SoftTRR: Protect Page Tables against Rowhammer Attacks using Software-only Target Row Refresh

Zhi Zhang*, Yueqiang Cheng*, Minghua Wang, Wei He, Wenhao Wang, Nepal Surya, Yansong Gao, Kang Li, Zhe Wang, Chenggang Wu
(*: co-first authors)
Outline

• Background

• Motivation

• Overview

• Evaluation

• Conclusion
Background
What is Rowhammer?
What is Rowhammer?

Rowhammer: Frequently accessing DRAM rows
DRAM bank

A bank has rows of cells
DRAM bank

A bank has rows of cells

A cell has a capacitor and an access-transistor
DRAM Refresh

★ capacitors of cells can lose charge over time

★ cells must be periodically refreshed

★ the refresh rate is typically 64 ms in DDR3 and DDR4
Rowhammer

Kim et al. (ISCA’14)

frequently opening rows $n+1$ & $n-1$ cause charge leakage (bit flips) in row $n$
Motivation
Rowhammer Attacks

- Rowhammer-induced page tables corruption is the most detrimental to system security and hard to mitigate (*CTA ASPLOS’19*)

- Mainstream rowhammer attacks target level-1 page table corruption
Limitations of the State-of-the-Art Works
Limitations of the State-of-the-Art Works

- **Practicality**
  - incurring modifications to kernel memory subsystem.
Limitations of the State-of-the-Art Works

- **Practicality**
  - incurring modifications to kernel memory subsystem

- **Effectiveness**
  - being ineffective against all existing rowhammer attacks targeting page tables (e.g., PThammer \textit{MICRO'20})
Explicit Rowhammer Attacks

- Require access to part of rows adjacent to L1PT rows for explicit hammering
Explicit Rowhammer Attacks

- Require access to part of rows adjacent to L1PT rows for explicit hammering

Implicit Rowhammer Attacks

- PThammer, the only instance
Overview
Goal: protect page tables from rowhammer attacks
Goal: protect page tables from rowhammer attacks

Design Principles:

- effective in protecting page tables from explicit and implicit attacks
Goal: protect page tables from rowhammer attacks

Design Principles:
- effective in protecting page tables from explicit and implicit attacks
- compatible with OS kernels
Goal: protect page tables from rowhammer attacks

Design Principles:
- effective in protecting page tables from explicit and implicit attacks
- compatible with OS kernels
- small performance overhead to a commodity system
Key Insights

DRAM-chip-based TRR (ChipTRR), widely deployed in DDR4 modules.

- high-level idea: ChipTRR counts rows’ activations and refreshes adjacent rows to suppress bit flips if the activation counts reach a pre-defined limit.
Key Insights

**DRAM-chip-based TRR (ChipTRR), widely deployed in DDR4 modules.**

- high-level idea: ChipTRR counts rows’ activations and refreshes adjacent rows to suppress bit flips if the activation counts reach a pre-defined limit.

- security limitation: ChipTRR only tracks a limited number of rows, which renders its rowhammer-free guarantee broken by TRRespass IEEE S&P’20.
Key Insights

DRAM-chip-based TRR (ChipTRR), widely deployed in DDR4 modules.

- high-level idea: ChipTRR counts rows’ activations and refreshes adjacent rows to suppress bit flips if the activation counts reach a pre-defined limit.
- security limitation: ChipTRR only tracks a limited number of rows, which renders its rowhammer-free guarantee broken by TRRespass IEEE S&P’20.

Software-only TRR (SoftTRR): protects page-table integrity by adopting the above idea while addresses the security limitation by leveraging MMU and OS kernel features.
Memory-access Mediation

- TLB (paging structures)
- MMU (address translation)
- Page Tables
- Unauthorized user access
- Page-fault Handler

.notify
.update
Memory-access Mediation

SoftTRR leverages page tables and page-fault handler to frequently trace memory accesses to any rows adjacent to rows hosting page-tables.
Overview
Overview

SoftTRR Module

Page Table Collector
Page-Table & Adjacent Page
Page & DRAM Information
Page-Table & Adjacent Page Tracer
Page-Table Rows
Charge-Leak Counters
Row Refresher

