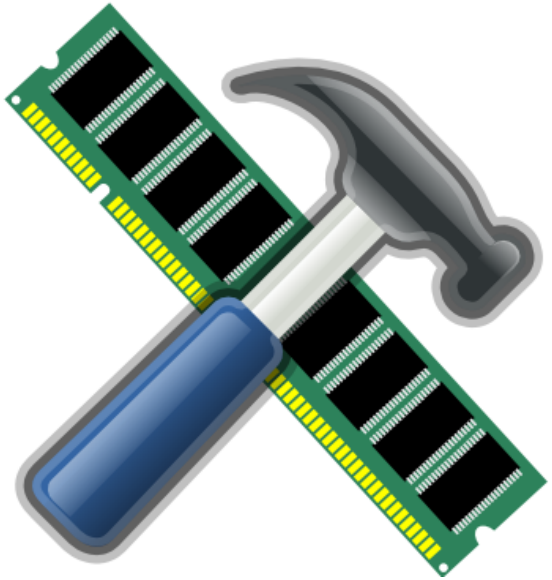
A New Approach for Rowhammer Attacks

Rui Qiao, Stony Brook University
Mark Seaborn, Google Inc.

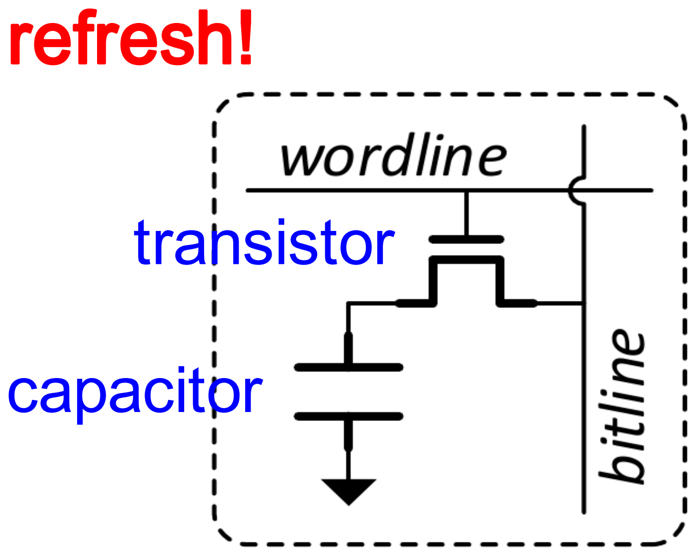
“Rowhammer”: A DRAM Bug



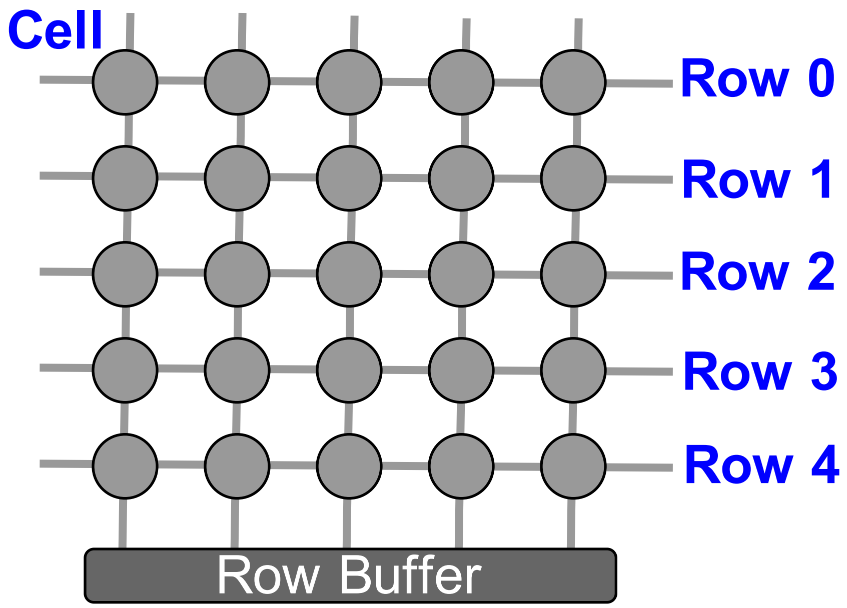
Repeated row activations (memory accesses) can cause **bit flips** in **adjacent rows**

[REF] Yoongu Kim et al, “Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors”, **ISCA 2014**

