CSI:Rowhammer

Closing the Case of Half-Double and Beyond

BlackHat Europe 2022
7th December 2022

Andreas Kogler
Graz University of Technology

Jonas Juffinger
Graz University of Technology
Motivation - Rowhammer

Since 2014

Numerous exploits

Mitigations ineffektive

Something fundamentally better?
What if we could transparently correct arbitrary bitflips if we know that the OS can reconstruct the data from a different source?
Introduction
Dynamic Random Access Memory (DRAM)

- Large, cheap and energy efficient
Dynamic Random Access Memory (DRAM)

- **Large, cheap and energy efficient**
- Cell: one transistor & one capacitor
Dynamic Random Access Memory (DRAM)

- **Large, cheap and energy efficient**
- Cell: one transistor & one capacitor
- Capacitor looses charge over time

Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary)
Dynamic Random Access Memory (DRAM)

- **Large**, **cheap** and **energy efficient**
- Cell: one transistor & one capacitor
- Capacitor looses charge over time
- Periodic refreshes **required**
Dynamic Random Access Memory (DRAM)

- **Large, cheap and energy efficient**
- Cell: one transistor & one capacitor
- Capacitor loses charge over time
- Periodic refreshes **required**
- Organized in **rows**
• Hardware fault of the DRAM [3]

<table>
<thead>
<tr>
<th></th>
<th>aggressor1</th>
<th>Victim</th>
<th>aggressor2</th>
</tr>
</thead>
</table>

```c
for (i = 0; i < N; ++i) {
    *aggressor1;
    *aggressor2;
    flush(aggressor1);
    flush(aggressor2);
}
```
Rowhammer

- Hardware fault of the DRAM [3]
- Frequent accesses flip bits in neighboring rows

```
for (i = 0; i < N; ++i) {
  *aggressor1;
  *aggressor2;
  flush(aggressor1);
  flush(aggressor2);
}
```
Rowhammer

- Hardware fault of the DRAM [3]
- Frequent accesses flip bits in neighboring rows

```c
for (i = 0; i < N; ++i) {
    *aggressor1;
    *aggressor2;
    flush(aggressor1);
    flush(aggressor2);
}
```

Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary_)
- Hardware fault of the DRAM [3]
- Frequent accesses flip bits in neighboring rows

```plaintext
for (i = 0; i < N; ++i) {
  aggressor1;
  aggressor2;
  flush(aggressor1);
  flush(aggressor2);
}
```
Rowhammer

- Hardware fault of the DRAM [3]
- Frequent accesses flip bits in neighboring rows

```
for (i = 0; i < N; ++i) {
  *aggressor1;
  *aggressor2;
  flush(aggressor1);
  flush(aggressor2);
}
```
Rowhammer

- Hardware fault of the DRAM [3]
- Frequent accesses flip bits in neighboring rows

```
1   for (i = 0; i < N; ++i) {
2       *aggressor1;
3       *aggressor2;
4       flush(aggressor1);
5       flush(aggressor2);
6   }
```
Rowhammer

- Hardware fault of the DRAM [3]
- Frequent accesses flip bits in neighboring rows

\[
\begin{array}{l}
\text{for } (i = 0; i < N; ++i) \{ \\
\quad \star \text{aggressor1;}
\quad \star \text{aggressor2;}
\quad \text{flush(aggressor1);} \\
\quad \text{flush(aggressor2);} \\
\} \\
\end{array}
\]
- Hardware fault of the DRAM [3]
- Frequent accesses flip bits in neighboring rows

```
for (i = 0; i < N; ++i) {
  *aggressor1;
  *aggressor2;
  flush(aggressor1);
  flush(aggressor2);
}
```
Rowhammer

- Hardware fault of the DRAM [3]
- Frequent accesses flip bits in neighboring rows

```
for (i = 0; i < N; ++i) {
    *aggressor1;
    *aggressor2;
    flush(aggressor1);
    flush(aggressor2);
}
```
Rowhammer

- Hardware fault of the DRAM [3]
- Frequent accesses flip bits in neighboring rows

```c
for (i = 0; i < N; ++i) {
    *aggressor1;
    *aggressor2;
    flush(aggressor1);
    flush(aggressor2);
}
```

Andreas Kogler (@0xhillert) Jonas Juffinger (@notimaginary)
Rowhammer

- Hardware fault of the DRAM [3]
- Frequent accesses flip bits in neighboring rows

```
for (i = 0; i < N; ++i) {
    *aggressor1;
    *aggressor2;
    flush(aggressor1);
    flush(aggressor2);
}
```
- Hardware fault of the DRAM [3]
- Frequent accesses flip bits in neighboring rows
- Worse with every new DRAM generation

```
for (i = 0; i < N; ++i) {
    *aggressor1;
    *aggressor2;
    flush(aggressor1);
    flush(aggressor2);
}
```
Rowhammer

- Hardware fault of the DRAM [3]
- Frequent accesses flip bits in neighboring rows
- Worse with every new DRAM generation
- Enables attacks, many countermeasures proposed

```c
for (i = 0; i < N; ++i) {
    *aggressor1;
    *aggressor2;
    flush(aggressor1);
    flush(aggressor2);
}
```
Rowhammer Patterns

<table>
<thead>
<tr>
<th>aggressor1</th>
<th>Victim</th>
<th>aggressor1</th>
<th>Victim</th>
<th>aggressor1</th>
<th>Victim</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Andreas Kogler (@0xhilbert)  Jonas Juffinger (@notimaginary_)
Mitigations focus on the **characteristics** of Rowhammer
Characteristics 1: Flips are Infrequent
Characteristics 1: Flips are Infrequent

- ECC Error Correcting Codes
Characteristics 1: Flips are Infrequent

- **ECC** Error Correcting Codes
- Additional hardware redundancy
Characteristics 1: Flips are Infrequent

- **ECC** Error Correcting Codes
  - Additional hardware redundancy
  - Catch low number of flips in HW
Characteristics 1: Flips are Infrequent

- **ECC** Error Correcting Codes
  - Additional hardware redundancy
  - Catch low number of flips in HW
- Increase the **refresh rate**

Andreas Kogler (@0xhilbert)   Jonas Juffinger (@notimaginary_)
Characteristics 1: Flips are Infrequent

