USENIX'23 Artifact Appendix: The Gates of Time: Improving Cache Attacks with Transient Execution

Daniel Katzman, William Kosasih, Chitchanok Chuengsatiansup, Eyal Ronen, Yuval Yarom

Tel-Aviv University
The University of Adelaide
The University of Melbourne

A Artifact Appendix

A.1 Abstract

We implement a speculative execution attack that improves cache attacks by means of amplifying cache states. In addition to that, the construct that we use to perform this amplification technique is also capable to facilitate robust computation on cache states. This project is available at https://github.com/0xADE1A1DE/GoT. In this Artifact Evaluation, we are applying for:

- "Artifact Available" badge
- "Artifact Functional" badge
- "Results Reproduced" badge

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

A.2.2 How to access

Artifact can be accessed here: https://github.com/0xADE1A1DE/GoT/commit/5bac7ece92f61025b7c1942c59b95e8a999ec231

A.2.3 Hardware dependencies

Amplification, Eviction Set, and Prime + Store Attack
Dynabook TECRA A50- EC, with an Intel(R) Core(TM) i5-8250U CPU
Trace Processing and Stitching
A High Performance Computing (HPC) node with 80 threads and AVX-2 instruction-set support.
Circuit Evaluation
Intel(R) Core(TM) i7-9750H with cores 0,1,6,7 isolated.

A.2.4 Software dependencies

Amplification, Eviction Set, and Prime + Store Attack
Ubuntu 20.04.3 LTS, Chromium commit hash (be87c21d2a7069363dfd66f278739d7e4211145e), emscripten.
Trace Processing and Stitching
A reasonably modern HPC node with Linux.
Circuit Evaluation
Ubuntu 21.04, huge-pages enabled, gnuplot.

A.2.5 Benchmarks

Amplification, Eviction Set, and Prime + Store Attack
None
Trace Processing and Stitching

- **Dataset:** The dataset consists of several components, including the raw traces obtained from the Prime+Store attack on ElGamal, the true key (referred to as the "ground truth") of the ElGamal encryption algorithm, and the frequency analysis of these traces. The latter serves as a guide for selecting the optimal starting trace for key expansion.
- **Location:** The dataset is located in the artifacts repository under the directory: "trace_processing_and_stitching/DATASETS"

Circuit Evaluation

- **Dataset:** The dataset contains raw circuit accuracy data which provides guidance on how to create the figures. Furthermore, scripts that can be used to generate figures that match those in the paper are also included in the dataset.
- **Location:** The dataset can be found in the artifacts repository under the directory: "circuits/FIGURES"

A.3 Set-up

A.3.1 Installation

Amplification, Eviction Set, and Prime + Store Attack

- Install gcc/g++, python3, matplotlib, gnuplot
- Install and activate emscripten

  ```
  git clone https://github.com/emscripten-core/emsdk.git
  cd emsdk
  ./emsdk install latest
  ./emsdk activate latest
  source ./emsdk_env.sh
  ```

- Switch the processor to run in performance mode (meaning the processor is set to its' maximum frequency), by run the following command as superuser ('sudo -s')

  ```
  for i in 0..7; do echo performance > /sys/devices/system/cpu/cpu$i/cpufreq/scaling_governor; done
  ```

- Clone the GoT repo:

  ```
  git clone https://github.com/0xADE1A1DE/GoT.git
  ```

- Get Chromium at commit hash `be87c21d2a7069363dfd66f278739d7e4211145e`

  Apply `/GoT/amplification_eviction_set_finding/wasm/v8.diff`, which does the following:

  - Creates a special native function `%CustomFn`
    - Gives access to `clflush`
    - Memory fences `mfence; lfence`
    - Non functioning virtual to physical address resolution (requires running Chromium without sandbox, and some more tedious setup, this part is not used by our experiments)
  - Alters the assembler to emit `rdtsc` when asked to emit a `mov register, imm` with `imm==0xddaa00ccbb00 || imm==0xddaa00ccbb80` This is used to provide our program with the ability to measure with `rdtsc`.

  Note: The experiment requiring this patch is for Figure 9, we only use the `clflush` and `fences` capabilities provided by this patch.

  Note: Due to the size of Chromium, and the limited capacity of the storage in our system, we suggest using our compilation located at `/home/acrypto/Documents/daniel/out/Default/chrome`

Trace Processing and Stitching

Access to an HPC node with 80 threads and AVX-2 instruction set support helps in speeding up evaluation process.