Kernel
DRAM Bank
Non Page-Table
Flip

refresh
Page Table
Page Table Collector

- In our implementation, SoftTRR focuses on protecting level-1 page tables (L1PTs) that are targeted by both explicit and implicit rowhammer attacks.
Page Table Collector

- In our implementation, SoftTRR focuses on protecting level-1 page tables (L1PTs) that are targeted by both explicit and implicit rowhammer attacks.
- Page table collector asks task_struct and hooks L1PT alloc and free functions for page collection
  - L1PT pages
  - DRAM-adjacent pages
  - their DRAM row locations
Page Table Collector

- In our implementation, SoftTRR focuses on protecting level-1 page tables (L1PTs) that are targeted by both explicit and implicit rowhammer attacks.
- Page table collector asks task_struct and hooks L1PT alloc and free functions for page collection
  - L1PT pages
  - DRAM-adjacent pages
  - their DRAM row locations
- DRAM-adjacent page
Page Table Collector

- In our implementation, SoftTRR focuses on protecting level-1 page tables (L1PTs) that are targeted by both explicit and implicit rowhammer attacks.
- Page table collector asks `task_struct` and hooks L1PT alloc and free functions for page collection
  - L1PT pages
  - DRAM-adjacent pages
  - their DRAM row locations
- DRAM-adjacent page: up to 6-row from a row hosting L1PTs (based on Kim et al. ISCA’20)

![Diagram of page table collection](image)
Page Table Collector

- In our implementation, SoftTRR focuses on protecting level-1 page tables (L1PTs) that are targeted by both explicit and implicit rowhammer attacks.
- Page table collector asks task_struct and hooks L1PT alloc and free functions for page collection
  - L1PT pages
  - DRAM-adjacent pages
  - their DRAM row locations
- DRAM-adjacent page: up to 6-row from a row hosting L1PTs (based on Kim et al. ISCA’20)
- An attacker can explicitly or implicitly hammer an adjacent page
- Page table collector maintains three red-black trees for the collected information
  - pt_rbtree
  - adj_rbtree
  - pt_row_rbtree
Overview

[Diagram of SoftTRR Module]

- Page Table Collector
- Adjacent Page Tracer
- Row Refresher

Error and DRAM information:
- Page-Table & Adjacent Page
- Page & DRAM Information
- Page-Table Rows Charge-Leak Counters

Flow:
- collect
- trace
- trigger
- maintain

Kernel to DRAM Bank:
- Non Page-Table
- Flip
- Page Table
- refresh
Overview

- Trace memory accesses to adjacent pages
- Maintain a counter for each page-table row
- Trigger row-refresher when the counter reaches a pre-defined limit, similar to ChipTRR
Adjacent Page Tracer
Adjacent Page Tracer

threshold

reaches count_limit

to t1 t2 t3 t↓n ↓ t

memory access
Adjacent Page Tracer

- Set-up tracing periodically
- Determine `timer_inr` and `count_limit`
Adjacent Page Tracer

- Set-up tracing periodically
- Determine `timer_inr` and `count_limit`
Adjacent Page Tracer

- Set-up tracing periodically
  - Configuring *present* bit or *rsrv* bit in leaf PTEs (page table entries) can capture a memory access of *read*, *write* or *instruction fetch*.
Adjacent Page Tracer

- Set-up tracing periodically
  - Configuring present bit or rsrv bit in leaf PTEs (page table entries) can capture a memory access of read, write or instruction fetch.

Diagram:

- TLB (paging structures)
- MMU (address translation)
- Leaf PTE (present bit: 0 or rsrv bit set: 1)
- Page-fault Handler

Access (read, write, or instruction fetch) notify update
MMU-supported page-fault error code

- present set to 0 corresponds to P bit set to 0

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>Reserved</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>SGX</td>
<td>Reserved</td>
</tr>
<tr>
<td>14</td>
<td>PK</td>
<td>V/D</td>
</tr>
<tr>
<td>13</td>
<td>RSV</td>
<td>U/S</td>
</tr>
<tr>
<td>12</td>
<td>W/R</td>
<td>P</td>
</tr>
</tbody>
</table>

- 0 means that the fault was caused by a non-present page.
- 1 means that the fault was caused by a page-level protection violation.

