a flush at the Commit stage to ensure all older instructions commit with the correct bounds. We verified that this design eliminates known instruction-related CSC vulnerabilities using our generators.

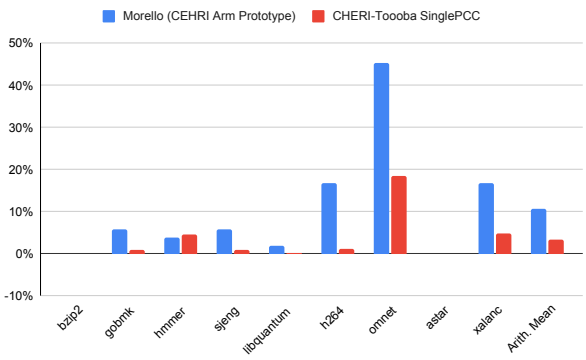

Figure 1: *SPECint2006 cycle overhead of Morello bounds forwarding and CHERI-Toooba with SinglePCC over full-prediction baseline.*

We compare with Arm's Morello prototype implementation of CHERI, which takes a version of the bounds-forwarding approach, but in a superscalar out-of-order pipeline. Unfortunately, PCC readers in Morello must block until the previous jump is executed, causing delay in many common cases.

Figure 1 compares the cycle overhead of SinglePCC, which fully enforces instruction-CSC, against Morello's bounds forwarding.

Conclusion

We have articulated the Capability Speculation Contract (CSC) that precludes transient-execution attacks against CHERI protection, and have demonstrated complete enforcement at a 3.43% performance loss in a superscalar, out-of-order implementation, with hope for further optimizations in the future. Our discovery of CSC violations in the CHERI-Toooba implementation, resulting in the Meltdown-CF vulnerability, is ample proof that such a clearly defined and testable contract is necessary to develop safe superscalar out-of-order CHERI processors. This work has paved the way for a standardized RISC-V CHERI extension (ratification pending) to encourage implementations that are safe from Meltdown-CF.

References

[1] Jason Kim et al. "FLOP: Breaking the Apple M3 CPU via False Load Output Predictions". In: *To appear: 34th USENIX Security Symposium (USENIX Security 25)*. Seattle, WA, USA: USENIX Association, Aug. 2025.

[2] Robert N. M. Watson and et al. *Capability Hardware Enhanced RISC Instructions: CHERI Instruction-Set Architecture (Version 9)*. Tech. rep. UCAM-CL-TR-987. University of Cambridge, Computer Laboratory, Sept. 2023. DOI: 10.48456/tr-987.

[3] Paul Kocher and et al. "Spectre Attacks: Exploiting Speculative Execution". In: *2019 IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2019, pp. 1–19.