Circuit Evaluation

Ensure that huge pages are enabled and core 0,1,6,7 (for Core i7-9750H with 12 logical cores, these are core 0,1 and their sibling cores) are isolated. On Ubuntu,

This can be achieved by adding the boot parameter:

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash
isolcpus=0.1,6,7 default_hugepagesz=1G
hugepagesz=1G hugepages=3000"
```

to /etc/default/grub.

Then run the command

```
sudo update-grub
```

Then Install gnuplot. In ubuntu, run

```
sudo apt install gnuplot
```

A.3.2 Basic Test

Circuit Evaluation To test that the program compiles successfully and the environment requirements are met. This test require huge pages to be enabled, refer to appendix A.3.1 on how to enable. Run the program with

```
taskset -c 1 ./out gol_glider_demo
```

This will run the game of life glider demo. Expect a glider to hover across your screen. Note that the process is pinned to core 1, as it is isolated from the rest of the system to reduce noise.

A.4 Evaluation workflow

Amplification, Eviction Set, and Prime + Store Attack

Refer to the readme file in `amplification_eviction_set_finding/README.md` and `prime_store_attack_finding/README.md`

Trace Processing and Stitching

Refer to the readme file in `trace_processing_and_stitching/README.md`

Circuit Evaluation

Refer to the readme file in `circuits/README.md`

A.4.1 Major Claims

(C1): Amplification, Eviction Set, and Prime + Store Attack *We can achieve a difference of 100ms between medians in native amplification (graph 8) We can achieve a difference of 1ms between medians in WASM amplification (graph 9) We can find eviction sets in WASM (graph 10). This is proven in experiment (E1)*

(C2): Trace Processing and Stitching *ElGamal Key Recovery using Prime+Store attack. This is proven by the experiment (E2) described in [Section 6.3 - 6.6] whose results are illustrated/reported in [Figure 11-12]*

(C3): Circuit Evaluation *Circuits achieve accuracy as described in the paper. This is proven by the experiment (E3) described in [Section 4] whose results are illustrated/reported in [Figure 3-6, and Table 1].*

A.4.2 Experiments

(E1): Amplification, Eviction Set, and Prime + Store Attack *[3 human-hours + 3 compute-hours + 500MB disk]: In this experiment*
How to: Follow the steps detailed in appendix A.4
Preparation: Install emscripten. As described in appendix A.3.1
Execution: Refer to the steps described in appendix A.4.
Results: Refer to claim (C1).

(E2): Trace Processing and Stitching *[1 human-hour + 1 compute-hour on (80 thread HPC) + 100MB disk]: In this experiment, ElGamal traces obtained from Prime+Store are transformed into square and multiply traces, and then key stitching is performed to combine partial traces into the complete ElGamal key.*
How to: Follow the steps detailed in appendix A.4
Preparation: To compile the trace processing and stitching programs, navigate to the "trace_processing_and_stitching" directory in the repository, and issue the make command.

```
make
```

Execution: Refer to the steps described in appendix A.4.
Results: You can follow the instructions outlined in section appendix A.4. Running the "sigproc" command will produce a set of square and multiply traces. It's important to note that these traces represent only portions of the complete ElGamal key, and must be combined or "stitched" together to retrieve the full key. This process is accomplished using the "stitch_parallel_simd_any_pos" and "stitch_parallel_simd" commands, which will generate the complete ElGamal key as the output in the form of square and multiply representation. Use the "sm_to_exponent" program to convert this into binary format.

(E3): Circuit Evaluation *[30 human-minutes + 3 compute-days + 100MB disk]: This experiment evaluates the accuracy of our circuits. Results are stored in each subdirectory of the experiments under the name "RESULT" which include the raw samples, their mean, and median.*
How to: Follow the steps detailed in appendix A.4
Preparation: 1) Navigate to the "circuits" directory in the cloned repository. 2) Execute `./compile.sh` to compile program.
Execution: Refer to the steps described in appendix A.4.
Results: Refer to the steps described in appendix A.4. Results of each circuit experiment is stored in a directory named "RESULT". This directory contains the raw samples, mean, and median. Use the "final.csv" to recreate the histograms presented in the paper. The "FINAL_MEAN" and "FINAL_MEDIAN" files carry the mean and median values respectively for each circuit, and are the values used in the paper. Follow the steps in appendix A.4 to generate figures identical to the ones found in the paper from collected data.

A.5 Notes on Reusability

All of our experiments are done on specific hardware, and reproducing on others may need additional tuning.

A.6 Version

Based on the LaTeX template for Artifact Evaluation V20220926. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at `https://secartifacts.github.io/usenixsec2023/`.