- 0 means that the fault was caused by reserved bit violation.
- 1 means that the fault was caused by a reserved bit set to 1 in a page-table entry
**MMU-supported page-fault error code**

- present set to 0 corresponds to P bit set to 0
- rsrv bit set to 1 corresponds to RSVD bit set to 1

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved</td>
</tr>
<tr>
<td>15</td>
<td>SGX</td>
</tr>
<tr>
<td>5</td>
<td>Reserved</td>
</tr>
<tr>
<td>4</td>
<td>P</td>
</tr>
<tr>
<td>3</td>
<td>V/D</td>
</tr>
<tr>
<td>2</td>
<td>RSVD</td>
</tr>
<tr>
<td>1</td>
<td>U/S</td>
</tr>
<tr>
<td>0</td>
<td>W/R</td>
</tr>
<tr>
<td></td>
<td>P</td>
</tr>
</tbody>
</table>

- 0 means that the fault was caused by a non-present page.
- 1 means that the fault was caused by a page-level protection violation.

- 0 means that the fault was caused by reserved bit violation.
- 1 means that the fault was caused by a reserved bit set to 1 in a page-table entry.
Adjacent Page Tracer

- Set-up tracing periodically
  - Configuring `present` bit or `rsrv` bit in leaf PTEs (page table entries) can capture memory access of *read*, *write* or *instruction fetch*.
  - Choose `rsrv` bit as configuring `present` bit causes kernel abort.
Adjacent Page Tracer

- Set-up tracing periodically
- Determine `timer_intr` and `count_limit`
  - `threshold = ?`
Adjacent Page Tracer

- Set-up tracing periodically
- Determine `timer_intr` and `count_limit`
  - \( \text{threshold} = \text{timer\_intr} \times (\text{count\_limit} - 1) \) and means no bit flip will be caused by hammering.

![Diagram showing memory access, timer_intr, and count_limit with threshold and reaches count_limit](image)
Adjacent Page Tracer

- Set-up tracing periodically
- Determine $\text{timer}_{\text{intr}}$ and $\text{count}_{\text{limit}}$
  - $\text{threshold} = \text{timer}_{\text{intr}} \times (\text{count}_{\text{limit}} - 1)$ and means no bit flip will be caused by hammering
  - A safe threshold is 1 ms (based on Kim et al. ISCA'20)
Adjacent Page Tracer

- Set-up tracing periodically
- Determine \texttt{timer\_intr} and \texttt{count\_limit}
  
  ✓ threshold = \texttt{timer\_intr} \times (\texttt{count\_limit} - 1) and means no bit flip will be caused by hammering
  
  ✓ A safe threshold is 1 ms
  
  ✓ \texttt{timer\_intr} is set to 1 ms and \texttt{count\_limit} is set to 2
Overview
Overview
Overview
Row Refresher

- Refresh A Specified Row
  - A simple read-access to a kernel virtual address can re-charge a specified row and prevent potential bit flips
  - A kernel virtual address should be mapped to the specified row
Row Refresher

- Refresh A Specified Row
  - A simple read-access to a kernel virtual address can re-charge a corresponding row and prevent potential bit flips
  - A kernel virtual address should be mapped to the specified row

- Direct-physical Map
  - Linux maps available physical memory into the kernel space
  - A kernel virtual address can be found based on the mapping between a physical address and a DRAM row location, and the direct-physical map
Evaluation
Security Evaluation

Three popular rowhammer attacks target corrupting level-1 page tables:

- **Memory Spray (Blackhat'15):** explicitly hammers user memory adjacent to L1PTEs
- **CATTmew (IEEE TDSC'19):** explicitly hammers device driver buffer adjacent to L1PTEs
- **Pthammer (MICRO'20):** implicitly hammers L1PTEs adjacent to other L1PTEs
Security Evaluation

Three popular rowhammer attacks target corrupting level-1 page tables:

