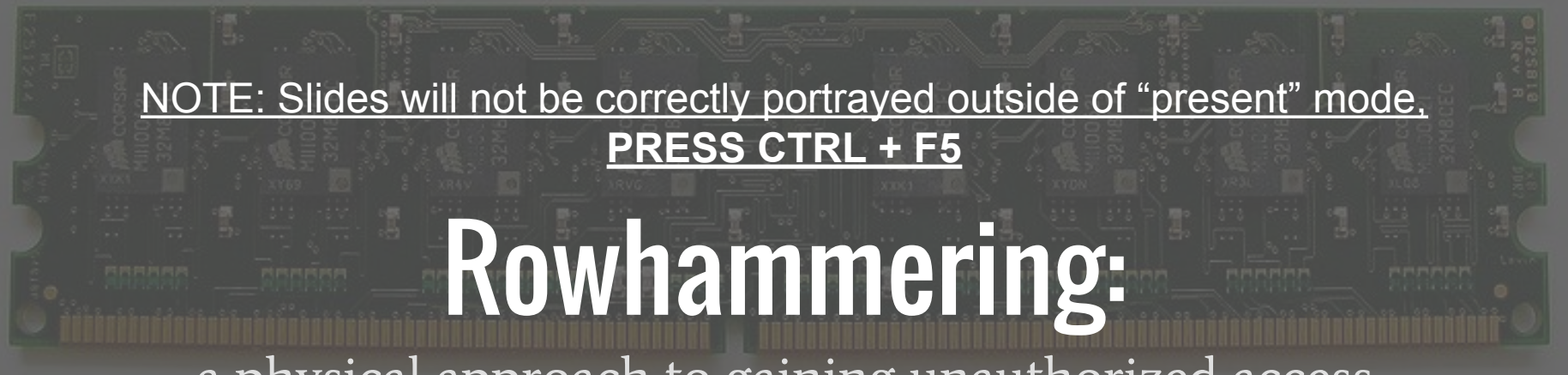


NOTE: Slides will not be correctly portrayed outside of “present” mode,
PRESS CTRL + F5

Rowhammering:

a physical approach to gaining unauthorized access

•••

Niccolas Alexander Ricci
University of Morris, Minnesota

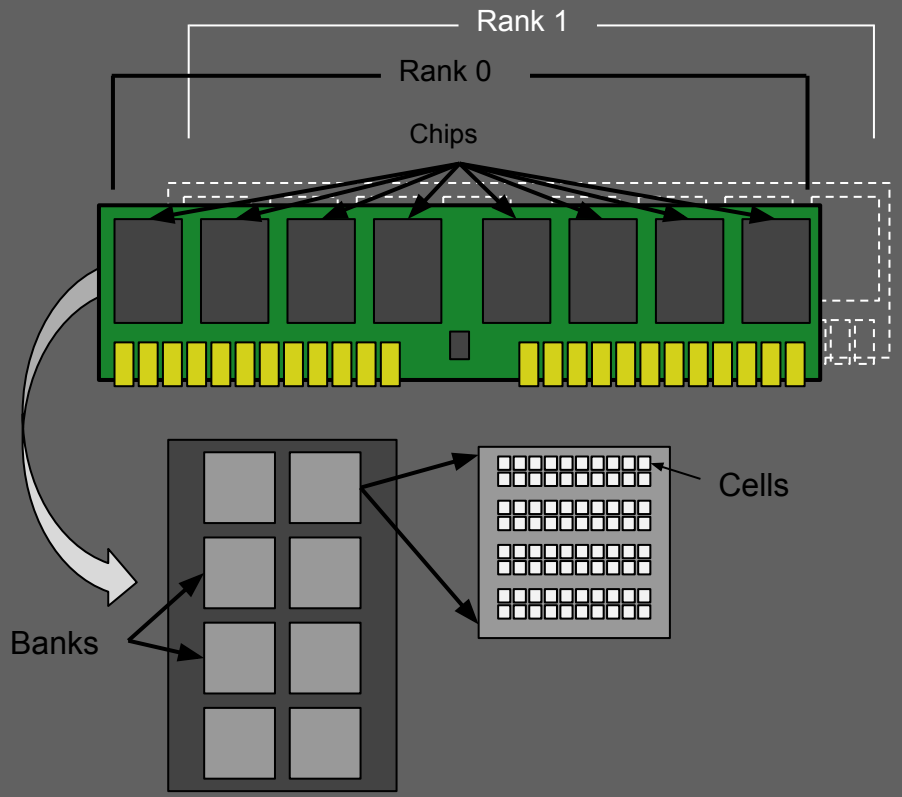
Outline

1. Introduction
    i. DRAM
    ii. Cells
    iii. Memory Controller
2. Rowhammering
    i. Google Project Zero
    ii. Rowhammer.js
3. Vulnerabilities
4. Solutions
5. Conclusion

DRAM

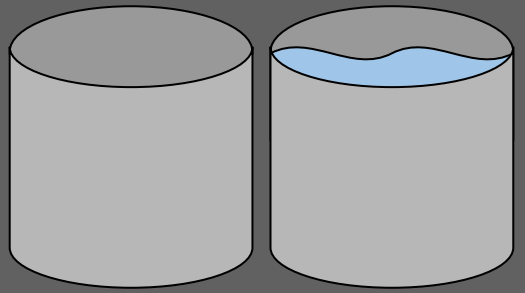
Dual Inline Memory Module (DIMM)

- Ranks
- Chips
- Banks
- Cell rows (of cells)



