USENIX Security '25 Artifact Appendix: McSee: Evaluating Advanced Rowhammer Attacks and Defenses via Automated DRAM Traffic Analysis

Patrick Jattke, ETH Zurich
Michele Marazzi, ETH Zurich
Flavien Solt, UC Berkeley
Max Wipfli, ETH Zurich
Stefan Gloor, ETH Zurich
Kaveh Razavi, ETH Zurich

A Artifact Appendix

A.1 Abstract

McSee is an open-source platform for capturing and analyzing DRAM bus traffic. It is based on a high-frequency oscilloscope with a custom-designed interposer and an optimized trace data processing pipeline, including a DDR4/5 decoder.

The artifacts of our work include the McSee platform, including the design of the custom-built DDR5 UDIMM interposer and the oscilloscope decoding and analysis pipeline. Further, we provide the code of our experiments and the captured oscilloscope traces for validating our claims.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

Our work does not involve any end-to-end Rowhammer attacks, neither does McSee capture any data signals from the DRAM bus. Therefore, our artifacts do not raise any security, privacy, or ethical concerns.

A.2.2 How to access

The artifacts and collected data is available on Zenodo at `https://doi.org/10.5281/zenodo.15610915`. To facilitate the access, we also publish the artifacts (code only) on GitHub at `https://github.com/comsec-group/mcsee`.

A.2.3 Hardware dependencies

Reproducing our results requires systems with specific CPUs, DDR5 UDIMMs requiring RFM, and the hardware of the McSee platform.

Systems. We used following CPUs for our experiments.

- **DDR5 experiments**:
  - Intel Alder Lake (i7-12700K), ucode: `0x3a`
  - Intel Raptor Lake (i7-13700K), ucode: `0x129`
  - AMD Zen 4 (Ryzen 7 7700X), ucode: `0xa601203`
- **DDR4 experiments**:
  - Intel Coffee Lake (i7-8700K), ucode: `0xf8`

DDR5 UDIMMs. Validating our results on real hardware requires DDR5 UDIMMs where the RFM required bit is set and valid RFM values are provided in the SPD data. This data can be obtained from the SPD EEPROM of the UDIMM by using the SPD reader tool provided in the artifacts.

Reproducing the DDR5 bit flips that we found on one of the devices (S_4, Samsung), requires the same DDR5 UDIMM as used in Zenhammer: a 16 GB Samsung UDIMM, model `M323R2GA3BB0-CQKOD` with 4800 MHz.

McSee platform hardware. The McSee platform is based on the following hardware components (see §4.1 for details):

- Oscilloscope (TELEDYNE SDA 806Zi-B)
- Digitizer (TELEDYNE HDA125-18-LBUS)
- Differential probe (TELEDYNE DH08-PB2)
- Analog probe (TELEDYNE PP021)
- Solder-in leads (TELEDYNE HDA-DLS-18QL)
- DDR5 UDIMM interposer (custom design, see artifacts)
- 5 GbE USB NIC (Startech US5GA30)

We provide the oscilloscope captures to reproduce our results such that no access to McSee and the exact same hardware is required.

A.2.4 Software dependencies

We require a Linux-based operating system to run our software. Our experiments were conducted on Ubuntu 20.04.6/22.04.5/24.04.1 LTS and GNU/Linux kernel versions 6.1.118/6.8.0/6.9.0. However, our experiments and tools do not rely on any particular OS kernel version.

A.2.5 Benchmarks

None.

A.3 Set-up

A.3.1 Installation

We refer to the README file in the repository for installation instructions.

A.3.2 Basic Test

The README of the respective component of our McSee platform contains information on how to check if it has been installed correctly.

A.4 Evaluation workflow

A.4.1 Major Claims

Our work makes the following major claims:

(C1): The **McSee data processing pipeline** is open source and allows to efficiently capture and decode DDR4/5 bus traffic (see §4.2). We describe where to find the different software components of McSee in the README of our artifacts repository.

(C2): The **PCB design** of our custom-built DDR5 UDIMM interposer as part of the McSee platform is open source (see §4.1). The design can be found in the `ddr5-udimm-interposer-pcb` directory of the artifacts repository.

(C3): The **DDR5 SPD decoder** is open source and allows to decode the RFM values of DDR5 UDIMMs (see §6.1). The decoder can be found in the `spd-decoder` directory of the artifacts repository. We describe in experiment (E1) how we used the decoder to extract the relevant values.

(C4): We show that **Sledgehammer's** activation throughput declines when hammering more than six banks in parallel (see §5.1). This is proven by experiment (E2) whose result is illustrated in Figure 7.

(C5): We show that **Sledgehammer** does not achieve less reordering of aggressor accesses when hammering more banks in parallel (see §5.1). This is proven by experiment (E3) whose result is illustrated in Figure 8.

(C6): We show that **RowPress'** average row-open time on a real system is in the range where the reduction of the minimum activation count (AC_{min}) to trigger a bit flip is small (see §5.2). This is proven by experiment (E4) whose result is illustrated in Figure 9.

(C7): **Systematic bit flipping** allows to recover the DRAM addressing functions of our Zen 4 CPUs (see §6.2). This is proven by experiment (E5) whose result is presented in Table 3.

(C8): Intel Raptor Lake CPUs do not issue any **RFM** commands, but they employ a memory controller-based mitigation **`pTRR`** that refreshes aggressor-adjacent rows (see §6.3). This is proven by experiment (E6) whose result is illustrated in Figure 10.

