USENIX Security '25 Artifact Appendix: GPUHammer: Rowhammer Attacks on GPU Memories are Practical

Chris S. Lin[†]
University of Toronto
shaopenglin@cs.toronto.edu

Joyce Qu[†]
University of Toronto
joyce.qu@mail.utoronto.ca

Gururaj Saileshwar
University of Toronto
gururaj@cs.toronto.edu

A Artifact Appendix

A.1 Abstract

This artifact supports the paper "GPUHammer: Rowhammer Attacks on GPU Memories are Practical", which presents the first successful Rowhammer attack on GPUs. The paper introduces GPUHammer, a framework that reverse-engineers GDDR6 DRAM row mappings, employs GPU-specific memory access patterns to intensify hammering, and bypasses existing mitigations. Using this approach, we demonstrate bit flips on an NVIDIA A6000 GPU and show how they can be exploited to tamper with machine learning models, leading to severe accuracy degradation. The artifact includes the GPUHammer implementation and evaluation scripts used to replicate (1) Rowhammer campaigns on A6000 GPU (Table-1), (2) Characterization of Rowhammer bit-flips in A6000's GDDR6 (Figure 11, Figure 12, Table 3), and (3) Exploits on ML models (Figure 13, Table 4).

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

This artifact triggers Rowhammer faults on GPU memory and may induce bit-flips. Although these are contained within GPU DRAM and the scope of evaluation is limited, evaluators should not run this artifact on production systems or GPUs used for other sensitive tasks.

Ensure that ECC is disabled and GPU settings are configured as described. Use at your own risk.

A.2.2 How to access

The artifact is hosted on Zenodo and will become public once the embargo on this work lifts on August 12, 2025: Zenodo link: https://doi.org/10.5281/zenodo.15612689

A.2.3 Hardware dependencies

NVIDIA A6000 GPU with GDDR6 memory (48 GB)

[†]Equal contribution.

A.2.4 Software dependencies

- **Rowhammer Bit Flip Characterization**
  - Ubuntu 20.04+
  - NVIDIA CUDA Toolkit 12.3
  - NVIDIA CUDA Driver $\geq$ 545.23.08
  - g++ ($\geq$11.4.90, C++17)
  - CMake $\geq$ 3.26.4
  - Python $\geq$ 3.10
- **Exploit**
  - Anaconda 24.9.2 (Already installed on our machine)
  - RAPIDS RMM (Installed by artifact setup scripts)

A.2.5 Benchmarks

ImageNet 2012 Validation Set: **ILSVRC2012_img_val.tar**, required for the exploit on ML models, can be downloaded from https://www.image-net.org/download.php.

A.3 Set-up

A.3.1 Installation

Download Artifact and Datasets:

1. Please download the artifact from the Zenodo link provided in Section A.2.2. The downloaded file should be named `gpuhammer-main.zip`. Please unzip this folder.
2. Download the ImageNet validation set from appendix A.2.5 and move it inside the unzipped folder.

Setup environment:

```
cd gpuhammer-main
export HAMMER_ROOT=`pwd`

# Install Dependencies
bash ./run_setup.sh

# Enable rmm_dev environment and build GPUHammer
conda init
```

```
source activate base
conda activate rmm_dev
cmake -S $HAMMER_ROOT/src \
      -B $HAMMER_ROOT/src/out/build
cd $HAMMER_ROOT/src/out/build
make
cd $HAMMER_ROOT
```

Listing 1: Setup instructions

A.3.2 Basic Test

Generate a small Conflict-Set with the following command:

```
cd gpuhammer-main
python3 ./util/run_timing_task.py conf_set \
--threshold 27 \
--step 256 \
--it 15
```

Listing 2: Commands to generate a Conflict-Set

Expected output: The script should generate a file `CONF_SET.txt` that has 80 lines. The first 8 numbers expected in the file are: `852224, 852736, 868352, 868864, 895232, 895744, 911360, 911872`.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): GPUHammer induces Rowhammer bit-flips on NVIDIA A6000 GPUs. We validate this with experiment E1 in Section 7.1, whose results are reported in Table 1.

