

# Half-Double: Hammering From the Next Row Over

USENIX Security 2022

12th August 2022



**Andreas Kogler**

Graz University of Technology

**Yoongu Kim**

Google

**Eric Shiu**

Rivos

**Jonas Juffinger**

Graz University of Technology, Lamarr

**Moritz Lipp**

Amazon Web Services

**Mattias Nissler**

Google

**Salman Qazi**

Google

**Nicolas Boichat**

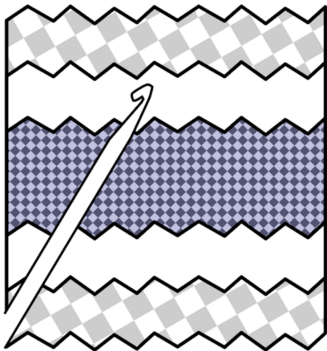
Google

**Daniel Gruss**

Graz University of Technology



- **Rowhammer**
  - Default refresh window of 64 ms
- **Error Correcing Code (ECC)**
  - **Correct** only one flip
- **Targeted Row Refresh (TRR)**
  - Refresh direct neighbours hammering rows
  - **Exhaustion** with multi-sided patterns [2, 1]
- Would perfect TRR fix Rowhammer attacks?



- Short answer: **No**
- Hammering with three rows between the aggressors
  - Causes **flips** on LPDDR4x commodity devices
  - 5 out of 7 mobile devices affected
  - With active **TRR** and on-chip **ECC**
- Is this *Distance-2* Rowhammer?
- What is the **root cause**?

Far Aggressor	$(\mathcal{F}_+)$
Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$
Far Aggressor	$(\mathcal{F}_-)$

- FPGA setup
  - Control DIMM via FPGA
  - Full control over the refreshes
  - Deactivated TRR
  - No need for data retention

Far Aggressor	$(\mathcal{F}_+)$
Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$
Far Aggressor	$(\mathcal{F}_-)$

- **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

Far Aggressor	$(\mathcal{F}_+)$
Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$
Far Aggressor	$(\mathcal{F}_-)$

- **Distance-2**
  - $(\mathcal{F}_+ \rightarrow \mathcal{F}_-)^{\infty}$
  - Distance two double-sided Rowhammer
- **First** flip after:
  - 4 000 000 hammers in 270 *ms*
  - ✗ **Not** within the refresh windows

Far Aggressor	$(\mathcal{F}_+)$
Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$
Far Aggressor	$(\mathcal{F}_-)$

- **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

- Attacker  $\rightarrow \mathcal{F}$
- TRR  $\rightarrow \mathcal{N}$

**Exploitable in the Wild?**





- Target PFN in Page Table Entry [3]
- **C1**: Allocation of Contiguous Memory
- **C2**: Alternative to Memory Templating
- **C3**: Memory Massaging
- **C4**: Bit-Flip Verification

$$\mathbf{X}_0 = b_8$$

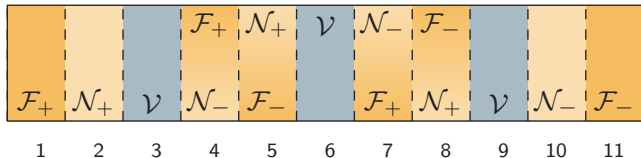
$$\mathbf{X}_1 = b_{12} \oplus b_{16}$$

$$\mathbf{X}_2 = b_{13} \oplus b_{17}$$

$$\mathbf{X}_3 = b_{14} \oplus b_{18}$$

- Mapping from virtual to physical addresses
- DRAM addressing function
- Mapping physical address to 16 DRAM banks
- **Specific** *bank* access pattern if contiguous memory
- ✓ Extract pattern with a timing side channel



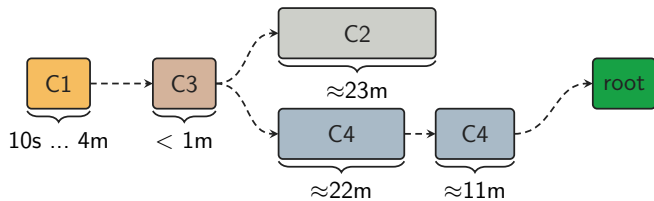


- **Skip** templating
- Spray page tables
- Hammer with Half-Double

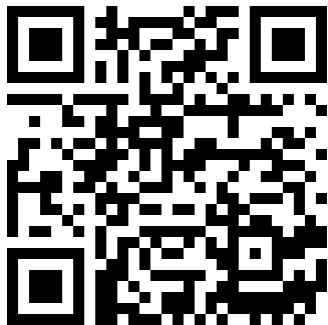
- Corrupt page table entries can **kill** the attacker process


```
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



- **45** minutes (*Chromebook<sub>2</sub>*)
- Full memory read & write primitive
- Deployable **inside** an APP



- **Open Source**  <https://github.com/IAIK/halfdouble>
- **Passed** artifact evaluation



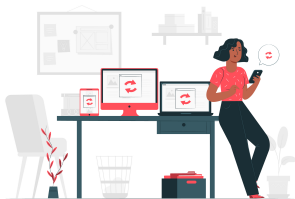
- **More details**
  - *Dance*-experiments
  - Contiguous memory solver
  - Physical memory bit recovery
  - ...

**Read the Paper**

- [1] Pietro Frigo, Emanuele Vannacci, Hasan Hassan, Victor van der Veen, Onur Mutlu, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. TRRespass: Exploiting the Many Sides of Target Row Refresh. In: S&P. 2020.
- [2] Finn de Ridder, Pietro Frigo, Emanuele Vannacci, Herbert Bos, Cristiano Giuffrida, and Kaveh Razavi. SMASH: Synchronized Many-sided Rowhammer Attacks From JavaScript. In: USENIX Security Symposium. 2021.
- [3] Mark Seaborn and Thomas Dullien. Test DRAM for bit flips caused by the rowhammer problem. Retrieved on July 27, 2015. 2015. URL: <https://github.com/google/rowhammer-test>.

**Additonal Slides**





- 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)

System	RAM	$N_{\text{Hammers}}$	$UC_{0 \rightarrow 1}$	$UC_{1 \rightarrow 0}$	$\text{Flush}_{0 \rightarrow 1}$	$\text{Flush}_{1 \rightarrow 0}$
Chromebook <sub>1</sub>	LPDDR4x	23 274	27	40	2	5
Chromebook <sub>2</sub>	LPDDR4x	23 586	235	2379	12	101
OnePlus 5T	LPDDR4x	25 687	2	30	1	24
Pixel 3	LPDDR4x	32 921	11	5	0	0
HTC U11	LPDDR4x	21 840	-	-	3	17