A Hardware-Based Cache Side Channel Attack Detection Mechanism for RISC-V Processors

Andreas Brokalakis[1,2]*, Alexandros Skyvalos[1], Sotiris Ioannidis[1], Iakovos Mavroidis[1] and Ioannis Papaefstathiou[2]

[1]School of Electrical and Computer Engineering, Technical University of Crete, Greece
[2]Exascale Performance Systems PLC, Greece

Abstract

Side-channel attacks rely on information that can be gathered (or leaked) by the fundamental way a computer system operates. For CPU-based systems, a prominent number of these attacks target the caches aiming at gaining unauthorised access to sensitive data. Proposed solutions provide limited remedies and come at great costs, especially associated with the difficulty to reliably identify such attacks. In this work, we analyse cache-based side channel attacks on RISC-V processors and demonstrate that they all depend on accessing specific architectural registers to successfully complete. We present a detection mechanism implemented at the hardware level that is able to detect all such attacks, without producing false negative detections and without requiring any software assistance or modifications.

Introduction

Given enough time, an attacker will discover and exploit the vulnerabilities of any static computing system. Even worse, in the absence of apparent design flaws or implementation weaknesses, attackers may be able to extract secrets or gain access to systems, by observing their operation through side channels (e.g., by timing the completion of a task). In fact although such attacks were proposed decades ago, every year an exploding number is reported [1].

No processors are immune to such attacks, including those based on the RISC-V architecture. Defense mechanisms need to be implemented in hardware to be really effective [2] or at least incur non-detrimental performance and compatibility issues. In practice, one of the key difficulties has been the reliable detection of an ongoing side channel attack, which forces mitigation solutions to degrade performance or limit access to processor features for all running processes. In this work, we explore how cache-based side channel attacks are launched on RISC-V processors and propose a purely hardware based detection mechanism. We employ an open source RISC-V processor (CVA6 [3]) and construct a monitoring scheme that is able to detect with confidence the execution of an attack, making it possible to relay this information to any hardware-based defense mechanism that can effectively mitigate the attack.

Background

In a modern processor, several microarchitectural resources (caches, TLBs, branch predictors and others) are shared among programs. These shared resources can leak information either as part of their fundamental design (e.g. a cache miss will lead to longer data read time than a cache hit) or because there may be flaws in their implementation or optimization tricks [2]. This information leakage may be observed and exploited by processes that run in the user space.

Cache-based side channel attacks (SCAs) are particularly important. Not only they are very serious attacks on their own, they constitute the underlying layer of most side-channel attacks. For example, the famous Meltdown and Spectre attacks that impact almost all modern CPUs, take advantage of weaknesses in the out-of-order and speculative execution hardware respectively but use a well-know cache-based attack method to complete their work.

Typically, cache SCAs are based on the principle that the access time of a particular cache line depends on whether it has been accessed by a victim process or not. Prime+Probe [4] attacks prime cache sets with arbitrary data, then lets a victim program run and, finally, accesses the cache sets again. A higher access time indicates that the victim program also filled the same sets and, therefore, the attacker can identify them. A lot of variations of this basic concept have been reported (e.g. Flush+Reload and Evict+Reload).

Different defense mechanisms have been proposed as countermeasures against cache side-channel attacks at the hardware level [1]. These solutions mainly focus on isolation and obfuscation. The former approaches fragment the cache and limit the cache capacity available to a given process, while the latter introduce added latency to cache accesses so that hits and misses cannot be distinguished clearly. Significant performance, area and power trade-offs are associated with all these solutions, while in most

*Corresponding author: abrokalakis@tuc.gr

cases changes are required to be made to the system and application software to render them effective. It is important to note that these defense mechanisms affect equally all processes running on a CPU - both legit and malicious ones. This comes as a result of the difficulty to reliably distinguish between those two kinds of processes and therefore the impact in performance affects them all.

Detecting Attacks at Hardware Level

We employed an open-source RISC-V processor design (CVA6 [3]) and successfully implemented a working system on an FPGA development board (Digilent Genesys 2) that booted and executed a Linux operating environment. Starting from the initial work of Martinoli et al [5], we were able to construct a trojan based on the architectural and implementation information of the available CVA6 source code, which enabled us to execute multiple cache-based side channel attacks on its L1 data cache (such as Prime+Probe and Evict+Reload).

We performed an analysis of the trojan to identify critical aspects of its success. We noticed that it required precise timing measurements to determine if an element is located inside the cache or not. To accomplish this, it employed low-level functions that access directly special Control Status Registers (CSR) of the RISC-V architecture, such as *rdcycle*. The latter returns the number of clock cycles executed by the processor's core. We replaced these calls with more high-level timers such as those provided by the Linux OS that we were running on top of the CVA6 processor. However this resulted in failure of the attacks, making it evident that the information provided by the CSR registers was crucial.

At the hardware level, we added a counter inside the CVA6 core to measure the amount of accesses to *rdcycle*. Using this counter, we monitored the overall accesses to the specific CSR in multiple occasions to determine access patterns and whether the OS and legitimate applications also rely heavily on this register. We executed the overall Linux stack, typical applications running on top of the processor, benchmarks provided by the RISC-V and CVA6 repositories and finally the trojan carrying out the attacks. On all legitimate operations, we measured that the frequency of accesses to the specific CSR is orders of magnitude lower than what is observed during a side channel attack. As such, we determined that performing such measurements provided a safe mechanism to detect with confidence the execution of an attack.

Indeed, with the above implementation we are able to detect 100% of the attacks executed on the core. The detection mechanism is not susceptible to false negatives, however it allows for a very low number of false positive detections. This is a side effect of the use of a counter and associated thresholds. It should be noted though, that because of the huge difference between normal number of rdcycle accesses and access during an attack, these false positives can only appear extremely rarely and after very long periods of time. They also do not compromise the security of the system, as no attacks can pass undetected.

The implementation cost of the detection mechanism is negligible in terms of resources in the tested FPGA system (Xilinx Kintex 7). Additionally, the added hardware does not affect any critical paths of the design and therefore it does not affect system clock. It should be noted that although it has been tested on the CVA6 processor, its use is general and can be added to any RISC-V processor that implements and exposes the specific CSRs.

Conclusions & Future Work

This work presents a cache SCA detection mechanism for RISC-V processors. By measuring accesses to specific CSRs, it can distinguish attacks with 100% confidence. It is implemented at the hardware level requiring negligible resources. It is completely transparent to the software layers and does not introduce any performance penalties. Future work will extend the detection scheme with a mitigation mechanism to provide a complete hardware-based defense solution.

References

[1] C. Su and Q. Zeng. "Survey of CPU cache-based side-channel attacks: systematic analysis, security models, and countermeasures". In: *Security and Communication Networks* (2021).

[2] G. Dessouky et al. "With great complexity comes great vulnerability: from stand-alone fixes to reconfigurable security". In: *IEEE Security & Privacy* 18.5 (2020), pp. 57–66.

[3] *CVA6 RISC-V CPU*. URL: `https://github.com/openhwgroup/cva6`.

[4] D.G. Osvik, A. Shamir, and E. Tromer. "Cache attacks and countermeasures: the case of AES". In: *Topics in Cryptology–CT-RSA 2006: The Cryptographers' Track at the RSA Conference 2006, San Jose, CA, USA, February, 2005.* Springer. 2006.

[5] Valentin Martinoli et al. "CCALK: (When) CVA6 Cache Associativity Leaks the Key". In: *Journal of Low Power Electronics and Applications* 13.1 (2023).