USENIX Security '25 Artifact Appendix: Not so Refreshing: Attacking GPUs using RFM Rowhammer Mitigation

Ravan Nazaraliyev[*1], Yicheng Zhang[1], Sankha Baran Dutta[2], Andres Marquez[3], Kevin Barker[3], Nael Abu-Ghazaleh[†1]

[1]*University of California, Riverside*
[2]*Brookhaven National Laboratory*
[3]*Pacific Northwest National Laboratory*

A Artifact Appendix

This Artifact Appendix provides the necessary details to enable evaluations of our artifacts.

A.1 Abstract

Ou artifact is composed of one main part, code scripts for GPU covert, side channel and slow-down attacks, and an additional part for machine learning classification algorithms along with pre-collected data.

A.2 Description & Requirements

We have provided code scripts to recreate covert channel and slow-down attacks locally. We also provide the side channel code that we used to collect data along with pre-collected data and classification models for fingerprinting attacks.

A.2.1 Security, privacy, and ethical concerns

Our codes do not cause any issues regarding security, privacy, and ethics. The provided artifact should be run locally on a machine, and the codes show proof-of-concept attacks. The codes, as expected, do not cause any harm to the machine nor steal private information.

A.2.2 How to access

We have provided our artifacts in Zenodo repository. The main part is located in notsorefreshing_artifact repository. When you enter the repository, you will see the folder **`notsorefreshing_artifact`** folder which is the main folder that contains code scripts. The other folders contain ML classification models for collected data. We provide a README file for main code scripts inside `notsorefreshing_artifact` folder. We provide an additional README file (**`ReadMe_For_5_side_channel_attacks.md`**).

A.2.3 Hardware dependencies

Our code scripts require a custom CPU (our machine is Intel Xeon E5-1620). The provided experiments require NVIDIA GeForce RTX 4060 and Jetson Orin AGX LPDDR5. If you do not intend to collect data on Jetson or do not have access to one, you can use the already provided dataset.

A.2.4 Software dependencies

The evaluation machine should have NVIDIA Driver and CUDA toolkit installed. We have provided the driver as part of artifact. Please refer to CUDA toolkit download for CUDA toolkit installation. To run Blender object fingerprinting, the machine needs to have the Blender package, and you can install Blender on Linux machine using *snap install blender –classic*. Please refer to the README file for further information.

A.2.5 Benchmarks

For all side channel experiments, we have provided necessary benchmarks as part of the artifact. We only do not provide Blender benchmark for slow-down attack as it is easy to get it from Blender benchmark. To run this benchmark, you have to have Blender package (refer to Sec A.2.4) and git clone Phoronix Test Suite.

A.3 Set-up

We do not require a specific setup. The attack codes (CUDA) need to be run with GUI mode off. You can disable GUI using `sudo service lightdm stop`. If your machine does not have `lightdm`, you can easily install it using `sudo apt install lightdm`. Please refer to `README` for further details.

[*]rnaza005@ucr.edu
[†]naelag@ucr.edu

A.3.1 Installation

Please refer to Sections A.2.2 and A.2.4 for related information. Please refer to README for further details.

A.3.2 Basic Test

The covert channel attack consists of Python scripts, as described in the README file. The test outputs the achieved bandwidth and error rate. For the slow-down attack, run the Blender benchmark with and without the attack active. When the attacker is active, the benchmark takes longer, and the ratio of execution times gives the slow-down factor. The side-channel scripts begin collecting data when executed. If pre-collected data suffices for demonstrating functionality or reproducibility, data collection is not required, as the necessary traces are already included.

A.4 Evaluation workflow

We provide the necessary details for running the experiments in the README files. Briefly, for the covert channel, run `RFM_Covertchannel_with_physidech.py` to send a random message, then run `covertchannelevaluation.py` to measure the error rate. This covert channel works on GeForce RTX 4060. Start MPS before running this attack (`nvidia-cuda-mps-control -d`). This artifact corresponds to Section 4.