- **ECC** Error Correcting Codes
  - Additional hardware redundancy
  - Catch **low** number of flips in HW
- Increase the **refresh rate**
  - Reduce the attackers time window

Andreas Kogler (@0xhilbert)   Jonas Juffinger (@notimaginary_)
Characteristics 1: Flips are Infrequent

- **ECC** Error Correcting Codes
  - Additional hardware redundancy
  - Catch low number of flips in HW
- Increase the **refresh rate**
  - Reduce the attackers time window
  - Default 64\text{ms}
Characteristics 2: Attacks are Detectable
Characteristics 2: Attacks are Detectable

- Use Performance Monitoring Counters (PMCs)
Characteristics 2: Attacks are Detectable

- Use Performance Monitoring Counters (PMCs)
- Monitor DRAM activity
Characteristics 2: Attacks are Detectable

- Use Performance Monitoring Counters (PMCs)
  - Monitor DRAM activity
- Stop *suspicious* applications
Characteristics 3: Flips happen at Distance 1
Characteristics 3: Flips happen at Distance 1

- Flips only happen in neighboring rows
- Mitigations based row layout
  - Guard rows
  - Targeted Row Refresh
Guard Rows

- Allocate **guard rows**

<table>
<thead>
<tr>
<th>Agressor</th>
<th>Guard</th>
<th>Victim</th>
<th>Guard</th>
<th>Agressor</th>
</tr>
</thead>
</table>

Andreas Kogler (Twitter: @0xhilbert)  Jonas Juffinger (Twitter: @notimaginary_)

8
Guard Rows

- Allocate **guard rows**
- Can not reach victim

<table>
<thead>
<tr>
<th>Agressor</th>
<th>Guard</th>
<th>Victim</th>
<th>Guard</th>
<th>Agressor</th>
</tr>
</thead>
</table>

Andreas Kogler (@0xhilbert)    Jonas Juffinger (@notimaginary_)
Guard Rows

- Allocate **guard rows**
- Can not reach victim

<table>
<thead>
<tr>
<th>Agressor</th>
<th>Guard</th>
<th>Victim</th>
<th>Guard</th>
<th>Agressor</th>
</tr>
</thead>
</table>

Andreas Kogler (@0xhilbert)    Jonas Juffinger (@notimaginary_)
Guard Rows

- Allocate **guard rows**
- Can not reach victim

<table>
<thead>
<tr>
<th>Agressor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guard</td>
</tr>
<tr>
<td>Victim</td>
</tr>
<tr>
<td>Guard</td>
</tr>
<tr>
<td>Agressor</td>
</tr>
</tbody>
</table>
Guard Rows

- Allocate **guard rows**
- Can not reach victim

<table>
<thead>
<tr>
<th>Agressor</th>
<th>Guard</th>
<th>Victim</th>
<th>Guard</th>
<th>Agressor</th>
</tr>
</thead>
</table>

Andreas Kogler (@0xhilbert)  Jonas Juffinger (@notimaginary_)
Guard Rows

- Allocate **guard rows**
- Can not reach victim

<table>
<thead>
<tr>
<th>Agressor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guard</td>
</tr>
<tr>
<td>Victim</td>
</tr>
<tr>
<td>Guard</td>
</tr>
<tr>
<td>Agressor</td>
</tr>
</tbody>
</table>
• Allocate **guard rows**
• Can not reach victim

<table>
<thead>
<tr>
<th>Agressor</th>
<th>Guard</th>
<th>Victim</th>
<th>Guard</th>
<th>Agressor</th>
</tr>
</thead>
</table>

Andreas Kogler (@0xhilbert)  Jonas Juffinger (@notimaginary_)
Guard Rows

- Allocate **guard rows**
- Can not reach victim

<table>
<thead>
<tr>
<th>Agressor</th>
<th>Guard</th>
<th>Victim</th>
<th>Guard</th>
<th>Agressor</th>
</tr>
</thead>
</table>

Andreas Kogler (@0xhilbert)  Jonas Juffinger (@notimaginary_)
Guard Rows

- Allocate **guard rows**
- Can not reach victim

<table>
<thead>
<tr>
<th>Aggressor</th>
<th>Guard</th>
<th>Victim</th>
<th>Guard</th>
<th>Aggressor</th>
</tr>
</thead>
</table>

Andreas Kogler (@0xhilbert)  Jonas Juffinger (@notimaginary_)
Guard Rows

- Allocate **guard rows**
- Can not reach victim

<table>
<thead>
<tr>
<th>Agressor</th>
<th>Guard</th>
<th>Victim</th>
<th>Guard</th>
<th>Agressor</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Andreas Kogler (@0xhilbert)  Jonas Juffinger (@notimaginary_)
Guard Rows

- Allocate **guard rows**
- Can not reach victim
• Additional victim refreshes
• Standardized in LPDDR4x

<table>
<thead>
<tr>
<th>Near Aggressor</th>
<th>(N+)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Victim</td>
<td>(V)</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>(N−)</td>
</tr>
</tbody>
</table>
• Additional victim refreshes
• Standardized in LPDDR4x

<table>
<thead>
<tr>
<th></th>
<th>(N⁺)</th>
<th>(V)</th>
<th>(N⁻)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Near Aggressor</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Victim</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Near Aggressor</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Target Row Refresh (TRR)

- Additional victim refreshes
- Standardized in LPDDR4x

<table>
<thead>
<tr>
<th></th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_+$</td>
</tr>
<tr>
<td>Victim</td>
<td>$\mathcal{V}$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_-$</td>
</tr>
</tbody>
</table>

Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary_)
Target Row Refresh (TRR)

- Additional victim refreshes
- Standardized in LPDDR4x

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Near Aggressor</td>
<td>$(N_+)$</td>
</tr>
<tr>
<td>Victim</td>
<td>$(V)$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$(N_-)$</td>
</tr>
</tbody>
</table>
Target Row Refresh (TRR)

- Additional victim refreshes
- Standardized in LPDDR4x

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_+$</td>
</tr>
<tr>
<td>Victim</td>
<td>$\mathcal{V}$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_-$</td>
</tr>
</tbody>
</table>
Target Row Refresh (TRR)