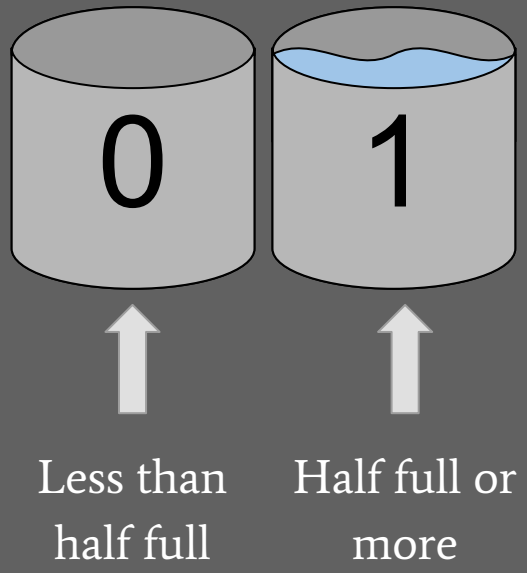
Cells

- Cells, bits, or capacitors all refer to the same thing
- A cell's charge determines its state
  - Similar to buckets full of water
  - "Fullness" determines bit-state



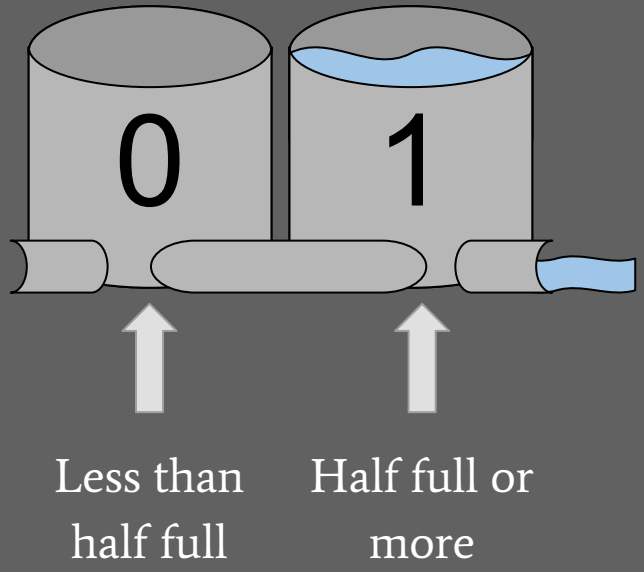
Cells

- Cells, bits, or capacitors all refer to the same thing
- A cell's charge determines its state
  - Similar to buckets full of water
  - "Fullness" determines bit-state



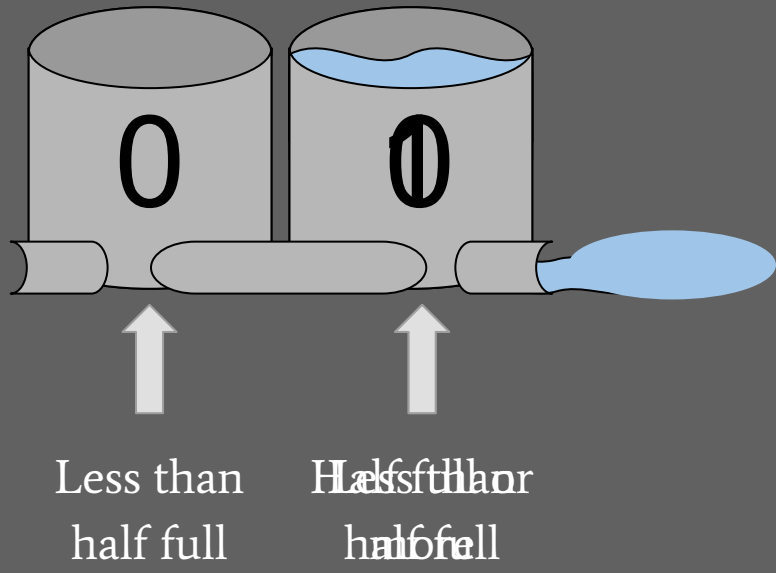
Cells

- Cells leak their charge
  - Caused by circuitry (pipes connecting buckets)



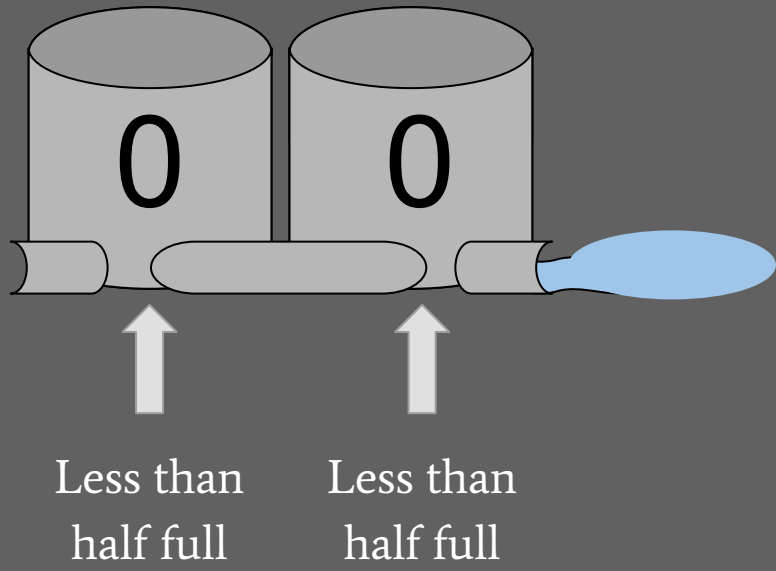
Cells

- Cells leak their charge
  - Caused by circuitry (pipes connecting buckets)
- Bit-states can change if enough charge is leaked