- Memory Spray (Blackhat’15): explicitly hammers user memory adjacent to L1PTEs
- CATTmew (IEEE TDSC’19): explicitly hammers device driver buffer adjacent to L1PTEs
- Pthammer (MICRO’20): implicitly hammers L1PTEs adjacent to other L1PTEs

<table>
<thead>
<tr>
<th>Machine Model</th>
<th>Hardware Configuration</th>
<th>Attack</th>
<th>SoftTRR Bit Flip Failed?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dell Optiplex 390</td>
<td>CPU Arch: KabyLake, CPU Model: i7-7700k, DRAM: Kingston DDR4 (99P5701-005.A00G)</td>
<td>Memory Spray [46]</td>
<td>✓</td>
</tr>
<tr>
<td>Dell Optiplex 990</td>
<td>CPU Arch: SandyBridge, CPU Model: i5-2400, DRAM: Samsung DDR3 (M378B5273DH0-CH9)</td>
<td>CATTmew [13]</td>
<td>✓</td>
</tr>
<tr>
<td>Thinkpad X230</td>
<td>CPU Arch: IvyBridge, CPU Model: i5-3230M, DRAM: Samsung DDR3 (M471B5273DH0-CH9)</td>
<td>PThammer [62]</td>
<td>✓</td>
</tr>
</tbody>
</table>

n = 50
Performance Evaluation

Three representative benchmarks:

- SPECspeed 2017 Integer: CPU-focused
- memcached: memory-focused
- Phoronix test suite: system as a whole
Performance Evaluation

Three representative benchmarks:
- SPECspeed 2017 Integer: CPU-focused
- memcached: memory-focused
- Phoronix test suite: system as a whole

Runtime overhead on benchmarks in two scenarios:
- $\Delta^{\pm 1}$: where an adjacent row is only 1-row from a row hosting level-1 page tables.
- $\Delta^{\pm 6}$: where an adjacent row is up to 6-row from a row hosting level-1 page tables.
# Runtime Overhead