Dynamic Random-Access Memory (DRAM)

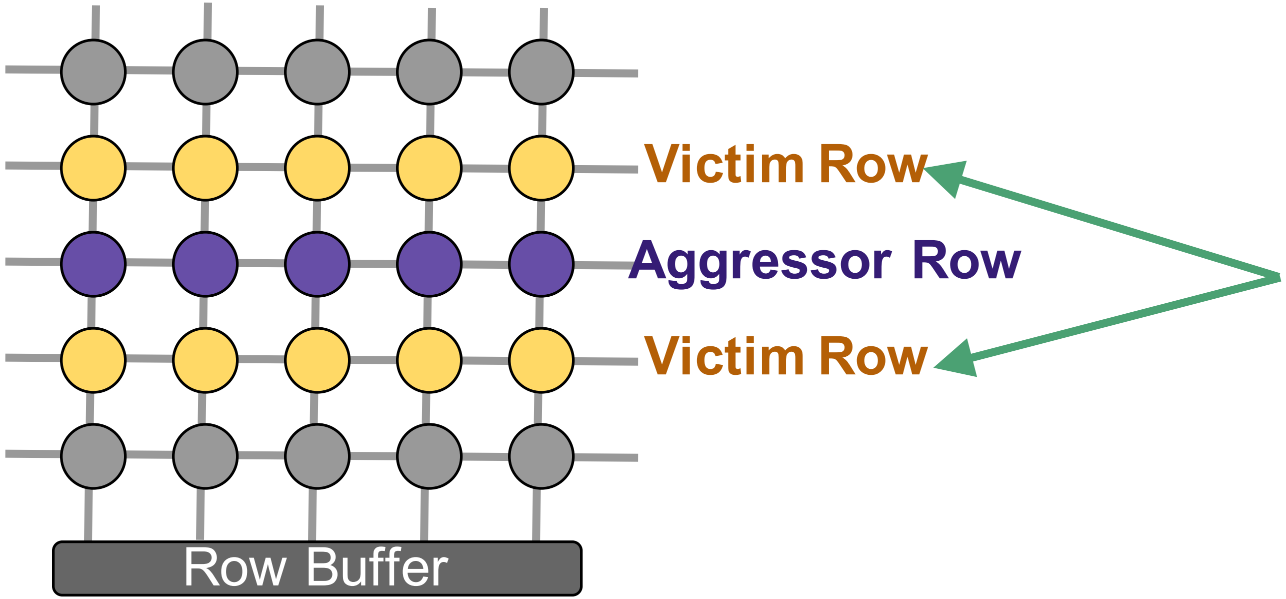


A single cell



DRAM bank

Background: Rowhammer



Bit flips in locations that are **NOT** accessed

Background: Rowhammer

- Why?
    - Electromagnetic disturbance (hypothesis)
    - Adjacent DRAM rows **discharge** at an **accelerated** rate
        - State change **before next refresh** => bit flip!
- When?
    - From 2010 onwards
        - Sub 40 nm process for DDR3
        - Larger capacity: fit larger amount of cells into the chip
            - Smaller cells, less charge
            - Closer cells, more interference

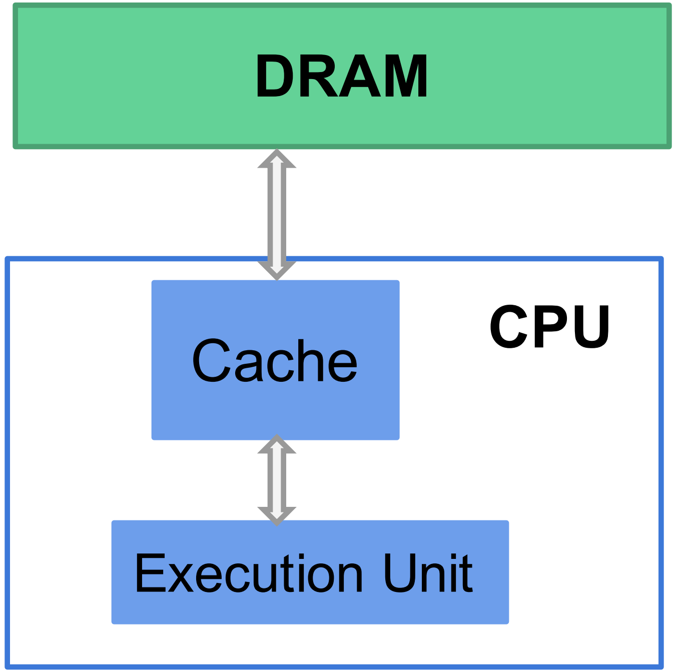
Trigger Rowhammer Programmatically

Rowhammer requires repeated **DRAM accesses (row activations)**, but **cache** can prevent DRAM accesses!