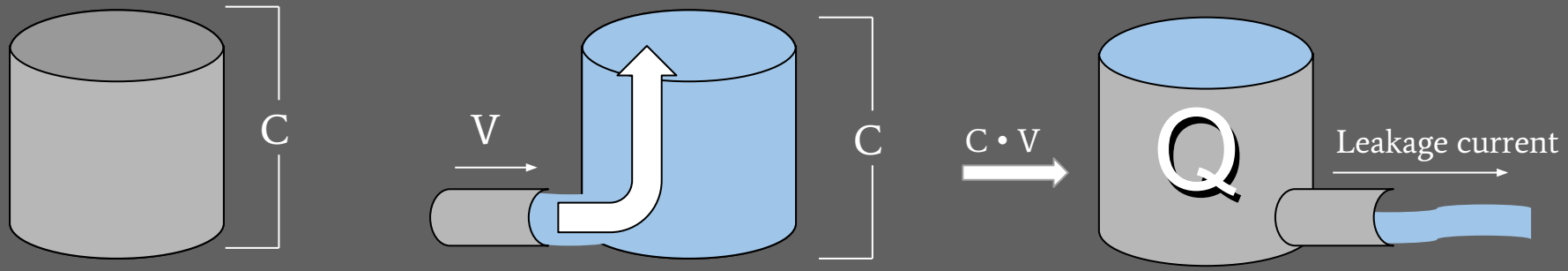


Cells

- Cells leak their charge
  - Caused by circuitry (pipes connecting buckets)
- Bit-states can change if enough charge is leaked
- Charge may also leak into adjacent cells

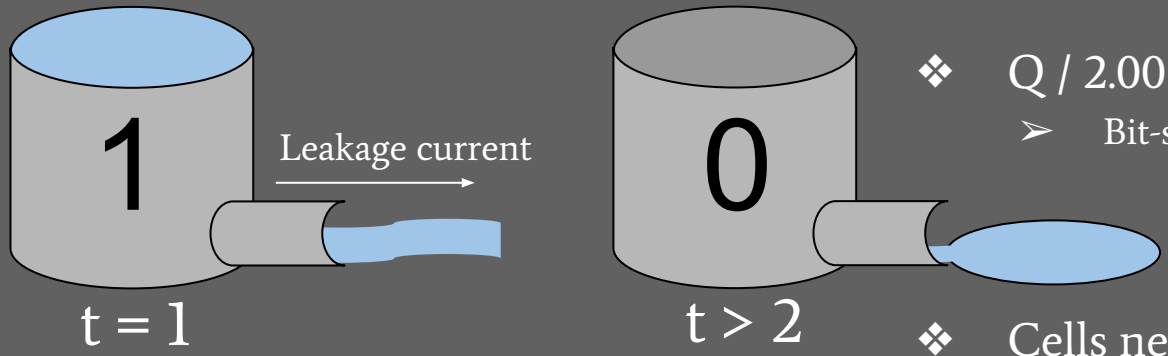


Cells



$$\frac{C \bullet V}{t} = \frac{Q}{t} = \text{Leakage current}$$

Cells

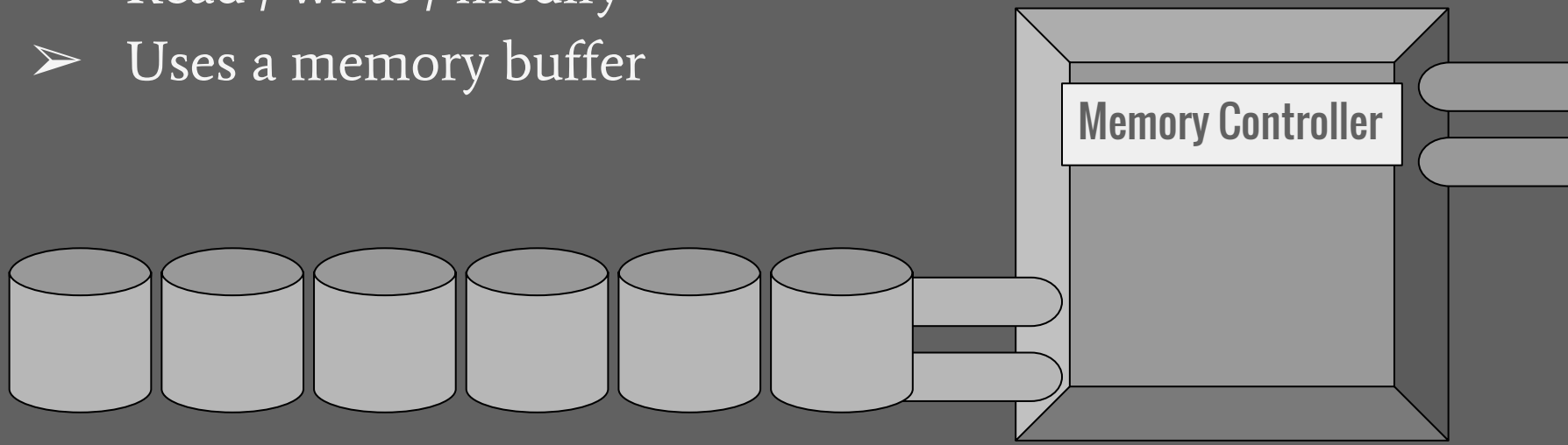


- Q / 2.001 = less than ½ capacity
  - ➢ Bit-state becomes 0
- Cells need to be refreshed before t > 2

