RowHammer

Mengjia Yan
Spring 2023
RowHammer In One Sentence

DRAM
Observation: Repeatedly accessing a row enough times can cause disturbance errors in nearby rows.
Why Care About RowHammer?

• One can predictably induce bit flips in commodity DRAM chips
• An example of how a simple hardware failure mechanism can create a widespread system security vulnerability
Outline

• Why does RowHammer happen? What is its working mechanism?

• How to perform the attack in practice? Challenges?

• Attack consequences? Mitigations?
DRAM Basics

- Each bit in DRAM is stored in a “cell” using a capacitor
- Read is destructive
- DRAM cells lose their state over time (hence Dynamic RAM)
- Data stored in DRAM cells needs to be “refreshed” at a regular interval

DRAM Basics

- Each bit in DRAM is stored in a “cell” using a capacitor
- Read is destructive
- DRAM cells lose their state over time (hence Dynamic RAM)
- Data stored in DRAM cells needs to be “refreshed” at a regular interval

Why we widely use DRAM given some of its unappealing properties?
- Speed
- Density
- Cost
  (2-10x slower than SRAM)
  (20x denser than SRAM)
  (~100x cheaper per MB)
• Bits stored in 2-dimensional arrays on chip
• Question: why read the entire row?
DRAM Refresh

- How to do refresh?
- Performance penalty of refresh
  - In an 8Gb memory, upwards of 10% of time is spent in refresh!
- The common refresh interval: 64ms
Aside: Cold Boot Attacks

<table>
<thead>
<tr>
<th></th>
<th>Seconds w/o power</th>
<th>Error % at operating temp.</th>
<th>Error % at −50°C</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDRAM (1999)</td>
<td>60 300</td>
<td>41 50</td>
<td>(no errors)</td>
</tr>
<tr>
<td>DDR (2001)</td>
<td>360 600</td>
<td>50 50</td>
<td>0.000095</td>
</tr>
<tr>
<td>DDR (2003)</td>
<td>120 360</td>
<td>41 42</td>
<td>0.00105 0.00144</td>
</tr>
<tr>
<td>DDR2 (2007)</td>
<td>40 80</td>
<td>50 50</td>
<td>0.025 0.18</td>
</tr>
</tbody>
</table>
See RowHammer Again

Observation: Repeatedly accessing a row enough times between refreshes can cause disturbance errors in nearby rows
Infrastructures to Understand Rowhammer

Kim et al; Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors; ISCA’14
Most DRAM Modules Are Vulnerable

A company

86% (37/43)

Up to $1.0 \times 10^7$ errors

B company

83% (45/54)

Up to $2.7 \times 10^6$ errors

C company

88% (28/32)

Up to $3.3 \times 10^5$ errors
Study RowHammer Characteristics

• Highly local nature of the bit-flipping capability

• The probability of bitflips are data-dependent
Study RowHammer Characteristics

• Highly local nature of the bit-flipping capability

• The probability of bitflips are data-dependent

• More advanced DRAM technologies suffer more from this disturb
Refresh + Hammering Interval Effects

Examining error rates for different refresh and hammering rates on DDR2 modules from 2011-2012

Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors
Apple’s Patch for RowHammer

  Available for: OS X Mountain Lion v10.8.5, OS X Mavericks v10.9.5

Impact: A malicious application may induce memory corruption to escalate privileges

Description: A disturbance error, also known as Rowhammer, exists with some DDR3 RAM that could have led to memory corruption. This issue was mitigated by increasing memory refresh rates.

CVE-ID

CVE-2015-3693 : Mark Seaborn and Thomas Dullien of Google, working from original research by Yoongu Kim et al (2014)

HP, Lenovo, and many other vendors released similar patches
Refresh + Hammering Interval Effects

Examining error rates for different refresh and hammering rates on DDR2 modules from 2011-2012

Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors
Density Trends

- As DRAM gets physically denser, it becomes even more vulnerable!
- Only a few thousand hammer iterations are required on modern DRAM to cause a bit-flip
Denser DRAM also can result in flips in rows which are not directly adjacent to the attacker.
Technology Scaling

• Capacitor must be large enough for reliable sensing
• The access transistor should be large enough for low leakage and high retention time
• Scaling beyond 40-35nm (2013) is challenging [ITRS, 2009]

Data from all of Facebook’s servers worldwide
Why Is RowHammer Happening?

