

Evaluating the Vulnerability of RISC-V CPUs Against Cache Timing Attacks

Alexandros Ntyrkai*, Dimitris Gizopoulos, Vasileios Karakostas
University of Athens, Greece

Abstract

Assessing the vulnerability of caches against side-channel attacks is of critical importance when enhanced microarchitectural security is a primary requirement for multicore CPU implementations. Previous works have proposed various metrics and methodologies to assess such vulnerabilities. However, those works suffer from limitations regarding either the range of target cache attacks, the support for the RISC-V ISA, or the reliance on in-house simulators and tools. The goal of this paper is to provide support for systematically evaluating RISC-V multicore CPUs against a wide range of cache timing attacks. Our approach enables the assessment of both hardware implementations and simulated systems, facilitating early-stage security evaluation during the processor design phase using the open-source gem5 microarchitectural simulator. Specifically, we extend the Cache Timing Vulnerability Score (CTVS) methodology along two axes. We port CTVS to RISC-V, and then we integrate it with gem5 to support both RISC-V and x86 simulated systems. Finally, we evaluate our enhanced CTVS framework on simulated RISC-V and x86 multicore CPUs to demonstrate its effectiveness.

Introduction

The microarchitectural side-channel attacks exploit side-effects that occur at the level of the microarchitecture by leveraging the otherwise functionally correct behavior of CPU components to leak sensitive information. The side-channel attacks can be classified into various categories based on the side-channel that they use or the microarchitectural behavior that they exploit. In this paper, we focus on cache timing attacks. In this type of attacks, the victim and the attacker are running on the same multicore CPU, and the victim process performs secret-dependent memory accesses, while the attacker process tries to identify a secret memory address or index that is used by the victim process and eventually infer the secret.

Various metrics and methodologies have been proposed to quantify the vulnerability of multicore CPUs against cache timing side-channel attacks. However, these approaches either quantify the vulnerability of caches against a subset of attacks, lack support for the RISC-V ISA, or rely on in-house simulators and tools.

Our goal in this paper is to provide support for systematically assessing RISC-V multicore CPUs against microarchitectural cache timing attacks. Our approach

allows for the evaluation of both hardware implementations and simulated systems, enabling early security evaluation in the design phase of the processor.

To achieve this goal, we base our approach on the *Cache Timing Vulnerability Score (CTVS)* [1] that has been proposed to systematically quantify the vulnerability of x86 and Arm multicore CPUs¹ against cache side-channel attacks, and enhance it in two ways. First, we port the CTVS methodology to the RISC-V ISA. Our porting effort consists primarily of the manual conversion of x86 assembly to RISC-V assembly, addressing various challenges with respect to ISA dependencies. Second, we enable the execution of CTVS for both RISC-V and x86 ISAs within the open-source gem5 microarchitectural simulator.

In summary, our contributions enable: (i) the security evaluation of both real and simulated RISC-V systems, (ii) the comparison of processors with different ISAs against cache attacks, and (iii) the assessment of mitigation mechanisms against cache attacks.

Background

The CTVS methodology [1] consists of a benchmark suite that is used to compute the CTVS score. The benchmark suite includes 1094 automatically generated tests that cover 88 strong theoretical vulnerabilities. The CTVS score quantifies the vulnerability of a cache design against cache timing attacks in the L1 data cache. A smaller CTVS means that the CPU is more secure.

¹ Note that concurrent work [2] has also ported CTVS to RISC-V, but lacks support for simulated systems with gem5.

*Corresponding author: antyrkai@di.uoa.gr

This work was supported by the European Union's Horizon Europe research and innovation programme under grant agreement No 101097224 (REBECCA), and by the framework of H.F.R.I call "3rd Call for H.F.R.I.'s Research Projects to Support Faculty Members & Researchers" (H.F.R.I. Project Number: 26508 - COMPLETEVM). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

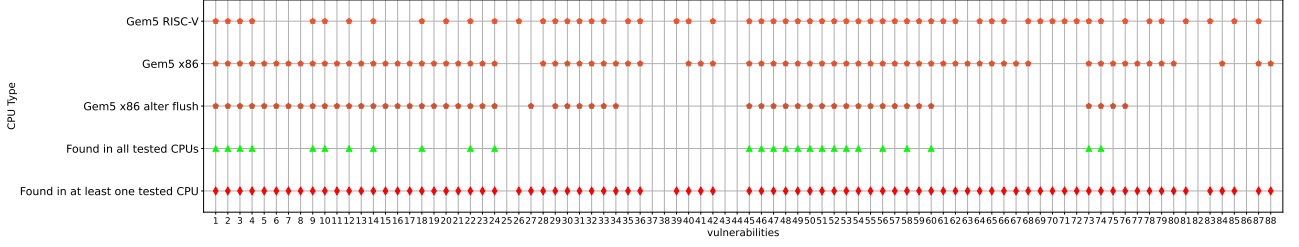


Figure 1: CTVS results for the 88 strong theoretical types of attacks on the simulated RISC-V & x86 systems.