$$\frac{C \cdot V}{t} = \frac{Q}{t} = \text{Leakage current}$$

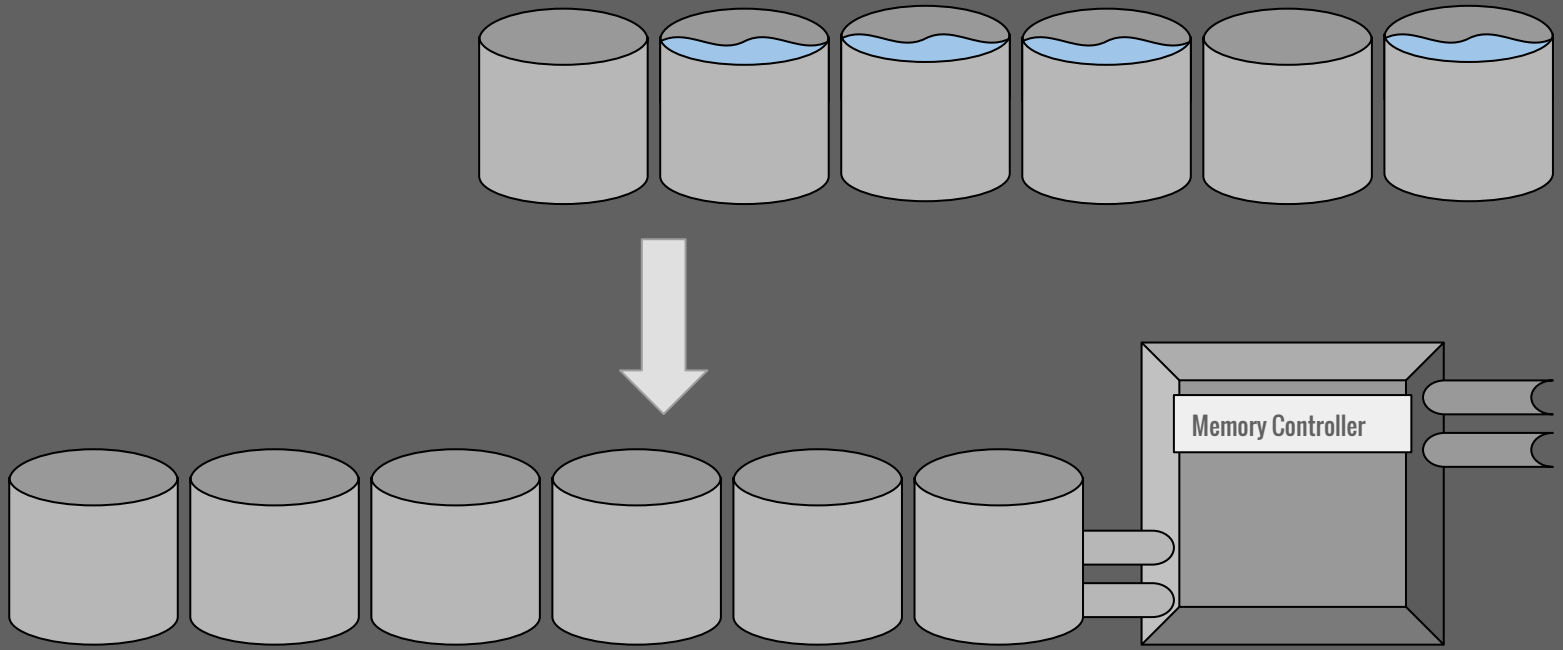
Memory Controller

- Conducts all operations in memory
  - Read / write / modify
  - Uses a memory buffer