(C9): Intel **pTRR** targets victim rows with a probability of 0.091% (see §6.4). This is proven by experiment (E7) whose result is illustrated in Figure 11.

(C10): Given the pTRR probability, we calculate the **pTRR bypass time**: devices protected with pTRR with a Rowhammer threshold of 13200, 16700, and 18800 activations can be bypassed with roughly 50% attack success probability in less than an hour, one day, and one week, respectively. This is proven by experiment (E8) whose result is illustrated in Figure 12.

A.4.2 Experiments

As we require multiple different systems attached to the McSee platform to reproduce our results, we provide the (unprocessed) oscilloscope traces of the following experiments. We report effort as a triple "[HM/CH/DU]" of human-minutes (HM), compute-hours (CH), and disk usage in gigabytes (DU).

(E1): **RFM values** [1:30 h / 0:20 h / 1 MiB]. We use our DDR5 SPD decoder to extract the RFM values from the SPD data of the 29 DDR5 UDIMMs that we use in our experiments.

Preparation: Follow the instructions in the README file in `experiments/e1-rfm-values`.

Execution: Extract the SPD data of the DIMMs, then parse the `arfm` object of the generated JSON files. We provide the collected SPD data in the experiment's `data/` directory.

Results: As shown in Tbl. 7, 19 of 30 devices have valid RFM values (i.e., `raaimt` and `raammt` are not set to `RFU`). One of the 19 devices has the RFM required bit (`rfm_req`) set.

(E2): **Sledgehammer's activation throughput** [2:20 h / 1h / 40 GiB]. Using McSee, we measure the per-bank and total activation throughput of Sledgehammer.

Preparation: Follow the instructions in the README file in `experiments/e2-sledgehammer`.

Execution: Run our modified Sledgehammer fork on the Coffee Lake system while taking three captures of each 1 ms with the McSee platform. Repeat this for 1 to 16 banks.

Results: The activation rate per bank strongly declines when hammering more than six banks in parallel, as shown in Fig. 7.

(E3): Sledgehammer's access reordering [0:05 h / 0:01 h / 40 GiB]. We analyze the distance (in activations) between consecutive accesses to the same aggressor in Sledgehammer.

Preparation: Follow the instructions in the README file in `experiments/e2-sledgehammer`.

Execution: No execution is required, we use the data gathered in (E2) and use a script to analyze the traces.

Results: Increasing the number of in parallel hammered banks, increases the frequency of reordering aggressor accesses, as shown in Fig. 8.

(E4): RowPress' row-open time [1h / 0:40 h / 20 GiB]. We run the real system demonstration of RowPress on a Coffee Lake system to measure the average row-open time using McSee.

Preparation: Follow the instructions in the README file in `experiments/e4-rowpress`.

Execution: Run our modified RowPress fork on the Coffee Lake system while taking two captures of each 1 ms with the McSee platform. Repeat this for 1, 4, 8, 16, 32, 64, 80, and 128 column reads.

Results: An average t_{AggON} time between approx. 50 ns (1 cache block read) and almost 300 ns (128 cache block reads) as shown in Fig. 9.

(E5): Systematic bit flipping [0:20 h / 0:30 h / 30 GiB]. We run the systematic bit flipping experiment on our Intel Raptor Lake system to recover the DRAM addressing functions and labels.

Preparation: Follow the instructions in the README file in `experiments/e5-systematic-bit-flipping`.

Execution: Run the provided code implementing the systematic bit flipping experiment described in Algorithm 1. This code automatically triggers Run the code on the three different DRAM configurations listed in Tbl. 3.

Results: The three different sets of DRAM addressing functions as shown in Tbl. 3.

(E6): Existence of pTRR [0:20 h / 0:15 h / 100 MiB]. We verify that on Intel Raptor Lake CPUs, the memory controller infrequently issues an activate command to a row neighboring our hammered aggressor row.

Preparation: Follow the instructions in the README file in `experiments/e6-ptrr-existence`.

Execution: We use the same hammering workload with a double-sided aggressor pair as in the pTRR probability experiment (E7).

Results: The aggressor-adjacent rows are refreshed by pTRR, as shown in Fig. 10, although they are never accessed in the experiment.

(E7): pTRR probability [0:30 h / 8h / 150 GiB] We run a double-sided hammering workload on our Intel Raptor Lake system to determine the probability of pTRR to targeting the victim row.

Preparation: Follow the instructions in the README file in `experiments/e7-ptrr-probability`.

Execution: We hammer a double-sided aggressor pair while capturing traces with the McSee platform.

Results: A bar plot fitting a binomial probability distribution with $p = 0.00091$ as shown in Fig. 11.

(E8): pTRR attack success rate [0:05 h / 0:05 h / 10 MiB] We use the determined pTRR probability to calculate the success rate of bypassing pTRR over time.

Execution: Run the provided MATLAB script `calculate_prob.mlx` either locally or using the MATLAB cloud service.

Results: A plot as in Fig. 12 showing that devices protected with pTRR with a Rowhammer threshold of 13200, 16700, and 18800 activations can be bypassed with roughly 50% attack success probability in less than an hour, one day, and one week, respectively.

A.5 Notes on Reusability

The core of our McSee platform is the DDR4/5 DRAM command decoder. As this decoder works on the oscilloscope traces that have been beforehand transformed into the CSV format, it is platform-agnostic and can be used with any other hardware setup that is capable of producing such CSV traces.

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/.