The key: bypass **cache**

CLFLUSH on x86

- Flush cache line so that DRAM *has to* be accessed



Bit Flips are Exploitable!

- Two exploits
  - Privilege escalation
  - Chrome Native Client (NaCl) sandbox escape
- NaCl background
  - Securely run (untrusted) native code on the web
- Key exploit technique
  - Single bit flip changes **read-only, control flow constraining** code

[REF] Mark Seaborn and Thomas Dullien, “Exploiting the DRAM rowhammer bug to gain kernel privileges”, BlackHat 2015

Rowhammer Defenses

- Error-Correcting Code (ECC)
  - Can mitigate rowhammer
  - Expensive: mostly used by servers
  - Limited detection/correction if there are multiple bit flips
- Target Row Refresh (TRR)
  - Idea: identify **"hot"** rows and refresh their neighbors
  - Not in DDR3/DDR4 standard
- Current mitigations: 2X refresh rate
  - Mitigates rowhammer, but no guarantee
  - Requires BIOS update: unlikely to be performed by end users
- Other proposed approaches
  - Performance counters, Probabilistic Adjacent Row Activation (PARA) ...

NaCl’s Mitigation

- Observation
  - Rowhammer requires **CLFLUSH**
- Mitigation
  - Disallow **CLFLUSH**
- Question:
  - **Other ways** to trigger rowhammer?

The Key for Rowhammer: Bypass Cache!

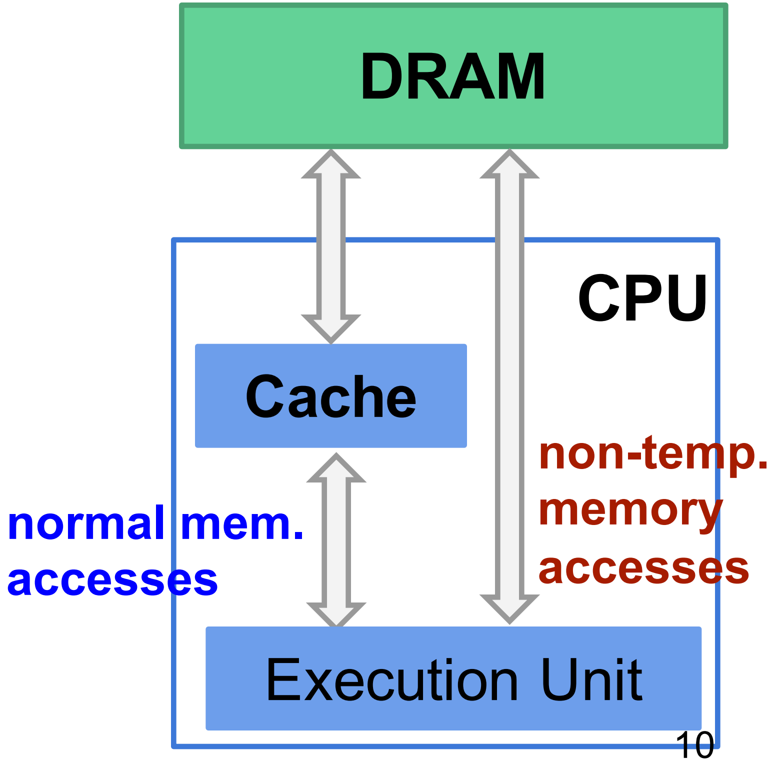
Normal memory accesses are **cached**

Non-temporal mem. accesses are **NOT cached**

- Data expected to be used only **once**
    - **no need** to bring to cache
    - avoid **cache pollution**

Non-temporal instructions (x86)