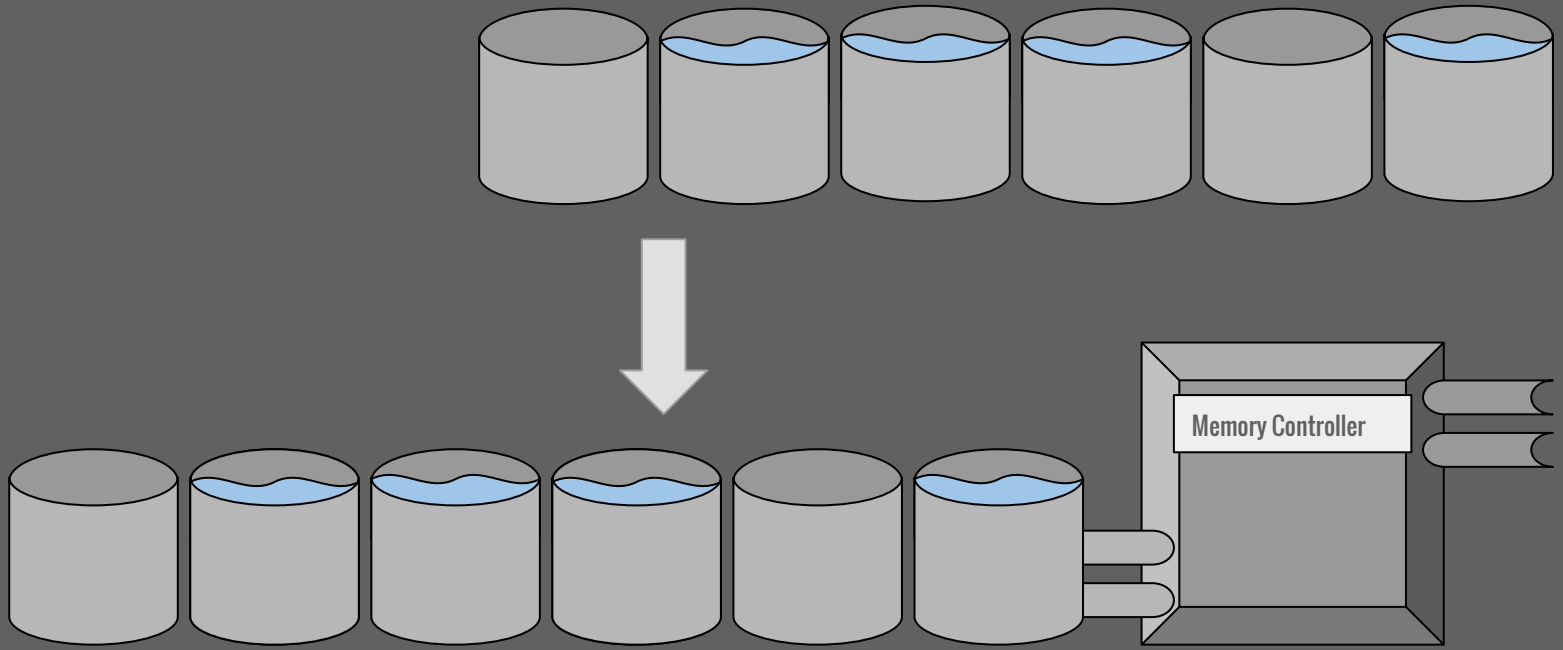
Memory Controller

- The memory controller copies a cell row into the buffer



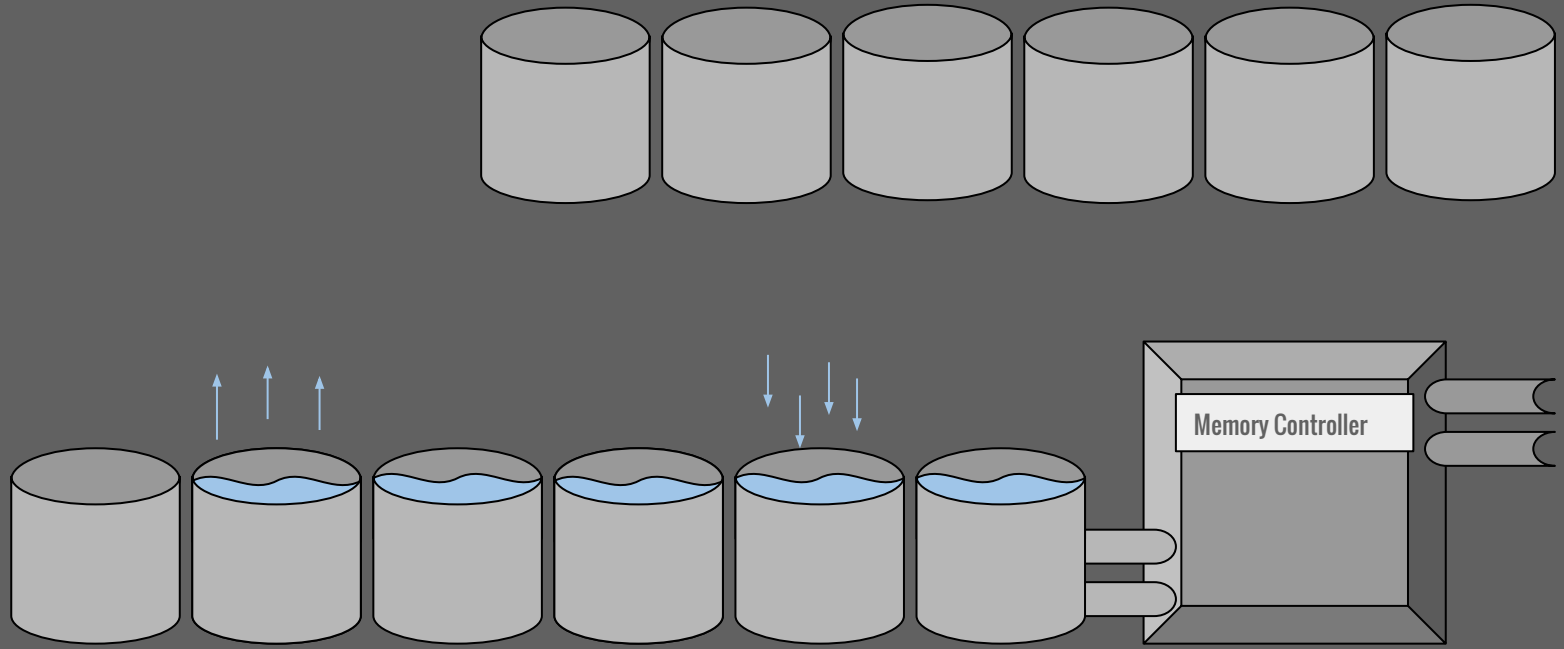
Memory Controller

- The memory controller copies a cell row into the buffer