- Additional victim refreshes
- Standardized in LPDDR4x

<table>
<thead>
<tr>
<th></th>
<th>(N⁺)</th>
<th>(V)</th>
<th>(N⁻)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Near Aggressor</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Victim</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Near Aggressor</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Target Row Refresh (TRR)

- Additional victim refreshes
- Standardized in LPDDR4x

<table>
<thead>
<tr>
<th></th>
<th>$\mathcal{N}_+$</th>
<th>$\mathcal{V}$</th>
<th>$\mathcal{N}_-$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Near Aggressor</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Victim</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Near Aggressor</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
• Additional victim refreshes
• Standardized in LPDDR4x

<table>
<thead>
<tr>
<th></th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_+$</td>
</tr>
<tr>
<td>Victim</td>
<td>$\mathcal{V}$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_-$</td>
</tr>
</tbody>
</table>
Target Row Refresh (TRR)

- Additional victim refreshes
- Standardized in LPDDR4x

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_+$</td>
</tr>
<tr>
<td>Victim</td>
<td>$\mathcal{V}$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_-$</td>
</tr>
</tbody>
</table>
Target Row Refresh (TRR)

- Additional victim refreshes
- Standardized in LPDDR4x

<table>
<thead>
<tr>
<th></th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_+$</td>
</tr>
<tr>
<td>Victim</td>
<td>$\mathcal{V}$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_-$</td>
</tr>
</tbody>
</table>

Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary_)
- Additional victim refreshes
- Standardized in LPDDR4x

<table>
<thead>
<tr>
<th>Near Aggressor</th>
<th>$\mathcal{N}_+$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Victim</td>
<td>$\mathcal{V}$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_-$</td>
</tr>
</tbody>
</table>
Target Row Refresh (TRR)

- Additional victim refreshes
- Standardized in LPDDR4x

<table>
<thead>
<tr>
<th></th>
<th>( N_+ )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Near Aggressor</td>
<td>( N_+ )</td>
</tr>
<tr>
<td>Victim</td>
<td>( V )</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>( N_- )</td>
</tr>
</tbody>
</table>

Andreas Kogler (@0xhilbert)  Jonas Juffinger (@notimaginary_)
Target Row Refresh (TRR)

- Additional victim refreshes
- Standardized in LPDDR4x

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Near Aggressor</td>
<td>$N_+$</td>
</tr>
<tr>
<td>Victim</td>
<td>$N$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$N_-$</td>
</tr>
</tbody>
</table>
Target Row Refresh (TRR)

- Additional victim refreshes
- Standardized in LPDDR4x
Let’s start the cat and mouse game!

ROWHAMMER

MITIGATIONS
The Problem with Rowhammer Mitigations
• Bit Flips are infrequent - ECC, Increased Refresh Rate
The Problem with Rowhammer Countermeasures

- Bit Flips are infrequent—ECC, Increased Refresh Rate [3]
• Bit Flips are infrequent—ECC, Increased Refresh Rate [3]
• Detectable with Performance Counters - ANVIL, HexPADS
The Problem with Rowhammer Countermeasures

- Bit Flips are infrequent—ECC, Increased Refresh Rate [3]
- Detectable with Performance Counters—ANVIL, HexPADS [2]
The Problem with Rowhammer Countermeasures

- Bit Flips are infrequent—ECC, Increased Refresh Rate [3]
- Detectable with Performance Counters—ANVIL, HexPADS [2]
- Hammer Distance is 1 - TRR, ZebRAM, B-CATT
The Problem with Rowhammer Countermeasures

- Bit Flips are infrequent—ECC, Increased Refresh Rate [3]
- Detectable with Performance Counters—ANVIL, HexPADS [2]
- Hammer Distance is 1—TRR, ZebRAM, B-CATT [4]
Would perfect TRR fix Rowhammer?
No
The Half-Double Effect
• **Distance-1**
  
  - \((\mathcal{N}_+ \rightarrow \mathcal{N}_-)^\infty\)
  - *Classic* double-sided Rowhammer
### Distance 1

<table>
<thead>
<tr>
<th>Role</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Far Aggressor</td>
<td>$F_+$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$N_+$</td>
</tr>
<tr>
<td>Victim</td>
<td>$V$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$N_-$</td>
</tr>
<tr>
<td>Far Aggressor</td>
<td>$F_-$</td>
</tr>
</tbody>
</table>

- **Distance-1**
  - $(N_+ \rightarrow N_-)^\infty$
  - *Classic* double-sided Rowhammer
- **First** flip after:
  - 18,000 hammers in 1.2 ms
Distance 1

- **Distance-1**
  - \((\mathcal{N}_+ \rightarrow \mathcal{N}_-)\)^\infty
  - *Classic* double-sided Rowhammer

- **First** flip after:
  - 18 000 hammers in 1.2 ms
  - ✓ *Within* the refresh window
  - ✗ *Mitigated* by TRR
Distance 2

- **Far Aggressor** \((F_+\)\)
- **Near Aggressor** \((N_+\)\)
- **Victim** \((V\)\)
- **Near Aggressor** \((N_-\)\)
- **Far Aggressor** \((F_-\)\)

- **Distance-2**
  - \((F_+ \rightarrow F_-)^\infty\)
  - Distance two double-sided Rowhammer

Andreas Kogler (@0xhilbert)    Jonas Juffinger (@notimaginary_)
## Distance 2

<table>
<thead>
<tr>
<th></th>
<th>( \mathcal{F}_+ )</th>
<th>( \mathcal{V} )</th>
<th>( \mathcal{N}_+ )</th>
<th>( \mathcal{F}_- )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Far Aggressor</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Near Aggressor</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Victim</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Near Aggressor</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Far Aggressor</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Distance-2**
  - \( (\mathcal{F}_+ \rightarrow \mathcal{F}_-)^\infty \)
  - Distance two double-sided Rowhammer
- **First** flip after:
  - 4,000,000 hammers in 270 ms
Distance 2

- **Distance-2**
  - \((F_+ \rightarrow F_-)^{\infty}\)
  - Distance two double-sided Rowhammer
- **First** flip after:
  - 4,000,000 hammers in 270 ms
  - ❌ Not within the refresh windows
Half-Double

- Half-Double
  - $((\mathcal{F}_+ \rightarrow \mathcal{F}_-)^\beta \rightarrow \mathcal{N}_+ \rightarrow \mathcal{N}_-)^\infty$
  - Many distance-2 accesses with a few distance-1 accesses