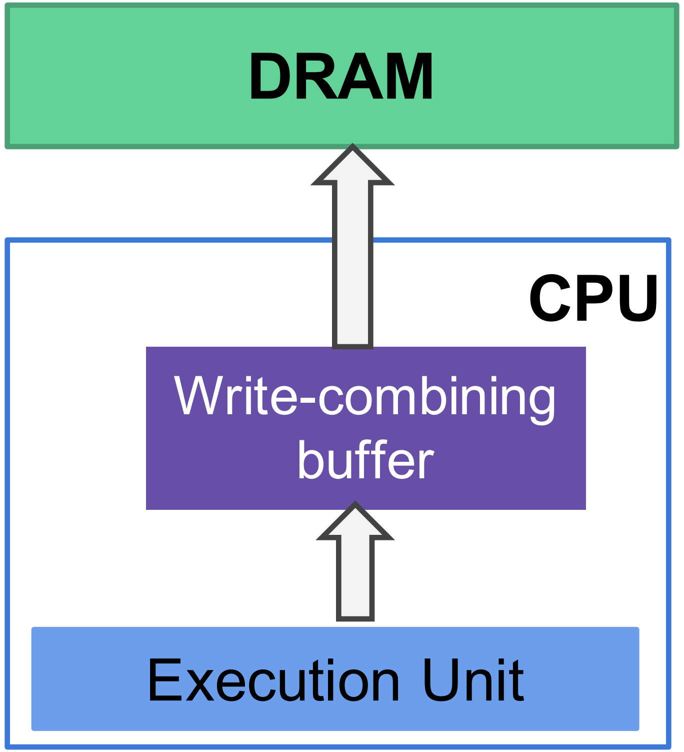
```
Examples:
    movnti %rax, (%rbx) // 8 bytes
    movntq %mm0, (%rbx) // 16 bytes
    movntdq %xmm0, (%rbx) // 32 bytes
    ...
```



Rowhammer with Non-temporal Instructions

- Non-temporal **reads** are not feasible
  - Special memory type required (memory mapped IO)
- Challenge for non-temporal **writes**
  - Write-Combining buffers
  - **Delays/Combines** WRITE

WR(**X**, **V1**), WR(**X**, **V2**), …… WR(**X**, **Vn**) => WR(**X**, **Vn**)



Rowhammer with Non-temporal Writes

Approach: **flush** write-combining buffer

```
movnti %eax, (X)
mov %eax, (X)
```

MOVNT-based Rowhammer Exploit

- Chrome Native Client sandbox escape
    - Adapted from original CLFLUSH-based exploit
- Differences
    - Bit flips by (non-temporal) memory **writes**, instead of (CLFLUSH plus) memory **reads**
    - More about the challenges and solutions in paper!

MOVNT is disallowed in latest NaCl implementation

MOVNT Instructions are Widely Used!

- Found in multimedia software, compilers, window managers, OS kernels ...
  - More often than CLFLUSH
- MOVNT is used in the C library
  - In **memset(3)** and **memcpy(3)**
  - To **reduce cache pollution**
- If rowhammer can be triggered by **only calling** memset/memcpy, almost all software can do rowhammer!

Rowhammer with memset(3) and memcpy(3)

- memset(**addr**, **value**, **size**)
- Both MOV and MOVNT-based implementations
- MOVNT is only executed when **size** > **threshold**
  - Cache pollution prevention only needed for large data copy
- Challenge:
  - We need memset(... large_size) to execute MOVNT
  - memset(... large_size) takes time, **slows down** row activations
    - Rowhammer requires 140K row activations per 64 ms

Rowhammer with memset(3) and memcpy(3)

MOVNT in libc implementations

C library	**Newlib**	**uClibc**	**Bionic (Android)**	**Glibc**	**musl**	**dietlibc**
used in memset/memcpy	Y	Y	Y	Y	N	N
MOVNT execution **threshold** size	256 bytes	120K bytes	128K bytes	~700K bytes	N/A	N/A
rowhammer-ready	**Y**	N	N	N	N/A	N/A

Security Implications

- Untrusted software
  - Only forbidding CLFLUSH is not enough
- Benign software
  - Likely to contain MOVNT
  - If malicious input can influence software to execute MOVNT in certain ways => remote rowhammer attacks

One more thing …

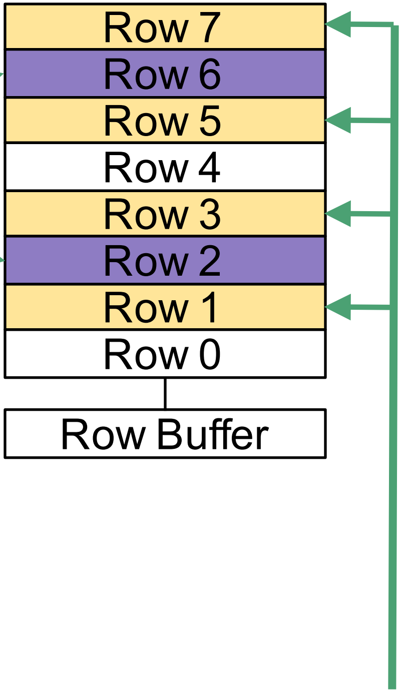
rowhammer demo
aggressor^
victims^

Thanks!