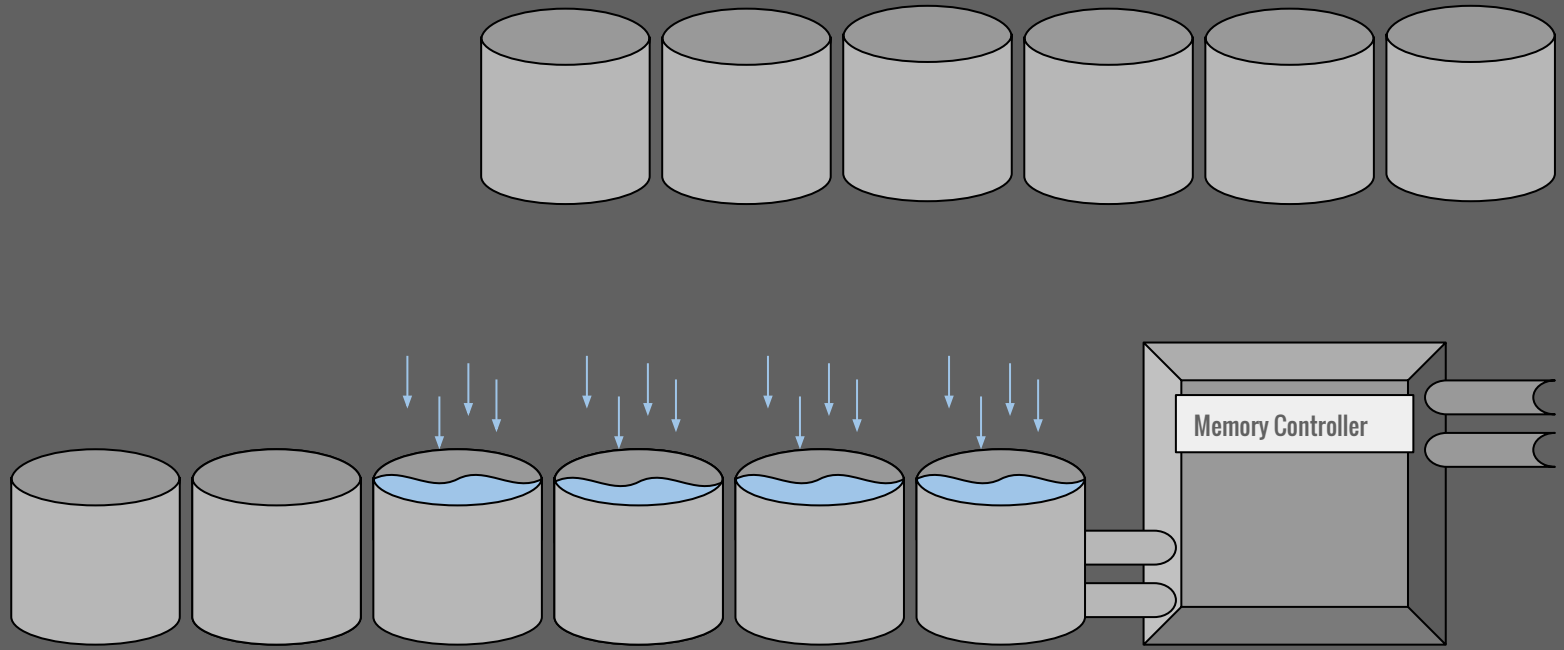
Memory Controller

- Changes (if any) are made in the buffer



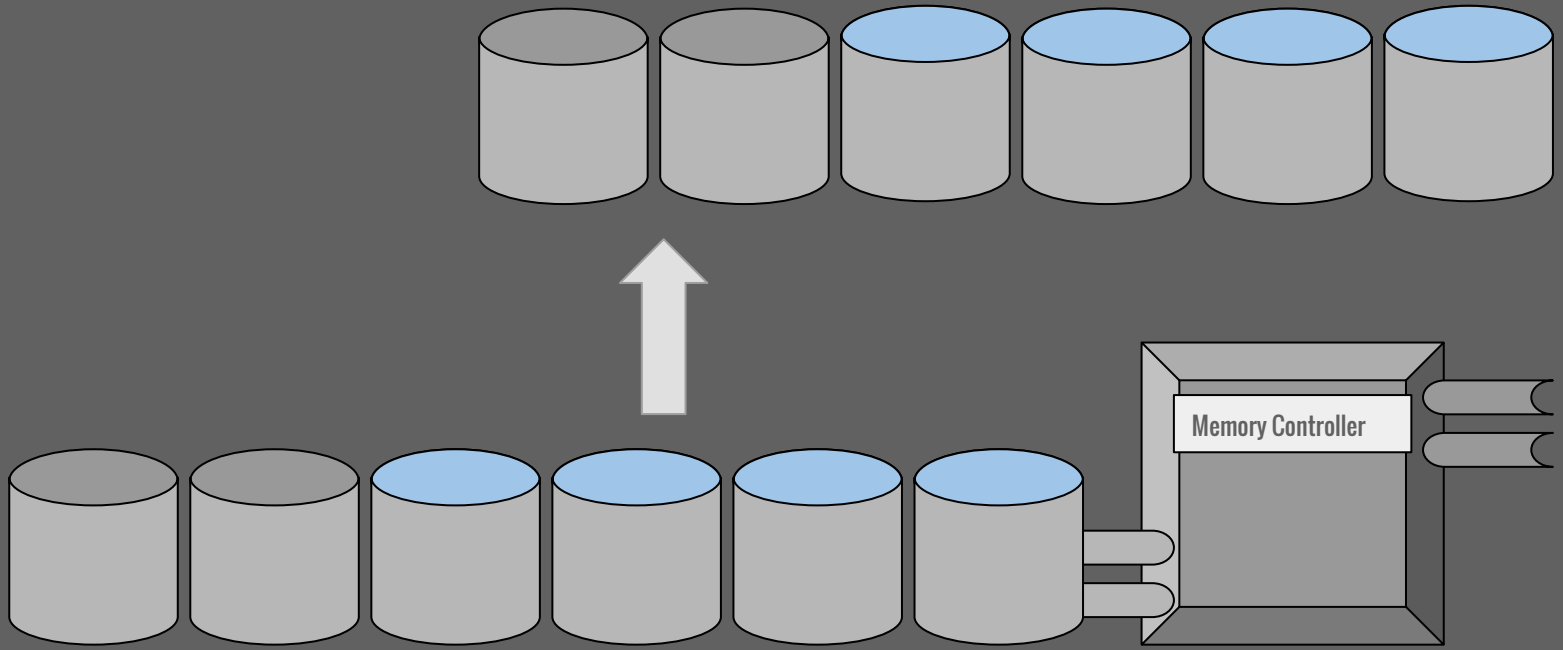
Memory Controller

- Cells are filled to capacitance