Table 1: Scores for the simulated RISC-V & x86 systems.

	RISC-V	x86	x86-alt	all CPUs	≥ 1 CPU
CTVS	59/88	71/88	51/88	26/88	81/88

Our Approach

Porting CTVS to RISC-V. First, we had to port the x86 assembly code of the CTVS generator program to RISC-V. More specifically, we modified the assembly code that is responsible for collecting timing measurements to use the `rdcycle` RISC-V instruction, instead of the `rdtsc` x86 instruction. We also transformed all the x86 load and store instructions (i.e., `movq`) to the corresponding RISC-V instructions (i.e., `ld` and `sb`). Furthermore, we changed all `mfence` x86 instructions to `fence` RISC-V instructions. In addition, we modified the registers that are used by each instruction and changed the inline assembly constraints.

Because some of the generated benchmarks use the `clflush` x86 instruction and the RISC-V ISA lacks such support, we created a new class of tests by transforming a flush operation into a prime operation. More specifically, we leverage the `CacheFlush()` function [3] that considers the cache organization and evicts the requested memory address by introducing multiple loads to different memory addresses that are mapped to the same cache set with the requested address.

Integrating CTVS with gem5. We integrated the CTVS methodology with gem5 using full-system simulation. We statically compiled the benchmarks and copied the generated binaries to the disk image of the simulated system. Our initial effort to execute the benchmarks in gem5 failed to generate any vulnerabilities. We realized that there was an issue with the `Pvalue()` function that computes the Welch’s t-test to identify vulnerabilities due to timing differences. To address that issue we replaced `lgamma1()` with `lgamma()` in the implementation of `Pvalue()`, and managed to observe vulnerabilities. Finally, we created scripts that automate the execution of the CTVS methodology in gem5 by running multiple simulation experiments in parallel for different tests.

Evaluation

To evaluate the use of CTVS in gem5, we simulate 4-core systems based on the RISC-V and x86 ISA. We use full-system simulation, with the Timing CPU model and classic caches. For the x86 CPU we report two sets of results: “gem5 x86” refers to the original tests that include the `clflush` instruction when necessary, while “gem5 x86 alter flush” refers to the modified tests that use the `CacheFlush()` function.

Figure 1 shows the preliminary results for the 88 types of attacks on different machines. A dot means the corresponding processor is vulnerable to that type. We observe that the RISC-V tests have less effective cases than those corresponding to the x86 tests with `clflush`, but more effective cases than the x86 tests with `CacheFlush()`. This behavior is also depicted in the CTVS score (Table 1). Overall, the CTVS score of the RISC-V tests (i.e., 59/88) is lower than that of the x86 tests with `clflush` (i.e., 71/88) and higher than that of the x86 tests with `CacheFlush()` (i.e., 51/88). Finally, most RISC-V tests indicate vulnerabilities that are also triggered by x86 tests, although there are some differences.

Conclusions & Future Work

In this paper, we ported the CTVS methodology to RISC-V and integrated it with gem5. Our contributions enable the systematic security assessment of both real and simulated RISC-V systems. As future work, we plan to implement mitigation mechanisms against cache side-channel attacks in gem5 and evaluate their effectiveness with the enhanced CTVS, and perform a comprehensive security comparison across ISAs.

References

- [1] Shuwen Deng, Wenjie Xiong, and Jakub Szefer. “A Benchmark Suite for Evaluating Caches Vulnerability to Timing Attacks”. In: *ASPLOS 2020*.
- [2] Cédric Austa, Jan Tobias Mühlberg, and Jean-Michel Dricot. “Systematic Assessment of Cache Timing Vulnerabilities on RISC-V Processors”. In: *ESORICS 2025*.
- [3] <https://github.com/riscv-boom/boom-attacks>.