SpecDoctor: Differential Fuzz Testing to Find Transient Execution Vulnerabilities

Jaewon Hur
Seoul National University
hurjaewon@snu.ac.kr

Suhwan Song
Seoul National University
sshkeb96@snu.ac.kr

Sunwoo Kim
Samsung Research
sunwoo28.kim@samsung.com

Byoungyoung Lee*
Seoul National University
byoungyoung@snu.ac.kr

ABSTRACT

Transient execution vulnerabilities have critical security impacts to software systems since those break the fundamental security assumptions guaranteed by the CPU. Detecting these critical vulnerabilities in the RTL development stage is particularly important, as it offers a chance to fix the vulnerability early before reaching the chip manufacturing stage.

This paper proposes SpecDoctor, an automated RTL fuzzer to discover transient execution vulnerabilities in the CPU. To be specific, SpecDoctor designs a fuzzing template, allowing it to test all different scenarios of transient execution vulnerabilities (e.g., Meltdown, Spectre, ForeShadow, etc.) with a single template. Then SpecDoctor performs a multi-phased fuzzing, where each phase is dedicated to solve an individual vulnerability constraint in the RTL context, thereby effectively finding the vulnerabilities.

We implemented and evaluated SpecDoctor on two out-of-order RISC-V CPUs, Boom and NutShell-Argo. During the evaluation, SpecDoctor found transient-execution vulnerabilities which share the similar attack vectors as the previous works. Furthermore, SpecDoctor found two interesting variants which abuse unique attack vectors: *Boombard*, and *Birgus*. *Boombard* exploits an unknown implementation bug in RISC-V Boom, exacerbating it into a critical transient execution vulnerability. *Birgus* launches a Spectre-type attack with a port contention side channel in NutShell CPU, which is constructed using a unique combination of instructions. We reported the vulnerabilities, and both are confirmed by the developers, illustrating the strong practical impact of SpecDoctor.

CCS CONCEPTS

• **Security and privacy** → **Side-channel analysis and countermeasures**;

KEYWORDS

transient-execution vulnerability; fuzzing; differential testing

ACM Reference Format:
Jaewon Hur, Suhwan Song, Sunwoo Kim, and Byoungyoung Lee. 2022. SpecDoctor: Differential Fuzz Testing to Find Transient Execution Vulnerabilities. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22), November 7–11, 2022, Los Angeles, CA, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3548606.3560578

1 INTRODUCTION

Transient execution vulnerabilities are critical vulnerabilities in modern CPUs. Since the first disclosure of Spectre [1] and Meltdown [2], many other transient execution vulnerabilities have been discovered—ForeShadow [3], RIDL [4], FPVI [5], Cacheout [6], etc. While such vulnerabilities critically harm the security of software on the affected CPU, vendors were not able to quickly release the patch as these are rooted in the micro-architecture of the CPU.

The root cause of these vulnerabilities is highly related to the speculative execution, which originally intended to maximize the CPU performance. Specifically, CPU attempts to predict the result of the earlier instruction, then execute the later instruction speculatively under the assumption that the prediction was correct [7]. However, if the prediction was wrong, CPU rollbacks the execution of the later instruction to preserve the execution correctness. Here, the rollbacked instructions are called transient instructions. As this rollback is not visible from the architectural point of view, transient execution does not harm the execution correctness of CPU—it only changes the micro-architectural states. The problem is that various attack techniques have been discovered to learn the traces in the micro-architectural states due to the transient instructions. Since these transient instructions are the artifact of the wrong prediction and thus should not be executed, execution results of transient instructions may include security sensitive data, violating the fundamental security isolation guarantees of CPU.

While there have been several previous approaches to automatically detect these vulnerabilities on the off-the-shelf CPUs (i.e., CPUs that are already manufactured) [8, 9], approaches detecting in the RTL development stage have not gained much attention. Detecting in the RTL development stage is particularly important because it offers a chance to fix the vulnerability early, even before the CPU chip is manufactured and released. Once it is released, it becomes extremely difficult to fix as the vulnerabilities are hardwired in the

*Corresponding author

CCS '22, November 7–11, 2022, Los Angeles, CA, USA.

ACM ISBN 978-1-4503-9450-5/22/11…$15.00
https://doi.org/10.1145/3548606.3560578