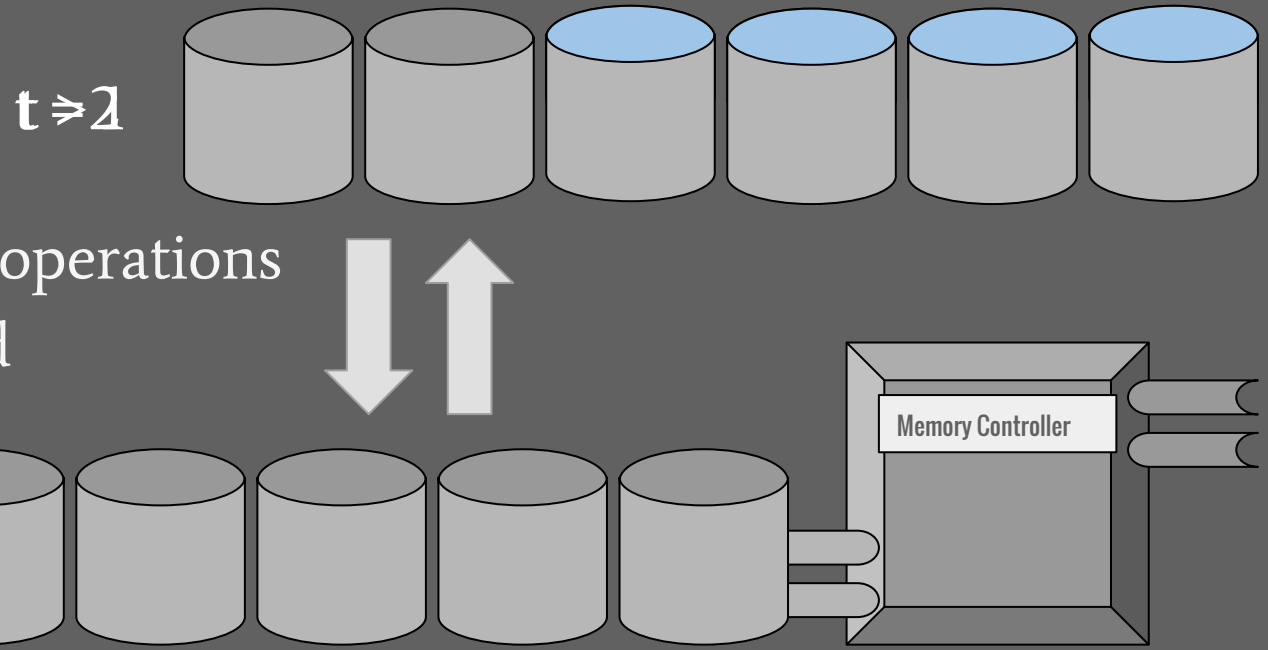
Memory Controller

- The buffer is copied back into the original cell row



Memory Controller

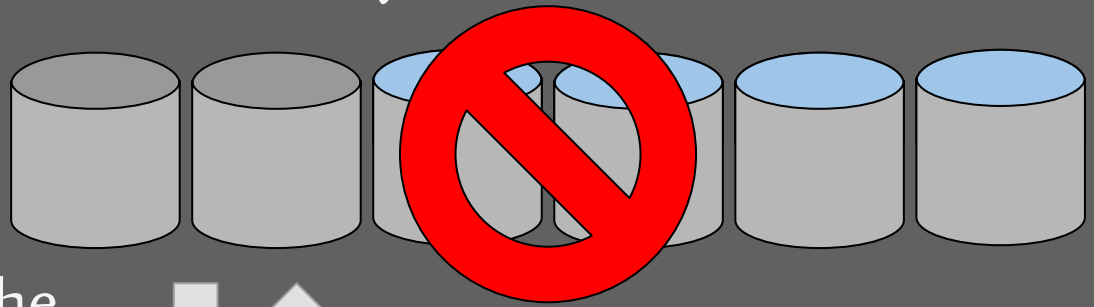
- Read/write operations don't happen frequently enough