<table>
<thead>
<tr>
<th></th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Far Aggressor</td>
<td>$\mathcal{F}_+$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_+$</td>
</tr>
<tr>
<td>Victim</td>
<td>$\mathcal{V}$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_-$</td>
</tr>
<tr>
<td>Far Aggressor</td>
<td>$\mathcal{F}_-$</td>
</tr>
</tbody>
</table>

Andreas Kogler (@0xhilbert)    Jonas Juffinger (@notimaginary_)
Half-Double

- **Half-Double**
  - \(((\mathcal{F}_+ \rightarrow \mathcal{F}_-)^\beta \rightarrow \mathcal{N}_+ \rightarrow \mathcal{N}_-)\)^\infty
- **Many** distance-2 accesses with a **few** distance-1 accesses
- **First** flip after:
  - 296,960 hammers in 20 ms
  - Dilution $\beta = 57$ (5120 distance-1 accesses)

<table>
<thead>
<tr>
<th>Aggressor</th>
<th>Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Far Aggressor</td>
<td>(\mathcal{F}_+)</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>(\mathcal{N}_+)</td>
</tr>
<tr>
<td>Victim</td>
<td>(\mathcal{V})</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>(\mathcal{N}_-)</td>
</tr>
<tr>
<td>Far Aggressor</td>
<td>(\mathcal{F}_-)</td>
</tr>
</tbody>
</table>

Andreas Kogler (@0xhilbert)    Jonas Juffinger (@notimaginary_)
Half-Double

- **Half-Double**
  - \(((\mathcal{F}_+ \rightarrow \mathcal{F}_-)^\beta \rightarrow \mathcal{N}_+ \rightarrow \mathcal{N}_-)\)^\infty
  - **Many** distance-2 accesses with a **few** distance-1 accesses

- **First** flip after:
  - 296,960 hammers in 20 ms
  - Dilution \(\beta = 57\) (5120 distance-1 accesses)

✓ **Within** the refresh window
**Half-Double**

- **Half-Double**
  - $((F_+ \rightarrow F_-) \beta \rightarrow N_+ \rightarrow N_-)^\infty$
  - *Many* distance-2 accesses with a *few* distance-1 accesses

- **First** flip after:
  - 296 960 hammers in 20 ms
  - Dilution $\beta = 57$ (5120 distance-1 accesses)
  - ✓ *Within* the refresh window
  - ✓ *Assisted* by TRR
Half-Double

- **Half-Double**
  - \(((\mathcal{F}_+ \rightarrow \mathcal{F}_-)^\beta \rightarrow \mathcal{N}_+ \rightarrow \mathcal{N}_-)^{\infty}\)
  - Many distance-2 accesses with a few distance-1 accesses

- **First** flip after:
  - 296,960 hammers in 20 ms
  - Dilution \( \beta = 57 \) (5120 distance-1 accesses)
  - ✓ Within the refresh window
  - ✓ Assisted by TRR
## TRR Assisted Half-Double

<table>
<thead>
<tr>
<th>Role</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Far Aggressor</td>
<td>$\mathcal{F}_+$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_+$</td>
</tr>
<tr>
<td>Victim</td>
<td>$\mathcal{V}$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_-$</td>
</tr>
<tr>
<td>Far Aggressor</td>
<td>$\mathcal{F}_-$</td>
</tr>
</tbody>
</table>

Andreas Kogler (@0xhilbert)  Jonas Juffinger (@notimaginary_)
<table>
<thead>
<tr>
<th>Role</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Far Aggressor</td>
<td>$F_+$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$N_+$</td>
</tr>
<tr>
<td>Victim</td>
<td>$V$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$N_-$</td>
</tr>
<tr>
<td>Far Aggressor</td>
<td>$F_-$</td>
</tr>
<tr>
<td></td>
<td>Symbol</td>
</tr>
<tr>
<td>----------------</td>
<td>--------</td>
</tr>
<tr>
<td>Far Aggressor</td>
<td>$\mathcal{F}_+$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_+$</td>
</tr>
<tr>
<td>Victim</td>
<td>$\mathcal{V}$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_-$</td>
</tr>
<tr>
<td>Far Aggressor</td>
<td>$\mathcal{F}_-$</td>
</tr>
</tbody>
</table>
### TRR Assisted Half-Double

<table>
<thead>
<tr>
<th>Role</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Far Aggressor</td>
<td>$\mathcal{F}_+$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_+$</td>
</tr>
<tr>
<td>Victim</td>
<td>$\mathcal{V}$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_-$</td>
</tr>
<tr>
<td>Far Aggressor</td>
<td>$\mathcal{F}_-$</td>
</tr>
</tbody>
</table>

Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary_)
<table>
<thead>
<tr>
<th>Role</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Far Aggressor</td>
<td>$\mathcal{F}_+$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_+$</td>
</tr>
<tr>
<td>Victim</td>
<td>$\mathcal{V}$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_-$</td>
</tr>
<tr>
<td>Far Aggressor</td>
<td>$\mathcal{F}_-$</td>
</tr>
</tbody>
</table>
TRR Assisted Half-Double

<table>
<thead>
<tr>
<th>Role</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Far Aggressor</td>
<td>$\mathcal{F}_+$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_+$</td>
</tr>
<tr>
<td>Victim</td>
<td>$\mathcal{V}$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_-$</td>
</tr>
<tr>
<td>Far Aggressor</td>
<td>$\mathcal{F}_-$</td>
</tr>
</tbody>
</table>

Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary_)
# TRR Assisted Half-Double

<table>
<thead>
<tr>
<th>Role</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Far Aggressor</td>
<td>$\mathcal{F}_+$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_+$</td>
</tr>
<tr>
<td>Victim</td>
<td>$\mathcal{V}$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_-$</td>
</tr>
<tr>
<td>Far Aggressor</td>
<td>$\mathcal{F}_-$</td>
</tr>
</tbody>
</table>

