A Artifact Appendix

A.1 Abstract

This artifact contains our reverse-engineering (RE) tools, the covert and side-channel attacks, and the analytical model described in the paper. The RE tools can be used to explore the mesh interconnect by testing various sender/receiver placements. After RE, the covert and side-channel proof-of-concepts can be run. All of our tools were tested on a bare-metal machine with Ubuntu 18.04 and a 24-core Intel Xeon Gold 5220R (Cascade Lake) processor. Other necessary software dependencies are outlined for each component. The artifacts should produce the graphs shown in the paper as well as reproduce the attack performance.

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

- **Compilation:** GCC 7.5.0
- **Run-time environment:** These artifacts have been tested on Ubuntu 18.04. The main software dependencies are GCC 7.5.0 and Python ($\geq$ 3.6). Root access is needed to facilitate the reverse-engineering process.
- **Hardware:** Intel Xeon Gold 5220R (Cascade Lake)
- **Run-time state:** The experiments are sensitive to the state of the on-chip network. This means that cache activity (which creates network traffic) can add noise to the experiments.
- **Security, privacy, and ethical concerns:** The experiments in this artifact do not attempt to maliciously exploit any systems.
- **How much disk space required (approximately)?:** 4 GB
- **How much time is needed to prepare workflow (approximately)?:** 1 hour
- **How much time is needed to complete experiments (approximately)?:** 48 hours
- **Publicly available (explicitly provide evolving version reference)?:** `https://github.com/CSAIL-Arch-Sec/dont-mesh-around`
- **Code licenses (if publicly available)?:** MIT License
- **Archived (explicitly provide DOI or stable reference)?:** `https://github.com/CSAIL-Arch-Sec/dont-mesh-around/releases/tag/usenix2022`

A.3 Description

A.3.1 How to access

The artifact can be downloaded by cloning our Github repository.

A.3.2 Hardware dependencies

These artifacts target the Intel Xeon Gold 5220R (Cascade Lake) processor. In particular, the processor must be an Intel Skylake SP or Cascade Lake processor which use the mesh interconnect. Evaluating the artifact should require less than 4 GB of disk space.

A.3.3 Software dependencies

These artifacts were tested on Ubuntu 18.04. The software requires Python ($\geq$ 3.6) and GCC 7.5.0.

A.3.4 Data sets

N/A

A.3.5 Models

N/A

A.3.6 Security, privacy, and ethical concerns

None of the experiments in the artifact attempt to maliciously exploit any systems. However, they require access to a server-class processor that is normally shared by many users and will change some system configurations. Be sure to check the original values of the configurations modified in the setup script and restore them afterwards.

A.4 Installation

All installation instructions are included in the artifact. Users should be comfortable using APT and Python/Pip to install dependencies. Familiarity with Git is needed to clone the repository. Additionally, some experiments are long-running and should be run inside a `tmux` session. More specialized experiments come with guidance on how to set them up.

A.5 Experiment workflow

Each experiment is contained in its own directory and is built with its own Makefile. A README within each directory contains detailed information on how to build and run the experiment. It is recommended that the experiments be run in the order presented.

A.6 Evaluation and expected results

Our paper makes the following claims:

1. We can reverse-engineer previously-unknown details about Intel's mesh interconnect.
2. The reverse-engineering results can inform the construction of covert and side-channel attacks.
3. The reverse-engineering results allow for the construction of an analytical model that accurately predicts interconnect-based leakage.
4. The analytical model offer insights into mitigating interconnect-based side channel attacks.

The key results of our paper are detailed below. Detailed instructions on how to reproduce each key result are included in the artifact.

A.6.1 NoC Reverse-Engineering

We reverse-engineered the lane-scheduling policy and priority arbitration policies that dictate how traffic flows on the interconnect. These policies are not specified by Intel and have not been publicly reverse-engineered prior to our paper but are critical to understanding the precise conditions necessary to generate contention. We verify these results by reproducing the two case studies shown in the paper.

A.6.2 Covert Channel

In this section, we demonstrate a working covert channel using only contention on the interconnect that can achieve a capacity comparable to that of previous interconnect-based covert channels (1.5 Mbps ± 0.3). Our artifact can reproduce the latency trace and capacity plot shown in the paper.

A.6.3 Side Channel

Secret keys can be extracted from vulnerable ECDSA and RSA implementations via the interconnect channel. Single-bits can be classified with an accuracy of at least 69% and 71% respectively and full keys can be recovered with majority voting.

A.6.4 Analytical Model

The reverse-engineering results can be used to construct an analytical model of network contention that accurately predicts observed results. The analytical model can be used to create non-invasive mitigations that reduce the effectiveness of our side-channel. These artifacts should be able to reproduce Figures 12, 13, and 14 in the paper.

A.7 Experiment customization

Running the experiments in the artifact requires first reverse engineering the layout of the tiles on the die, including where the partially and fully disabled tiles are. Because the location of these disabled tiles may differ between different units of the same processor, this must be done before attempting to run the experiments on a different machine. Guidelines for how to do this reverse-engineering are described in Section 3 and Appendix B of the paper.

A.8 Notes

N/A

A.9 Version

Based on the LaTeX template for Artifact Evaluation V20220119.