Slowdown attack. Run `./slow_down` to start the slow-down code, then launch the benchmark. We used the Blender benchmark; see Section A.4. Start MPS before this attack (`nvidia-cuda-mps-control -d`). This artifact corresponds to Section 7.

Side-channel codes:

GeForce RTX 4060 (with MPS). For Blender 3D object fingerprinting, run `sidechannelautomatic.py`. It launches the memory profiler, renders Blender objects, collects samples, runs `memorygram.py`, and deletes CSV files to save space. Requirements: 1) install Blender (Section A.2.4), 2) start MPS (`nvidia-cuda-mps-control -d`). This artifact corresponds to Section 5.

Jetson Orin AGX 64GB (without MPS). Go to the `AttacksOnJetson` folder. The three attacks are in separate folders. For SPEC fingerprinting, run `./specfingerprint.sh`. For Web and Video attacks, run `./timeseriesdatacollection_web` and `./timeseriesdatacollection_video`, respectively. Data is saved in CSV files. This artifact corresponds to Section 6.

A.4.1 Major Claims

Our artifacts provide reproducible and functional proof-of-concept code implementations for the attacks presented in the paper.

(C1): *We demonstrate a covert channel across two GPU processes based on RFM Rowhammer mitigation. The channel achieves high bandwidth with a low error rate. Results are shown in Table 1.*

(C2): *We demonstrate side-channel fingerprinting attacks on GDDR and LPDDR systems. On GeForce RTX GPUs, we achieve over 90% precision in identifying the victim application (Attack1) or Blender character (Attack2) rendered on a shared GPU (Tables 2 and 3). On Jetson Orin AGX (LPDDR), we show that a spy process can capture RFM fingerprints of a CPU process. We demonstrate application, web, and video fingerprinting attacks (Attack3–5), each achieving near or above 90% accuracy (Table 4).*

(C3): *We show that an attacker can significantly slow down another application by exploiting RFM-based Rowhammer mitigation, with an average slowdown of 4.8×.*

A.4.2 Experiments

We have already provided the necessary details on how to run our code to reproduce results. Below, we provide explicit instructions corresponding to claims in the previous section.

(E1): *[Covert channel] [a couple of minutes of human time]:*

How to: Start MPS using the `nvidia-cuda-mps-control -d` command. In the main artifact folder, run `RFM_Covertchannel_with_physidech.py` to initiate covert communication. After completion, it will output status messages and the channel bandwidth. Then, run `covertchannelevaluation.py` to obtain the error rate.

Preparation: Disable GUI using `sudo service lightdm stop`. If `lightdm` is not installed, install it via `sudo apt install lightdm`.

Execution: Run the mentioned Python scripts to obtain results.

Results: You should observe bandwidth and error rate values matching or close to those reported in the paper.

(E2): *[Side channel] [>1 day for full data collection]:*

How to: We provide scripts for both data collection and classification. Full data collection takes more than a day per script (4 total). For convenience, we suggest running ML classifiers on pre-collected data, which was gathered using the same scripts.

Preparation: For data collection, disable GUI using `sudo service lightdm stop`. No preparation is needed to run the ML classifiers.

Execution: Follow the README instructions to run data collection and analysis scripts.

Results: Data collection alone does not yield results. Run ML scripts to obtain F1 score, precision, and recall.

(E3): *[Slow-down attack] [2–3 hours]:*

How to: Run the Blender benchmark with and without

the attacker to observe slowdown.

Preparation: Disable GUI via `sudo service lightdm stop`. If `lightdm` is not installed, use `sudo apt install lightdm`.

Execution: The Blender benchmark is available at OpenBenchmarking. To run it, install Blender (see Section A.2.4) and clone Phoronix Test Suite. To launch the attacker, go to the main artifact folder and run `./slow-down`.

Results: You should observe increased benchmark runtime when run concurrently with the attacker.

A.5 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2025/.