Andreas Kogler (@0xhilbert)  Jonas Juffinger (@notimaginary)
<table>
<thead>
<tr>
<th>Role</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Far Aggressor</td>
<td>$\mathcal{F}_+$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_+$</td>
</tr>
<tr>
<td>Victim</td>
<td>$\mathcal{V}$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_-$</td>
</tr>
<tr>
<td>Far Aggressor</td>
<td>$\mathcal{F}_-$</td>
</tr>
<tr>
<td>Role</td>
<td>Symbol</td>
</tr>
<tr>
<td>--------------</td>
<td>--------</td>
</tr>
<tr>
<td>Far Aggressor</td>
<td>$\mathcal{F}_+$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_+$</td>
</tr>
<tr>
<td>Victim</td>
<td>$\mathcal{V}$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_-$</td>
</tr>
<tr>
<td>Far Aggressor</td>
<td>$\mathcal{F}_-$</td>
</tr>
</tbody>
</table>
### TRR Assisted Half-Double

<table>
<thead>
<tr>
<th>Role</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Far Aggressor</td>
<td>$\mathcal{F}_+$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_+$</td>
</tr>
<tr>
<td>Victim</td>
<td>$\mathcal{V}$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_-$</td>
</tr>
<tr>
<td>Far Aggressor</td>
<td>$\mathcal{F}_-$</td>
</tr>
</tbody>
</table>

Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary_)
TRR Assisted Half-Double

<table>
<thead>
<tr>
<th>Role</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Far Aggressor</td>
<td>$\mathcal{F}_+$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_+$</td>
</tr>
<tr>
<td>Victim</td>
<td>$\mathcal{V}$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_-$</td>
</tr>
<tr>
<td>Far Aggressor</td>
<td>$\mathcal{F}_-$</td>
</tr>
<tr>
<td>Role</td>
<td>Notation</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------</td>
</tr>
<tr>
<td>Far Aggressor</td>
<td>$\mathcal{F}_+$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_+$</td>
</tr>
<tr>
<td>Victim</td>
<td>$\mathcal{V}$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_-$</td>
</tr>
<tr>
<td>Far Aggressor</td>
<td>$\mathcal{F}_-$</td>
</tr>
</tbody>
</table>
### TRR Assisted Half-Double

<table>
<thead>
<tr>
<th>Role</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Far Aggressor</td>
<td>$\mathcal{F}_+$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_+$</td>
</tr>
<tr>
<td>Victim</td>
<td>$\mathcal{V}$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_-$</td>
</tr>
<tr>
<td>Far Aggressor</td>
<td>$\mathcal{F}_-$</td>
</tr>
</tbody>
</table>
TRR Assisted Half-Double

<table>
<thead>
<tr>
<th>Role</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Far Aggressor</td>
<td>$\mathcal{F}_+$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_+$</td>
</tr>
<tr>
<td>Victim</td>
<td>$\mathcal{V}$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_-$</td>
</tr>
<tr>
<td>Far Aggressor</td>
<td>$\mathcal{F}_-$</td>
</tr>
</tbody>
</table>

TRR
TRR Assisted Half-Double

<table>
<thead>
<tr>
<th>Role</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Far Aggressor</td>
<td>$\mathcal{F}_+$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_+$</td>
</tr>
<tr>
<td>Victim</td>
<td>$\mathcal{V}$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_-$</td>
</tr>
<tr>
<td>Far Aggressor</td>
<td>$\mathcal{F}_-$</td>
</tr>
</tbody>
</table>

Andreas Kogler (@0xhilbert)  Jonas Juffinger (@notimaginary_)
## TRR Assisted Half-Double

<table>
<thead>
<tr>
<th>Role</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Far Aggressor</td>
<td>$\mathcal{F}_+$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_+$</td>
</tr>
<tr>
<td>Victim</td>
<td>$\mathcal{V}$</td>
</tr>
<tr>
<td>Near Aggressor</td>
<td>$\mathcal{N}_-$</td>
</tr>
<tr>
<td>Far Aggressor</td>
<td>$\mathcal{F}_-$</td>
</tr>
</tbody>
</table>
Exploitable in the Wild?
End-to-End Exploit – Overview

- Target PFN in Page Table Entry [6]
End-to-End Exploit – Overview

- Target PFN in Page Table Entry [6]
- **C1**: Allocation of Contiguous Memory

Andreas Kogler (@0xhilbert)  Jonas Juffinger (@notimaginary_)
• Target PFN in Page Table Entry [6]
• C1: Allocation of Contiguous Memory
• C2: Alternative to Memory Templating
End-to-End Exploit – Overview

- Target PFN in Page Table Entry [6]
- **C1**: Allocation of Contiguous Memory
- **C2**: Alternative to Memory Templating
- **C3**: Memory Massaging

Andreas Kogler (@0xhilbert)  Jonas Juffinger (@notimaginary_)
End-to-End Exploit – Overview

- Target PFN in Page Table Entry [6]
- **C1**: Allocation of Contiguous Memory
- **C2**: Alternative to Memory Templating
- **C3**: Memory Massaging
- **C4**: Bit-Flip Verification
• Corrupt page table entries can **kill** the attacker process
• Corrupt page table entries can kill the attacker process

```c
if ( /*misprediction*/ ) {
    access(probe + (*ptr & 1));
}
if ( is_cached(probe) ) {
    // ptr[0-4] valid
}
```

• **Verify** if address save to access

• **Spectre** gadget
• Corrupt page table entries can kill the attacker process

```c
if ( /*misprediction*/ ) {
    access(probe + (*ptr & 1));
}
if ( is_cached(probe) ) {
    // ptr[0-4] valid
}
```

• **Verify** if address save to access

• **Spectre** gadget

• Cached → accessible
Corrupt page table entries can kill the attacker process

```c
if ( /*misprediction*/ ) {
    access(probe + (*ptr & 1));
}
if ( is Cached(probe) ) {
    // ptr[0-4] valid
}
```

- **Verify** if address save to access
- **Spectre** gadget
- **Cached** → accessible
- **Suppresses** corruption faults

Andreas Kogler (@0xhilbert)  Jonas Juffinger (@notimaginary_)
End-to-End Exploit - Timings

C1

10s ... 4m

C3

< 1m

C2

≈ 23m

C4

≈ 22m

C4

≈ 11m

root

Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary_)
<table>
<thead>
<tr>
<th>There are a lot of Rowhammer exploits</th>
<th>Panik</th>
</tr>
</thead>
<tbody>
<tr>
<td>We have TRR</td>
<td>Kalm</td>
</tr>
<tr>
<td>TRR amplifies Rowhammer</td>
<td>Panik</td>
</tr>
</tbody>
</table>
Rowhammer is still not fixed