Backup slides

Original Rowhammer Approach (x86)

```
code1a:
  mov (X), %eax // Read from addr X
  mov (Y), %ebx // Read from addr Y
  clflush (X) // Flush cache for addr X
  clflush (Y) // Flush cache for addr Y
  jmp code1a
```



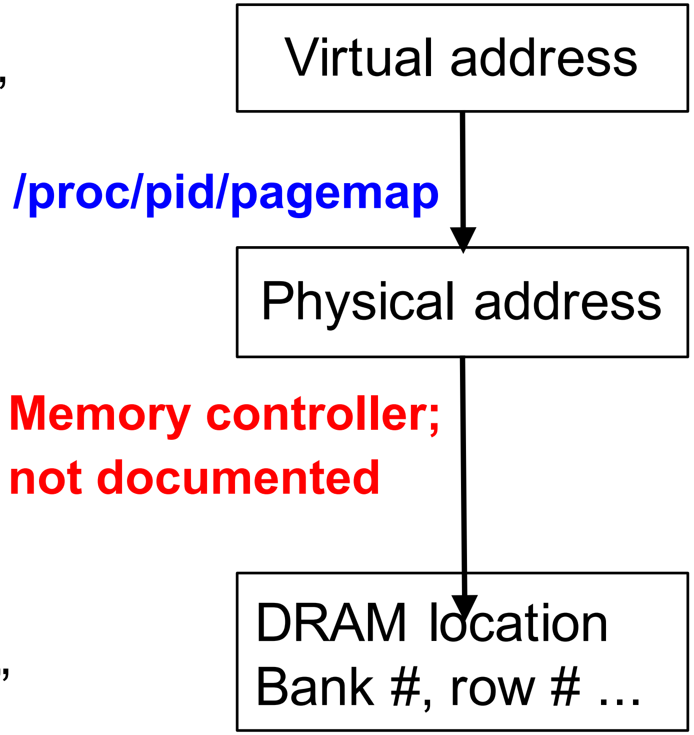
DRAM accesses

Repeated row activations can cause bit flips in **adjacent rows**

Address Selection for Rowhammer

Address selection: “same bank different rows”

Challenge: needs two mappings



Virtual address
/proc/pid/pagemap
Physical address
Memory controller; not documented
DRAM location Bank #, row # …

probabilistic approach:

Select two random virtual addresses

8 banks in total => 1/8 chance to satisfy “SBDR”

- 1/8 chance to satisfy “Same Bank”
- A bank has many (e.g., 2^15) rows => “Same Bank Same Row” negligible

Rowhammer with Non-temporal Writes

Approach: **flush** write-combining buffer

```
// This is NOT sufficient
// for rowhammer!

code2a:
   movnti %eax, (X)
   movnti %eax, (Y)
   jmp code2a
```

```
// This CAN do rowhammer!

code2b:
   movnti %eax, (X)
   mov %eax, (X)
   movnti %eax, (Y)
   mov %eax (Y)
   jmp code2b
```

Exploit: NaCl Sandbox Escape

How the exploit works

```
andl $~31, %eax // make address in %eax 32 bytes aligned
addq %r15, %rax // add base register r15; limit %rax to some address space area
jmp *%rax  => jmp *%rcx // the unconstrained register rcx is used!
```

- Control flow sandboxing escape => **arbitrary code execution**
- Insight: a single bit flip has changed **validated, read-only** code!
- 13% of bit flips are usable
- The above code is "sprayed" in the NaCl process
  - Very likely bit flips will land on it!

Getting Bit Flips: How “Hard” to “Hammer”?

- How fast?
  - At most 500 ns between two row activations
- How many times?
  - At least 139K row activations
- Refresh rate
  - DDR3 Standard: DRAM rows need to be refreshed every 64 ms