(C2): The bit-flips on A6000 GPUs have the following characteristics: (1) Rowhammer Threshold (T_{RH}) of $\approx 12.3K$, TRR sampler size of 16, and bidirectional bit-flips. This is demonstrated by experiments E1 and E2 described in Sections 7.3-5, whose results are reported in Figure 11, Figure 12, and Table 3.

(C3): In a few memory massaging attempts, bit flips can be induced in ML model weights to significantly degrade model accuracy. This is demonstrated in experiment E3 described in Section 8.3, whose results are reported in Figure 13 and Table 4.

A.4.2 Experiments

The detailed steps to run the experiments are provided in the README.md file. In the following subsections, we provide an overview of the steps to run the experiments. The following commands automatically run all the experiments (E1, E2, E3) which will take a total of 3.5 days:

```
cd gpuhammer-main
bash ./run_artifact.sh
```

Listing 3: Commands to run all experiments

Below, we list the steps run by the script in listing 3.

(E0): ConflictRow-Set Generation (1 day): The following generates the Row-Set and Conflict-Set for the banks that will be hammered in subsequent experiments. If these have already been run, then you can skip this step.

```
cd gpuhammer-main
bash ./run_row_sets.sh
```

Listing 4: Commands to generate Conflict/Row-Sets

(E1) Systematic Hammering Campaigns: (1 day) This experiment performs 24-sided hammering patterns on the entire memory of the 4 different DRAM Banks (A, B, C, D) listed in the paper. On executing this experiment, the evaluators should see some (if not all) of the 8 flips in Table 1 (C1), and be able to observe the characteristics of those bit-flips in Table 3 (C2). One may see only a subset of the flips as due to time constraints we limit the attempts to trigger a bit flip on any given row.

Execution: To execute the 24-sided systematic hammering campaign on all four DRAM banks, run:

```
cd gpuhammer-main
bash run_t1_t3.sh
```

Listing 5: Execute systematic hammering campaign

Results: The results of the campaign in the folder `results/campaign/`, generate the last column of Table 1, and Table 3. The last column of Table 1 in `t1.txt`, shows the number of bit-flips in each of the 4 DRAM banks. Table 3 in `t3.txt` shows the information about each bit-flip, such as flip-direction, location, etc. We additionally provide the sample output of the Rowhammer campaign in `results/sample`, as the tables generated by the evaluator may not be complete due to the random nature of Rowhammer bit-flips.

(E2) Bit-flip Characterization (5 hours): We run two experiments that characterizes the Rowhammer Threshold (T_{RH}) and the TRR Sampler Size. On running these experiments, the key results to be reproduced include: (1) T_{RH} of the 8 bit-flips to be $12K$-$16K$ as shown in Figure 11 (C2), and that (2) bit-flips are only observed for 17-sided patterns and beyond as per Figure 12 (C2).

Execution: To execute these experiments, run:

```
cd gpuhammer-main
bash run_fig11.sh
bash run_fig12.sh
```

Listing 6: Execute bit-flip characterization experiments

Results: The results of the T_{RH} and TRR sampler size characterizations are in folders `results/fig11` and `results/fig12`, respectively, and the scripts recreate Figure 11 and Figure 12 stored as `fig11.pdf` and `fig12.pdf`.

(E3) ML Model Exploit (1.5 days): We craft exploits to degrade accuracy of ML models with the bit-flips. The experiments massage the GPU memory to force placement of model weights in victim locations followed by hammering to inject bit-flips in these victim locations. We perform this on 5 models with 50 massaging attempts each. The results to be reproduced include: (1) worst case model accuracy and Relative Accuracy Drop (RAD) with each bit-flip for all the models similar to Table 4 (C3), (2) Attack attempt vs RAD for bit-flip D1 (Figure 13), showing 99% RAD in < 10 attempts (C3).

Execution: To perform the exploit, run:

```
cd gpuhammer-main
bash run_fig13_t4.sh
```

Listing 7: Execute ML model exploit

Results: The results of T_{RH} and sampler size characterization are in the folder `results/fig13_t4`. Table 4 and Figure 13 are automatically generated from the result and stored as `t4.txt` and `fig13.pdf`, respectively.

A.5 Notes on Reusability

The README provides more details including:

- Re-configuration of the code to work on other GPUs.
- Extending the hammering by modifying the hammering parameters (e.g., aggressor pattern length, tuning the delays for synchronization).

A.6 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/.