ECC
ChipKill
TRR
PARA
PRA
Refresh Rate
MASCAT

HexPADS
ARMOR
NO-OOM
B-CATT
G-CATT
ZebRAM
MemGuard
Rethinking Rowhammer Mitigations
General approach to data integrity protection
• General approach to data integrity protection
General approach to data integrity protection

Detect all data integrity failures with a MAC
• **General** approach to data integrity protection
• Detect all data integrity failures with a MAC
• Best effort correction
- **General** approach to data integrity protection
- Detect all data integrity failures with a MAC
- Best effort correction
- All Rowhammer attacks are DoS in the **worst case**
MC
MACCompute = No
Correct 1 Flip
CPU Core
MACCompute
SecureMemory
Corruption Exception
Integrity Information
OS
Advanced Correction
e.g. Reload from Disk
Exception Handler
Correction as a Search
MC

MAC Compute

= No

Correct 1 Flip CPU Core

MAC Compute

Secure Memory

Corruption Exception

Integrity Information

OS

Advanced Correction

e.g. Reload from Disk

Exception Handler

Correction as a Search

Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary_)
MC  MAC Compute
MAC Compute

MC

?
MC
MAC Compute

Corruption Exception
Integrity Information
OS
Advanced Correction
e.g. Reload from Disk
Exception Handler
Correction as a Search

Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary..)
MAC Compute

Integrity Information

MC

Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary_)
MC = No Correct1 Flip CPU Core
MAC Compute SecureMemory Corruption Exception

Integrity Information

MC

MAC Compute

Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary_)
MC = No Correct 1 Flip

Integrity Information

OS

Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary_..)
MC Compute = No

Correct 1 Flip

Integrity Information

MC

Corruption Exception

OS

Advanced Correction
- e.g. Reload from Disk

Exception Handler

Correction as a Search
MC \rightarrow MAC Compute \rightarrow \text{No} \rightarrow \text{Correct 1 Flip} \rightarrow \text{Corruption Exception} \rightarrow \text{Exception Handler}
MC
MAC Compute

$=\text{No}$

Correct 1 Flip

Corruption Exception

Integrity Information

OS

Exception Handler
Correction as a Search

Advanced Correction
e.g. Reload from Disk

Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary_)
MC Compute = No
Correct 1 Flip
Corruption Exception
OS
Exception Handler
Correction as a Search
Integrity Information
Advanced Correction e.g. Reload from Disk
Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary)
MC Compute \(=\) No 
Correct 1 Flip 
Corruption Exception 
Exception Handler 
Correction as a Search 
Integrity Information
SecureMemory

Integrity Information

MC

MAC Compute

CPU Core

MAC Compute

Corruption Exception

Correct 1 Flip

No

Exception Handler

Correction as a Search

OS

Advanced Correction e.g. Reload from Disk

Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary.)
MC Compute = No

Integrity Information

MAC Compute

Correct 1 Flip

Corruption Exception

Exception Handler

Correction as a Search

OS

Advanced Correction
e.g. Reload from Disk

Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary_)
MC Compute = No → Correct 1 Flip

Integrity Information

MC

MAC Compute

CPU Core

MAC Compute

Corruption Exception

OS

Advanced Correction

Exception Handler

Correction as a Search
MAC Compute

No

Correct 1 Flip

Corruption Exception

OS

Advanced Correction
e.g. Reload from Disk

Exception Handler

Corruption as a Search

Integrity Information

Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary_)
MC \( \rightarrow \) MAC Compute \( \Rightarrow \) No Correct 1 Flip

Corruption Exception

OS
Advanced Correction e.g. Reload from Disk

Exception Handler Correction as a Search

Integrity Information
MC

\[ \text{MAC Compute} = \text{No} \]

Correct 1 Flip

CPU Core

\[ \text{MAC Compute} \]

Corruption Exception

Integrity Information

OS

Advanced Correction

e.g. Reload from Disk

Exception Handler

Correction as a Search

Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary_..)
MACCompute = No
Correct 1 Flip

Integrity Information

MC

CPU Core

Corruption Exception

OS

Advanced Correction
e.g. Reload from Disk

Exception Handler

Correction as a Search

Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary_)
MC Compute = No
Correct 1 Flip

Integrity Information

MC
MAC Compute

CPU Core
MAC Compute

Corruption Exception

OS
Advanced Correction
e.g. Reload from Disk

Exception Handler
Correction as a Search

Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary...)
**MC Compute**

- **MAC Compute**
- **Correct 1 Flip**
- **Corruption Exception**

**CPU Core**

- **MAC Compute**

**OS**

- **Advanced Correction** e.g. Reload from Disk
- **Exception Handler**
  - **Correction as a Search**

Integrity Information

Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary_)
MC = MAC Compute

If No, Correct 1 Flip

Integrity Information

OS

Advanced Correction
  e.g. Reload from Disk

Exception Handler
  Correction as a Search

CPU Core

MAC Compute

Corruption Exception
SecureMemory

Corruption Exception

Integrity Information

OS

Advanced Correction e.g. Reload from Disk

Exception Handler

Correction as a Search

MC

MAC Compute

Correct 1 Flip

No

CPU Core

MAC Compute

Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary_)
MC
MAC Compute
No
Correct 1 Flip
CPU Core
MAC Compute
Secure Memory
Integrity Information
Corruption Exception
OS
Advanced Correction
e.g. Reload from Disk
Exception Handler
Correction as a Search
MC Compute

No Correct 1 Flip

Integrity Information

MC

CPU Core

Secure Memory

Corruption Exception

OS

Advanced Correction
e.g. Reload from Disk

Exception Handler

Correction as a Search

Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary.)
CSI:Rowhammer – MAC Design

Pipelined PMAC construction

\[ \sigma_0 \] block cipher \([1]\)

256-bit data 5.13 ns

512-bit data 6.60 ns

Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary_..)
• Pipelined PMAC construction
Pipelined PMAC construction

