SecSMT: Securing SMT Processors against Contention-Based Covert Channels

Mohammadkazem Taram, *University of California San Diego;* Xida Ren and Ashish Venkat, *University of Virginia;* Dean Tullsen, *University of California San Diego*

https://www.usenix.org/conference/usenixsecurity22/presentation/taram

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 31st USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 31st USENIX Security Symposium.

August 10–12, 2022 • Boston, MA, USA

978-1-939133-31-1

Open access to the Artifact Appendices to the Proceedings of the 31st USENIX Security Symposium is sponsored by USENIX.

A Artifact Appendix

A.1 Abstract

This artifact describes the frameworks used for our evaluations. The artifact consists of two relatively separable components: (1) a covert channel discovery framework that runs directly on real hardware, and (2) a gem5 simulation infrastructure that evaluates our mitigation strategies. The covert channel framework can be used to replicate the results presented in Section 4, and in particular, the results summarized in Table 1. The gem5 infrastructure can be used to replicate the performance and security results presented in Section 7.

A.2 Artifact check-list (meta-information)

- **Programs:** SPEC CPU2017, SunSpider JS Benchmarks, WolfSSL RSA and AES benchmarks.
- **Compilation:** LLVM with -O3 for SPEC.
- **Hardware:** AMD Ryzen Threadripper 3960X and Intel Core i7-6770HQ for Covert Channel Framework. The gem5 simulator runs on any modern x86 hardware.
- **Run-time Environment:** The provided covert channel framework and gem5 simulator are tested on Ubuntu 20.04. The scripts for running SPEC benchmarks on gem5 assume an available Slurm workload manager.
- **Output:** The covert channel framework outputs the covert channels bandwidth and error rate. The gem5 simulator outputs the execution time and performance in terms of Cycles Per Instruction (CPI).
- **Experiments:** Scripts and instructions are provided in the artifact README files.
- **How much disk space required (approximately)?:** About 250 GB of disk space is required for the SPEC 2017 simpoints, and around 5 GB is needed for the gem5 code and binaries. The covert channel framework requires less than 100 MB.
- **How much time is needed to prepare workflow (approximately)?:** About 30 minutes to download the frameworks and install requirements, and around 30 minutes to compile gem5.
- **How much time is needed to complete experiments (approximately)?:** Assuming enough available parallelism, the gem5 experiments need at least 3 hours. The covert channel measurements require about 2 hours to complete.
- **Publicly available?:** Yes, the code is available on Github (see Section A.3.1).
- **Code licenses:** GPL v3.

A.3 Description

A.3.1 How to access

The artifact is available on github at the following URL: https://github.com/mktrm/SecSMT_Artifact/tree/86286e06f6f1d8ce9583af950edacb87f14e39ba.

A.3.2 Hardware dependencies

A bare-metal machine is required to run the covert channel measurements. The bandwidth of the covert channels are measured on two specific processors: Intel Core i7-6770HQ and AMD Ryzen Threadripper 3960X. While the covert channel framework can be adapted for other processors, it requires extensive parameter fine-tuning to achieve the best channel.

A.3.3 Programs

We provide Simpoints created from SPEC 2017 benchmarks for the artifact evaluators, but we cannot publish them as they are under copyright. Other programs used for evaluations are publicly available.

A.4 Installation

The artifact provides scripts to install requirements as well as building the provided tools.

A.5 Experiment workflow

This section provides a high-level overview of the experimental workflow. Please follow the instructions in the README for a detailed, step-by-step guide.

The covert channel framework needs to first install a kernel module that facilitates reading values for performance counters. Note that those performance counter values are only used for debugging purposes and our covert channel communication is entirely based on execution time (cycle time). To make a fair comparison between the covert channels, we make sure the processors are configured to be always on performance mode (scripts are provided). Then, the provided makefile detects the hardware (Intel or AMD) and compiles the covert channel measurement codes for all the available covert channels for that platform. It then runs multiple rounds of the experiments for each channel and reports the average bandwidth and error rate of all successful channels (if the error rate is less than 10%).

For the evaluation of the mitigation strategies, we need to first compile the gem5 code and prepare our benchmark programs. Then, we run multiple gem5 simulations for each pair of benchmark programs. Each experiment is configured to represent one of the following multithreading approaches: (1) a fully dynamically shared insecure baseline, (2) a fully statically partitioned pipeline, (3) our Adaptive partitioning, and (4) our Asymmetric SMT approach in which we apply Asymmetric SMT on top of adaptive partitioning for some resources. Finally, once the simulations are finished, we run the provided scripts to extract the results from gem5 simulations and draw the figures.

A.6 Evaluation and expected results

A successful run of the covert channel framework will result in a set of bandwidth and error rate pairs. If the measurements

take place on the mentioned processors, the bandwidth numbers should be around the results reported in Table 1. Note that the fluctuation in bandwidth and error rate numbers can be caused by various sources of noise such as voltage/frequency scaling, OS scheduling, etc.

The gem5 simulation of the mitigation strategies should result in performance numbers that match those presented in the paper.