ProSpeCT: Provably Secure Speculation for the Constant-Time Policy

Lesly-Ann Daniel, Marton Bognar, and Job Noorman, *imec-DistriNet, KU Leuven;* Sébastien Bardin, *CEA, LIST, Université Paris Saclay;* Tamara Rezk, *INRIA, Université Côte d'Azur, Sophia Antipolis;* Frank Piessens, *imec-DistriNet, KU Leuven*

https://www.usenix.org/conference/usenixsecurity23/presentation/daniel

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 32nd USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 32nd USENIX Security Symposium.

August 9–11, 2023 • Anaheim, CA, USA

978-1-939133-37-3

Open access to the Artifact Appendices to the Proceedings of the 32nd USENIX Security Symposium is sponsored by USENIX.

USENIX'23 Artifact Appendix: PROSPECT: Provably Secure Speculation for the Constant-Time Policy

Lesly-Ann Daniel, Marton Bognar, Job Noorman, Sébastien Bardin, Tamara Rezk, Frank Piessens

A Artifact Appendix

A.1 Abstract

The artifact contains the source code of the base Proteus processor extended with PROSPECT, alongside the benchmarks and security tests from our paper. All materials (except for the tool required for hardware cost measurements) are bundled into a Docker container and distributed on GitHub.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

None, our artifact is contained in a Docker container, it does not perform any attacks against the host system and it does not use user data.

A.2.2 How to access

The artifact is available on GitHub at the following URL: https://github.com/proteus-core/prospect/tree/usenix_artifact.

A.2.3 Hardware dependencies

None.

A.2.4 Software dependencies

Our artifact uses the following two tools, which are available for both Windows and Linux.

- Docker and 7 GB of disk space for the container (https://docs.docker.com/engine/install/).
- Xilinx Vivado 2022.2 Standard Edition, requiring approximately 55 GB of disk space (https://www.xilinx.com/products/design-tools/vivado/vivado-ml.html).

A.2.5 Benchmarks

Our evaluation uses modified benchmarks from the SpectreGuard paper, which are included in our artifact.

A.3 Set-up

A.3.1 Installation

1. Install the two dependencies (Docker and Vivado). Our repository contains detailed instructions on setting up Vivado to minimize the required disk space.
2. Clone our GitHub repository or download the Dockerfile from the root directory (https://github.com/proteus-core/prospect/tree/usenix_artifact).
3. Build the Docker container by following the instructions in the `README.md` of the repository (building takes approximately 2 hours on a mid-range desktop).

A.3.2 Basic Test

The security evaluation can be run from the Docker container using the following commands:

```
// first, launch the container
$ docker run -i -t prospect

// inside the container, run the tests
# cd /prospect/tests/spectre-tests/
# ./eval.py /proteus-base/sim/build/base \
    /prospect/sim/build/prospect
TEST secret-before-branch
SECURE VARIANT:   Secret did not leak!
INSECURE VARIANT: Secret leaked!
[...]
```

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): PROSPECT prevents the leakage of secrets from well-annotated programs via Spectre attacks. This is shown by experiment (E1) described in Section 6.2, which executes programs vulnerable to Spectre on the baseline and the extended secure implementation.

(C2): PROSPECT incurs no overhead on precisely annotated constant-time code. This is shown by experiment (E2), described in Section 6.2 (Runtime overhead) and Table 1.

(C3): ProSpeCT only incurs a small overhead in terms of hardware cost. This is shown by experiment (E3), described in Section 6.2 (Hardware cost).

A.4.2 Experiments

(E1): [Security tests, 5 human-minutes]:
How to: The experiment is performed in the container by launching a script (identical to the basic test A.3.2).
Preparation: Launch the container with `docker run -i -t prospect` and navigate to the experiment with `cd /prospect/tests/spectre-tests`.
Execution: Run the following command:
```
./eval.py /proteus-base/sim/build/base \
/prospect/sim/build/prospect
```
This will run and evaluate the experiments with both the baseline implementation (first argument) and the ProSpeCT-extended version (second argument).
Results: The results are displayed as text. The security evaluation should fail with the baseline implementation and succeed with the extension, validating claim (C1).

(E2): [Runtime overhead, 5 human-minutes + 9 compute-hours]:
How to: The experiment is performed in the container by launching a script.
Preparation: Launch the container with `docker run -i -t prospect` and navigate to the experiment with `cd /prospect/tests/synthetic-benchmark`.
Execution: Run the following command:
```
./eval.py \
/proteus-base/sim/build/base_nodump \
/prospect/sim/build/prospect_nodump
```
This will run and evaluate the experiments with both the baseline implementation (first argument) and the ProSpeCT-extended version (second argument), using the variants compiled with no waveform dumping to save disk space.
Results: The results are displayed as text. The generated table should reflect Table 1 from the paper, validating claim (C2).

(E3): [Hardware cost, 1 human-hour + 2 compute-hours]:
How to: The experiment is performed in Vivado, using generated Verilog files from the Docker container.
Preparation: Follow the instructions under the heading *Hardware overhead* in `README.md` to obtain the Verilog files used for the synthesis and to set up the Vivado project (*Creating the Vivado project*).
Execution: Follow the instructions under the heading *Running the Vivado evaluation* in `README.md` to (iteratively) obtain the hardware costs of both the baseline and the ProSpeCT-extended hardware design.
Results: The results of the synthesis should be interpreted according to the description under the heading *Interpreting the results* in the `README.md` and compared to the reported numbers in the paper under the heading *Hardware cost* (Section 6.2).

A.5 Notes on Reusability

Using the `newlib` board support package included in this repository and building on the scripts used for our benchmarks, it is possible to run other benchmarks on Proteus and ProSpeCT, making additional benchmarking and security tests possible. The source code of ProSpeCT can also be modified to investigate tradeoffs or to extend the offered security guarantees.

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/.