Double Trouble: Combined Heterogeneous Attacks on Non-Inclusive Cache Hierarchies

Antoon Purnal, Furkan Turan, and Ingrid Verbauwhede, *imec-COSIC, KU Leuven*

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

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

G Artifact Appendix

G.1 Abstract

The artifacts consist of various cache-timing experiments. The initial experiments help to understand the non-inclusive LLC structure and DDIO-based accesses to it. The later experiments allow reverse-engineering the underlying mechanisms, exploiting the findings and performance analysis.

As our work explores the use of FPGAs for cache side-channel attacks, the artifacts require an FPGA acceleration card. Though we used our local setup with Intel Programmable Accelerator Card (PAC), we also tested the artifacts with remote access to Intel Labs (IL) Academic Compute Environment (ACE) (https://wiki.intel-research.net/FPGA.html).

For people who have neither, we provide detailed documentation for each experiment, which provides the expected execution output and the inferences we draw from them. **Notably, Figures 3, 4, 7, 8 and 9 were produced based on experiments like these.**

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

- **Algorithm:** SW+HW Combined Cache Attacks, Eviction Set Construction, Reverse-Engineering DDIO
- **Compilation:** `make`, `gcc` (tested versions are 7.5.0 and 4.8.5)
- **Binary:** For software: binaries are not provided. Not needed as compilation is straightforward. For hardware: A bitstream file synthesised for Intel Arria 10 PAC is provided. This should save quite some time as synthesis may take several hours. Instructions on how to synthesise for other PACs are available.
- **Run-time environment:** `Ubuntu 18.04.5 LTS` and `CentOS Linux 7.7.1908` The basic experiments do not require any root access. However, experiments on non-default system configurations and reverse engineering require `sudo`. We included these experiments as they were requested by our reviewers.
- **Hardware:** Intel PAC (Programmable Accelerator Card)
- **Execution:** Compilation and execution of binaries on command line.
- **Security, privacy, and ethical concerns:** Demonstrates cache attacks allowing to steal secrets of victims sharing the same computer as attacker.
- **Metrics:** Cache access timings, execution time, accuracy.
- **Output:** Console output with exact numerical results.
- **Experiments:** The repository is a collection of various experiments, such as eviction set construction, cache access timings, eviction candidate determination, effect of shared access over the eviction candidate, evidence for DDIO$^+$ region, etc. A separate documentation file is provided for each experiment.
- **How much disk space required:** The repository is 13 MB. When de-compressed, the bitstream file reaches has a size of 133 MB. All in all, it is less than 150 MB.
- **How much time is needed to prepare workflow:** Almost no additional preparation is needed if you already have access to an Intel PAC0based FPGA-accelerated computation server. Otherwise, an access request to *Intel Labs Academic Compute Environment* (https://wiki.intel-research.net/FPGA.html) and getting a response might take at least a day.
- **How much time is needed to complete experiments:** It can take several hours.
  Suppose evaluators do not have access to an Intel PAC based FPGA accelerated computation server. In that case, they can go through the documentation files and observe the expected execution results and the inferences drawn from them. This option would take an hour.
- **Publicly available:** https://github.com/KULeuven-COSIC/Double-Trouble
- **Code licenses:** MIT License
- **Archived:** https://github.com/KULeuven-COSIC/Double-Trouble/tree/ArtifactsAvailable

G.3 Description

The repository contains a set of experiments, listed in the following table along with the relevant section of the paper.

Experiment	Figure/Section
Basic Functionality	
Shared Access	Figure 4d
CPU Read	Figure 4f
Secondary Write	Figure 4e
CPU Write	Figure 4c
Cache Timing Histogram	Appendix A
Eviction Candidate	Section 3.2.2
DDIO Replacement Policy	Appendix C
Eviction with Reduced EvSet	Section 6.2
EvSet Const	Section 8.1.1
Reverse Engineering of DDIO	Section 5

The first experiment is provided as a warm-up to the basic API usage, the building of the attacker-victim framework, eviction set creation, and cache-timing measurements. For each experiment, a dedicated documentation file is provided. This file explains the conducted experiment, how to execute it, and the expected results.

G.3.1 How to access

The artifacts are published on GitHub. The version, improved with evaluators' suggestions, is tagged as 'ArtifactsAvailable'.

The Stable URL is: https://github.com/KULeuven-COSIC/Double-Trouble/tree/ArtifactsAvailable

We wish to keep the repository solely for the artifacts of the paper, so the most recent commit should always apply for this paper.