<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>Programs</th>
<th>SoftTRR Overhead $\Delta_{\pm 1}$</th>
<th>$\Delta_{\pm 6}$ (default)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$\Delta_{\pm 1}$</td>
<td>$\Delta_{\pm 6}$ (default)</td>
</tr>
<tr>
<td><em>SPECspeed 2017 Integer</em></td>
<td></td>
<td>0.67%</td>
<td>0.67%</td>
</tr>
<tr>
<td></td>
<td>gcc_s</td>
<td>0.23%</td>
<td>0.92%</td>
</tr>
<tr>
<td></td>
<td>mcf_s</td>
<td>-0.76%</td>
<td>0.30%</td>
</tr>
<tr>
<td></td>
<td>omnetpp_s</td>
<td>-0.81%</td>
<td>1.82%</td>
</tr>
<tr>
<td></td>
<td>xalancbmk_s</td>
<td>0.36%</td>
<td>2.50%</td>
</tr>
<tr>
<td></td>
<td>xz64_s</td>
<td>0.00%</td>
<td>0.61%</td>
</tr>
<tr>
<td></td>
<td>deepsjeng_s</td>
<td>0.00%</td>
<td>0.28%</td>
</tr>
<tr>
<td></td>
<td>leela_s</td>
<td>0.23%</td>
<td>0.46%</td>
</tr>
<tr>
<td></td>
<td>exchange2_s</td>
<td>-0.70%</td>
<td>-0.23%</td>
</tr>
<tr>
<td></td>
<td>xz_s</td>
<td>1.48%</td>
<td>0.93%</td>
</tr>
<tr>
<td><strong>Mean</strong></td>
<td></td>
<td>0.07%</td>
<td>0.83%</td>
</tr>
<tr>
<td><em>Phoronix</em></td>
<td></td>
<td>0.16%</td>
<td>0.32%</td>
</tr>
<tr>
<td></td>
<td>unpack-linux</td>
<td>1.31%</td>
<td>1.84%</td>
</tr>
<tr>
<td></td>
<td>iozone</td>
<td>0.89%</td>
<td>-1.15%</td>
</tr>
<tr>
<td></td>
<td>postmark</td>
<td>0.89%</td>
<td>0.00%</td>
</tr>
<tr>
<td></td>
<td>stream:Copy</td>
<td>0.01%</td>
<td>0.00%</td>
</tr>
<tr>
<td></td>
<td>stream:Scale</td>
<td>0.60%</td>
<td>0.23%</td>
</tr>
<tr>
<td></td>
<td>stream:Triad</td>
<td>0.07%</td>
<td>0.37%</td>
</tr>
<tr>
<td></td>
<td>stream:Add</td>
<td>0.03%</td>
<td>0.35%</td>
</tr>
<tr>
<td></td>
<td>compress-7zip</td>
<td>1.52%</td>
<td>2.24%</td>
</tr>
<tr>
<td></td>
<td>openssl</td>
<td>0.14%</td>
<td>0.13%</td>
</tr>
<tr>
<td></td>
<td>pybench</td>
<td>0.00%</td>
<td>0.52%</td>
</tr>
<tr>
<td></td>
<td>phpbench</td>
<td>0.92%</td>
<td>0.01%</td>
</tr>
<tr>
<td></td>
<td>cachebench:read</td>
<td>-0.38%</td>
<td>0.26%</td>
</tr>
<tr>
<td></td>
<td>cachebench:write</td>
<td>-0.26%</td>
<td>-0.44%</td>
</tr>
<tr>
<td></td>
<td>cachebench:modify</td>
<td>-0.01%</td>
<td>0.67%</td>
</tr>
<tr>
<td></td>
<td>ramspeed:INT</td>
<td>-0.09%</td>
<td>-0.63%</td>
</tr>
<tr>
<td></td>
<td>ramspeed:FP</td>
<td>-0.15%</td>
<td>0.63%</td>
</tr>
<tr>
<td><strong>Mean</strong></td>
<td></td>
<td>0.22%</td>
<td>0.24%</td>
</tr>
<tr>
<td><strong>Statistics</strong></td>
<td></td>
<td>0.39%</td>
<td>0.18%</td>
</tr>
<tr>
<td></td>
<td>Ops</td>
<td>0.39%</td>
<td>0.15%</td>
</tr>
<tr>
<td></td>
<td>TPS</td>
<td>0.39%</td>
<td>0.15%</td>
</tr>
<tr>
<td></td>
<td>Net_rate</td>
<td>0.46%</td>
<td>0.31%</td>
</tr>
</tbody>
</table>

60
Runtime Memory Consumption

- In a LAMP (Linux, Apache, MySQL and PHP) system with SoftTRR deployed
- Nikto stresses the LAMP system from another machine
Runtime Memory Consumption

- In a LAMP (Linux, Apache, MySQL and PHP) system with SoftTRR deployed
- Nikto stresses the LAMP system from another machine
## System Robustness

<table>
<thead>
<tr>
<th>Linux Test Project</th>
<th>Vanilla System</th>
<th>SoftTRR</th>
<th>SoftTRR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$\Delta_{+1}$</td>
<td>$\Delta_{+6}$ (default)</td>
</tr>
<tr>
<td><strong>File</strong></td>
<td>open</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>close</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>ftruncate</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>rename</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Network</strong></td>
<td>Listen</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Socket</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Send</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Recv</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Memory</strong></td>
<td>mmap</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>munmap</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>brk</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>mlock</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>munlock</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>mremap</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Process</strong></td>
<td>getpid</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>exit</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>clone</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Misc.</strong></td>
<td>ioctl</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>prctl</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>vhangup</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

✓ the stress test does not report any problem
Conclusion

★ SoftTRR is a more effective and practical software-only mitigation, Compared to existing works

★ In its implementation, SoftTRR works as a loadable kernel module to defend against rowhammer attacks on L1PT pages. SoftTRR leverages MMU and OS kernel features to collect L1PT pages, track memory access, and refresh target L1PT pages

★ SoftTRR is evaluated to be effective against 3 representative rowhammer attacks and incur small overhead and memory footprints
Thanks & Questions?