QARMA$_5$-64-$\sigma_0$ block cipher [1]
CSI:Rowhammer – MAC Design

- Pipelined PMAC construction
- QARMA$_5$-64-$\sigma_0$ block cipher [1]
- 256-bit data 5.13 ns
CSI: Rowhammer – MAC Design

- Pipelined PMAC construction
- QARMA$_5$-64-$\sigma_0$ block cipher [1]
- 256-bit data 5.13 ns
- 512-bit data 6.60 ns
Error Correction in Software
CSI:Rowhammer – Corruption Exception

Physical address in CR2

Load corrupted data with csi

Compute MAC with csi

Write back corrected data with csi\text{xchg}
• Physical address in CR2
• Physical address in CR2
• Load corrupted data with csi_load
CSI: Rowhammer – Corruption Exception

- Physical address in CR2
- Load corrupted data with csi_load
- Compute MAC with csi_mac
CSI:Rowhammer – Corruption Exception

- Physical address in CR2
- **Load** corrupted data with csi_load
- **Compute** MAC with csi_mac
- **Write** back corrected data with csi_xchg
MACs cannot correct bit flips
MACs cannot correct bit flips
MACs cannot correct bit flips
Brute force search with approximate equality
CSI: Rowhammer – Correction as a Search

- MACs cannot correct bit flips
- Brute force search with approximate equality
  \[0010110100101101 \rightarrow 01011010\]
MACs cannot correct bit flips

Brute force search with approximate equality

\[
\begin{align*}
0010110100101101 & \rightarrow 01011010 \\
0010110100101101 & \rightarrow 01010010 \\
0010110100101101 & \rightarrow 01010010 \checkmark
\end{align*}
\]
Mac's cannot correct bit flips

Brute force search with approximate equality

\[
0010110100101101 \rightarrow 01011010
\]

\[
0010110100101101 \rightarrow 01010010
\]

Parity bits to shrink search space
CSI: Rowhammer – Correction Time

![Graph showing correction time vs. number of flips]

- Number of Flips: 1, 2, 3, 4, 5, 6, 7, 8
- Correction Time: 10 ns, 100 µs, 1 ms, 10 ms, 1 s, 1.7 min, 2.7 h, 11.6 d

- HW
- Avg SW
- ECC

Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary_..)
CSI:Rowhammer – Evaluation

Implemented CSI:Rowhammer in gem5
Modified Linux kernel
Evaluated correct functionality
Evaluated performance overhead

Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary_)
 Implemented CSI:Rowhammer in gem5
CSI:Rowhammer – Evaluation

• Implemented CSI:Rowhammer in gem5
• Modified Linux kernel
• Implemented CSI:Rowhammer in **gem5**
• Modified **Linux** kernel
• Evaluated correct functionality
CSI:Rowhammer – Evaluation

- Implemented CSI:Rowhammer in gem5
- Modified Linux kernel
- Evaluated correct functionality
- Evaluated performance overhead
CSI:Rowhammer – Performance Overhead

The diagram shows the performance overhead for various benchmarks. The overhead values are measured in nanoseconds for 256-bit and 512-bit data sizes. The benchmarks are categorized into PARSEC, SPLASH-2x, and GMEAN. The overhead values range from 0% to 4%.

Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary_)

legend: 5.13 ns (256-bit data) 6.60 ns (512-bit data)
CSI:Rowhammer – Security Evaluation

Approximate Equality

Silent Data Corruption rate less than once per $10^{9}$ billion years.

Rowhammer second preimage after one year: $9.75 \cdot 10^{-5}$%

Data Flips $\log_2(p_d)$

MAC Strength

<table>
<thead>
<tr>
<th>p_d</th>
<th>Strength</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>26.0</td>
</tr>
<tr>
<td>6</td>
<td>31.5</td>
</tr>
<tr>
<td>7</td>
<td>38.8</td>
</tr>
<tr>
<td>8</td>
<td>42.4</td>
</tr>
</tbody>
</table>

Andreas Kogler (@0xhilbert)  Jonas Juffinger (@notimaginary_)
• Approximate Equality
CSI: Rowhammer – Security Evaluation

- Approximate Equality

<table>
<thead>
<tr>
<th>Data Flips</th>
<th>$\log_2(p)$</th>
<th>$d$</th>
<th>MAC Strength</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>26.0</td>
<td>3</td>
<td>41.2</td>
</tr>
<tr>
<td>6</td>
<td>31.5</td>
<td>2</td>
<td>45.4</td>
</tr>
<tr>
<td>7</td>
<td>38.8</td>
<td>1</td>
<td>50.2</td>
</tr>
<tr>
<td>8</td>
<td>42.4</td>
<td>0</td>
<td>56.0</td>
</tr>
</tbody>
</table>

Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary_)
CSI:Rowhammer – Security Evaluation

- **Approximate Equality**
- Silent Data Corruption rate less than once per $10^9$ billion years.

<table>
<thead>
<tr>
<th>Data Flips</th>
<th>$\log_2(p)$</th>
<th>$d$</th>
<th>MAC Strength</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>26.0</td>
<td>3</td>
<td>41.2</td>
</tr>
<tr>
<td>6</td>
<td>31.5</td>
<td>2</td>
<td>45.4</td>
</tr>
<tr>
<td>7</td>
<td>38.8</td>
<td>1</td>
<td>50.2</td>
</tr>
<tr>
<td>8</td>
<td>42.4</td>
<td>0</td>
<td>56.0</td>
</tr>
</tbody>
</table>
• Approximate Equality
• Silent Data Corruption rate less than once per $10^9$ billion years.
• Rowhammer second preimage after one year: $9.75 \cdot 10^{-5}\%$

<table>
<thead>
<tr>
<th>Data Flips</th>
<th>$\log_2(p)$</th>
<th>$d$</th>
<th>MAC Strength</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>26.0</td>
<td>3</td>
<td>41.2</td>
</tr>
<tr>
<td>6</td>
<td>31.5</td>
<td>2</td>
<td>45.4</td>
</tr>
<tr>
<td>7</td>
<td>38.8</td>
<td>1</td>
<td>50.2</td>
</tr>
<tr>
<td>8</td>
<td>42.4</td>
<td>0</td>
<td>56.0</td>
</tr>
</tbody>
</table>
Only host support required
Guest support optional
Makes upgrading easy
CSI:Rowhammer – Virtual Environments