G.3.2 Hardware dependencies

Our work requires an FPGA acceleration card. Specifically, we tested with the following Intel PACs (Programmable Accelerator Cards):

- Intel Arria 10 PAC
- Intel Stratix 10 PAC

One can either have a local setup that employs an Intel PAC or work with remote access to such a platform. We did both. For the latter, Intel Labs (IL) Academic Compute Environment (ACE)

(https://wiki.intel-research.net/FPGA.html) provides various remotely accessible platforms.

For people who do not have access to an FPGA accelerated platform but are still interested in our findings, we provided detailed documentation on our repository. The documentation includes a separate file for each experiment, where we provide an example output of the execution for proving the claims we made out of them.

G.3.3 Software dependencies

On a platform with Intel PAC, we can assume that the corresponding Intel OPAE SDK (https://github.com/OPAE) is available.

The compilation is straightforward with `Makefile`s. We tested on two different setups with different `gcc` versions:

- On our local setup: Ubuntu 18.04.5 LTS with `gcc` 7.5.0
- On Intel Labs ACE: CentOS Linux 7.7.1908 with `gcc` 4.8.5

A few experiments require installing additional libraries. Instructions to install them are available for each repository, though they need root privileges on the machine.

- `intel-cmt-cat` https://github.com/intel/intel-cmt-cat is used for the experiments in Section 5. This library allows to fix the LLC ways used by a CPU core.
- `intel-msr-tools` https://github.com/intel/msr-tools is used for changing the default cache configuration, with the purpose of giving the FPGA (or DDIO in general) broader cache access. Experimenting with these non-default configurations was a suggestion by the reviewers.

G.3.4 Security, privacy, and ethical concerns

Our work demonstrates a timing-based cache side-channel framework, aiming for the disclosure of our security and privacy concerns associated with the underlying platforms.

G.4 Installation

Cloning the repository is adequate for obtaining the source files.

G.5 Experiment workflow

All experiments consist of two steps; compilation with a provided Makefile and execution of the generated binary. Depending on the platform, there can be various customizations, e.g., pinning processes to specific CPU cores.

Each experiment in the repository comes with a separate documentation file. These documents provide experiment-specific compilation (with `Makefile` targets) and execution commands, besides the experiment explanation and expected outputs.

G.6 Evaluation and expected results

The evaluation is divided into multiple experiments. The initial experiments help understand the non-inclusive LLC structure and its DDIO-based access. The later experiments entail:

- reverse engineering the underlying mechanisms,
- demonstrating the findings, and
- analyzing the performance.

G.6.1 Figure 4

We have four experiments respectively for the observations provided in Figures 4c, 4d, 4e and 4f. The expected results are timing measurements that support the claimed observations in Figures 4 and Section 3.2.

G.6.2 Cache Timing Histogram

This experiment measures the timings for cache line accesses from various levels in the cache hierarchy and constructs a histogram that helps to distinguish accesses by their latency. The expected result is a histogram similar to Figure 9 given in Appendix A, though the timings can vary on different platforms.

G.6.3 Eviction Candiate

This experiment determines whether DDIO reads and writes are recorded by the cache replacement policy, i.e., whether they change the eviction candidate. The expected results consist of cache timing measurements prooving the claims made in Section 3.2.2 including Figure 3.

G.6.4 DDIO Replacement Policy

This experiment performs various access patterns with DDIO lines, checks the cache contents after these accesses, and compares them with the expected contents for different replacement policies. The expected result is the re-construction of Table 10.

G.6.5 Reduced Eviction

This experiment implements the eviction with the reduced-eviction approach explained in Section 6.2. It creates random bits, indicating whether the victim accesses the target address or not. The attacker monitors the victim's activity and determines whether the victim has accessed the target. The expected results are the measures indicating the success of eviction with a reduced eviction set.

G.6.6 Eviction Set Construction

This experiment is used to evaluate the eviction set construction performance for various configurations, which are:

- Non-default DDIO way settings
- Huge or small pages
- Different levels of stress
- Options of congruence checks integrated into the eviction set construction

The expected results comprise a debug log of FPGAs eviction set construction e.g., how many guesses were needed for every congruent address, total construction time, and failed attempts. These results are used in Section 8.1.1 and to construct Figure 8.

G.6.7 Reverse Engineering the DDIO and DDIO+ regions

This experiment is used to reproduce the findings in Section 5. The expected results indicate the ways available to DDIO and DDIO$^+$ allocations, hence allowing to redraw Figure 7.

G.7 Experiment customization

The experiments offer limited customizability and interactions, including picking the associations of processes to specific CPUs.

Specifically for the *Eviction Set Construction* experiment, various measurements, indicated in Appendix G.6.6, can be perfomed by customizing the command line arguments to the binary.

G.8 Version

Based on the LaTeX template for Artifact Evaluation V20220119.