• DRAM cells are too close to each other
  • They are not electrically isolated from each other

• Access to one cell affects the value in nearby cells
  • Due to electrical interference between the cells and wires used for accessing the cells
  • Also called cell-to-cell coupling/interference

• Example: When we activate (apply high voltage) to a row, an adjacent row gets slightly activated as well
  • Vulnerable cells in that slightly-activated row lose a little bit of charge
  • If row hammer happens enough times, charge in such cells gets drained
RowHammer Attacks in Practice

- Aggressor Row = Hammered Row

Challenges:

1. How to hammer? Need to access aggressor row enough times between refreshes.

2. Address mapping. How to find addresses map to neighboring rows?

3. How to map victim’s data to vulnerable cells?
Hammer Attempt #1: repeat accesses

```
loop:
    mov (A), %eax
    mfence
    jmp loop
```

Will this work? Why?

No. Because we will hit the cache.
Hammer Attempt #2: use clflush

Will this work? Why?

No. Because we will hit the row buffer.
Hammer Attempt #3: force row open/close

| Row | V1 | A | V2 | Row |

**loop:**

```
    mov (A), %eax
    mov (A_dummy), %ecx
    clflush (A)
    clflush (A_dummy)
    mfence
    jmp loop
```
“Single-Sided” Rowhammer

```assembly
loop:
    mov (A), %eax
    mov (A_dummy), %ecx
    clflush (A)
    clflush (A_dummy)
    mfence
    jmp loop
```
“Double-Sided” Rowhammer

Increase the stress:
Repeatedly accessing both adjacent rows significantly increases the error rate of the victim row.
Challenge #2: DRAM Addressing

Virtual Address → Address translation Using Page table → Physical Address → Map to cells?

Diagram: Cell rows and buffers.
DRAM Organization: Top-down View
DRAM Organization: Top-down View

Channel -> DIMM -> Rank -> Chip -> Bank -> Row/Column
Reverse Engineer the Mapping

- Approach #1: Physical Probe
- Approach #2: Timing Side Channel via Row Buffer
Address Mapping Examples

(a) Sandy Bridge – DDR3 [23].

(b) Ivy Bridge / Haswell – DDR3.

Pessl et al. DRAMA: Exploiting DRAM Addressing for Cross-CPU Attacks. USENIX’16
Native Client (NaCl) Sandbox Escape

- NaCl is a sandbox for running native code (C/C++)
- Runs a “safe” subset of x86, statically verifying an executable
- Use bit flips to make an instruction sequence unsafe!

Example “Safe” Code:

```
andl $~31, %eax  // Truncate address to 32 bits
                 // and mask to be 32-byte-aligned.
addq %r15, %rax  // Add %r15, the sandbox base address.
jmp *%rax        // Indirect jump.
```

Exploiting the DRAM rowhammer bug to gain kernel privileges (Seaborn and Dullien)
Native Client (NaCl) Sandbox Escape

We can flip bits to allow for (unsafe) non 32-byte-aligned jumps!

Exploited “Safe” Code:

```
andl $~31, %ecx  // Truncate address to 32 bits
                 // and mask to be 32-byte-aligned.
addq %r15, %rax  // Add %r15, the sandbox base address.
jmp *%rax        // Indirect jump.
```

Exploiting the DRAM rowhammer bug to gain kernel privileges (Seaborn and Dullien)
Kernel Privilege Escalation

What could happen if a user could gain direct write access to a page table?
Other Attacks

- Virtual machine takeover
  - Use page de-duplication to corrupt host machine
- OpenSSH attacks
  - Overwrite internal public key with attacker controlled one
- Drammer
  - Rowhammer privilege escalation on ARM
  - Utilizes determinism in page allocation to target vulnerable DRAM rows
- Rowhammer.js
  - Remote takeover of a server vulnerable to rowhammer

Without memory integrity, any software-based security mechanism is insecure!
Rowhammer Mitigations?

- Manufacturing “better” chips
- Increasing refresh rate
- Error Correcting Codes
- Targeted row refresh (TRR) - Used in DDR4!
- Retiring vulnerable cells
- Static binary analysis
- User/kernel space isolation in physical memory
Rowhammer Solutions?

- Manufacturing “better” chips
- Increasing refresh rate
- Error Correcting Codes
- Targeted row refresh (TRR) - Used in DDR4!
- Retiring vulnerable cells
- Static binary analysis
- User/kernel space isolation in physical memory

- Cost
- Performance, power
- Cost, power
- Cost, power, complexity
- Cost, power, complexity
- Security
Error Correcting Codes (ECC)

- **Basic Idea:** Store extra *redundant* bits to be used in case of a flip!
- **Naive Implementation:** Store multiple copies and compare
- **Actual Implementation:** Hamming codes

Hamming codes allow for *single-error correction, double error detection* (aka SECDED)

How about more than 2-bit flips?
Takeaways

Reliability ↔ Security Implications

Robust Physical-World Attacks on Deep Learning Visual Classification - Eykholt et al.
Next:

Paper Discussions