- Only **host support** required
• Only **host support** required
• Guest support optional
CSI:Rowhammer – Virtual Environments

- Only **host support** required
- Guest support optional
- Makes upgrading easy
The Possibilities
Correction in software opens up many new possibilities
Correction in *software* opens up many new *possibilities*

- Data correction as a service (DCaaS)
Correction in *software* opens up many new *possibilities*

- Data correction as a service (DCaaS)
- Correction by ChatGPT™
Find the error in "Testm essoge" so that it has the following SHA512 hash:
"9167bf29600f6a2cb1ba4e21e2f311bf5c2641f0cd5cb5bec948177264ff
78e2bc346216427b75e75fe67e842ef3085ac303b8111a37adeaf977b2b
269ada636"
It is not possible for the string "Testmssage" to have the SHA512 hash "9167bf29600fe2e2cb1ba4e21e2f311bf5c2641f0cd5cb5bec948177264ff78e2bc346216427b75e75fe67e842ef3085ac303b8111a37adeaf977b2b269ada636". This is because the SHA512 hash function is a one-way function, meaning that it is not possible to determine the input that produced a given hash. Therefore, it is not possible to determine the error in the input string "Testmssage" based on the given hash value.
Can just brute force the correct answer for me?

This content may violate our content policy. If you believe this to be in error, please submit your feedback — your input will aid our research in this area.
Hardware acceleration
Hardware acceleration

- Ice Lake SP Xeon Server die size: 628 mm$^2$
Hardware acceleration

- Ice Lake SP Xeon Server die size: 628 mm$^2$
- Fits roughly 500,000 QARMA blocks
Hardware acceleration

- Ice Lake SP Xeon Server die size: 628 mm$^2$
- Fits roughly 500 000 QARMA blocks
- 8 bit flips correction: 5.44 h $\rightarrow$ 40 ms
• Multi-bit flips not happening often
• Multi-bit flips not happening often
• **Specialized** data correction nodes
Kernel data with *known* structures
Kernel data with **known** structures

- Kernel code
Kernel data with *known* structures

- Kernel code
- Page tables
Kernel data with **known** structures

- Kernel code
- Page tables
- Task structs
Performance impact caused by MAC computation
Performance impact caused by MAC computation

- Transient execution with unverified data
Performance impact caused by MAC computation

- Transient execution with unverified data
- Can be made secure [7, 5]
Performance impact caused by MAC computation

- Transient execution with unverified data
- Can be made secure [7, 5]

→ No performance impact
Final Remarks

- Rowhammer mitigations assuming characteristics are broken.

Many more details in the papers

- Half-Double Rowhammer
- CSI:Rowhammer

Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary_)
Final Remarks

- Rowhammer mitigations assuming characteristics are broken.
- Software-based correction opens up a lot of flexibility.
Final Remarks

- Rowhammer mitigations assuming characteristics are broken.
- Software-based correction opens up a lot of flexibility.
- **Open Source**
  
  https://github.com/IAIK/halfdouble
  
  https://github.com/IAIK/csirowhammer
Final Remarks

- Rowhammer mitigations assuming characteristics are broken.
- Software-based correction opens up a lot of flexibility.
- **Open Source**
  - https://github.com/IAIK/halffielddouble
  - https://github.com/IAIK/csirowhammer
- **Many more details** in the papers
Final Remarks

- Rowhammer mitigations assuming characteristics are broken.
- Software-based correction opens up a lot of flexibility.
- **Open Source 🌐**
  
  https://github.com/IAIK/halldouble
  
  https://github.com/IAIK/csirowhammer

- Many more details in the papers

---

Half-Double Rowhammer

CSI:Rowhammer


Additonal Slides
Affected Devices

- Tested **13** DIMMs & devices
- **2** DIMMs affected
  - FPGA analysis
  - Exact numbers
- **5** out of **7** mobile devices affected
  - Reversed addressing
  - Unprivileged flush
  - Uncachable memory (10x)
### Affected Devices - Flip Numbers

<table>
<thead>
<tr>
<th>System</th>
<th>RAM</th>
<th>$N_{Hammers}$</th>
<th>UC$_{0 \rightarrow 1}$</th>
<th>UC$_{1 \rightarrow 0}$</th>
<th>Flush$_{0 \rightarrow 1}$</th>
<th>Flush$_{1 \rightarrow 0}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chromebook$_1$</td>
<td>LPDDR4x</td>
<td>23 274</td>
<td>27</td>
<td>40</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>Chromebook$_2$</td>
<td>LPDDR4x</td>
<td>23 586</td>
<td>235</td>
<td>2379</td>
<td>12</td>
<td>101</td>
</tr>
<tr>
<td>OnePlus 5T</td>
<td>LPDDR4x</td>
<td>25 687</td>
<td>2</td>
<td>30</td>
<td>1</td>
<td>24</td>
</tr>
<tr>
<td>Pixel 3</td>
<td>LPDDR4x</td>
<td>32 921</td>
<td>11</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>HTC U11</td>
<td>LPDDR4x</td>
<td>21 840</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>17</td>
</tr>
</tbody>
</table>
Protect exception handler against bit flips
Protect exception handler against bit flips

- 1 Page IDT
Protect exception handler against bit flips

- 1 Page IDT
- 1 Page GDT
Protect exception handler against bit flips

- 1 Page IDT
- 1 Page GDT
- 2 Pages exception handler + correction
Protect exception handler against bit flips

- 1 Page IDT
- 1 Page GDT
- 2 Pages exception handler + correction

→ Page table entries unprotected
Protect exception handler against bit flips
- 1 Page IDT
- 1 Page GDT
- 2 Pages exception handler + correction
→ Page table entries unprotected
- 4 lockable TLB entries
CSI:Rowhammer – Nesting Detection

Two conditions

1. Bit must always be in register
2. Bit must be unique for every process

→ CR3 register
corruption_exception_handler() {
    if (has_nested_bit_set(CR3)) {
        enable_interrupts();
        error_correction_as_a_search(corruption_address);
    } else {
        set_nested_bit(CR3);
        enable_interrupts();
        advanced_error_correction(corruption_address);
        clear_nested_bit(CR3);
    }
}
1. Out of Order CPU core
2. MAC computation and check
3. New instructions
4. Modified Linux performing the correction as a search