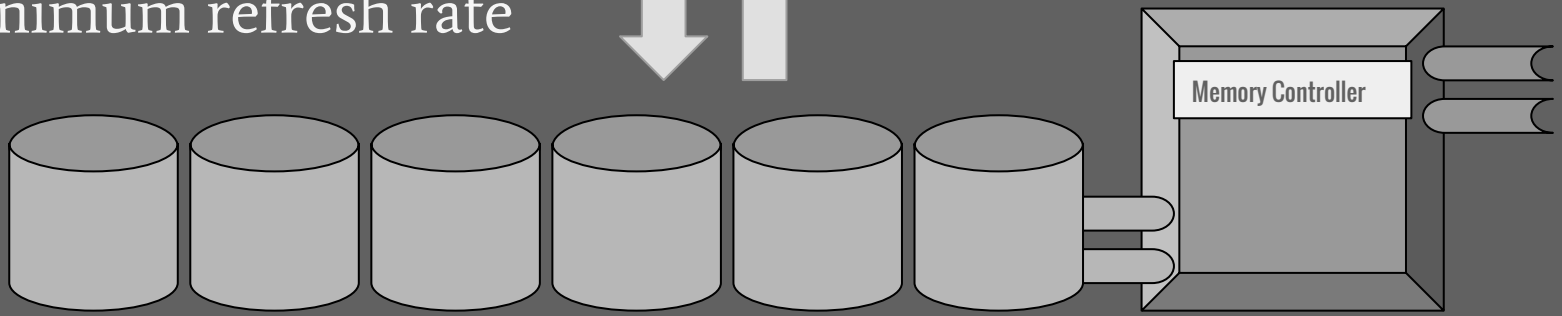
- So, "dummy" operations are performed

Memory Controller

- When cells are being refreshed, they cannot be used

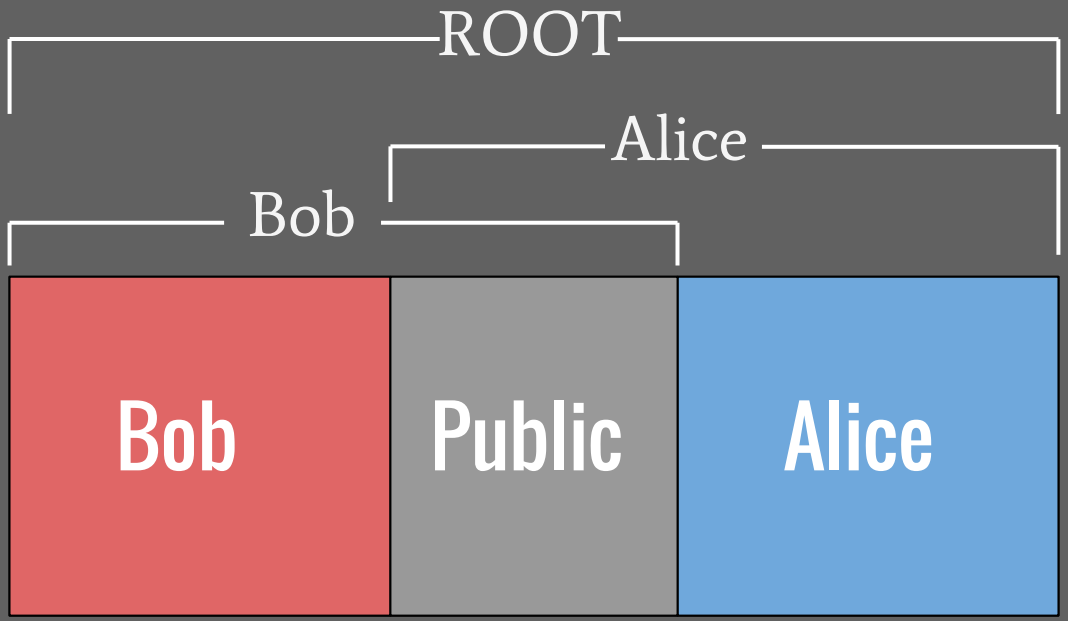


- Manufacturers use the minimum refresh rate



Privilege

- Bob and Alice have private data. ROOT has access to all data

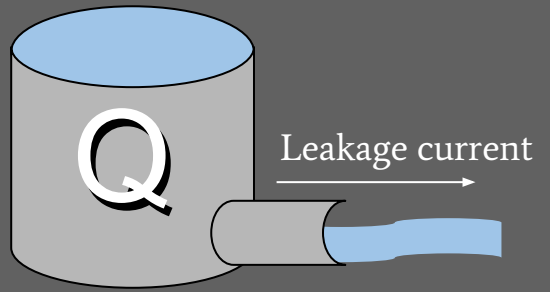


Outline

1. Introduction
2. Rowhammering
    i. Google Project Zero
    ii. Rowhammer.js
3. Vulnerabilities
4. Solutions
5. Conclusion

Rowhammering

- Rowhammering exacerbates cell leakage
- Repeatedly fills a cell row (hammering)
  - Causes charge to leak into adjacent cell rows
- Used to change the states of cells without access
- Performed in between refresh cycles
  - between t = 1, t = 2



Rowhammering

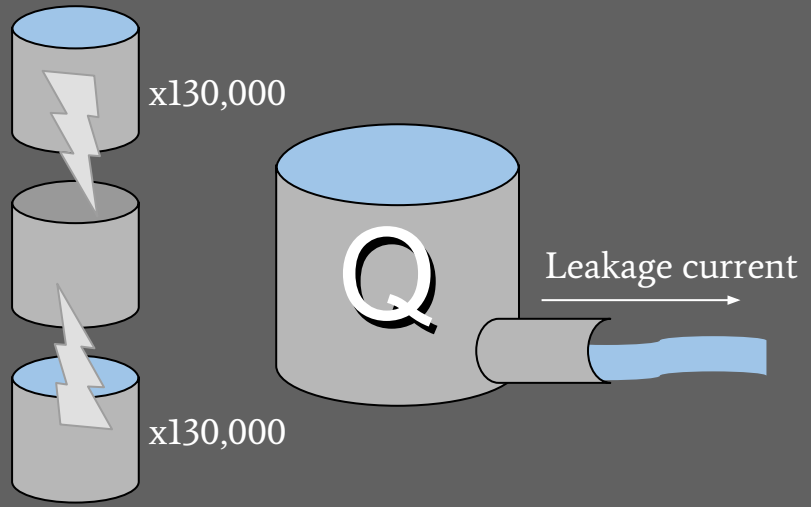
$RH = \alpha \cdot (\text{Leakage Current [LC]})$

α = increased LC under hammering

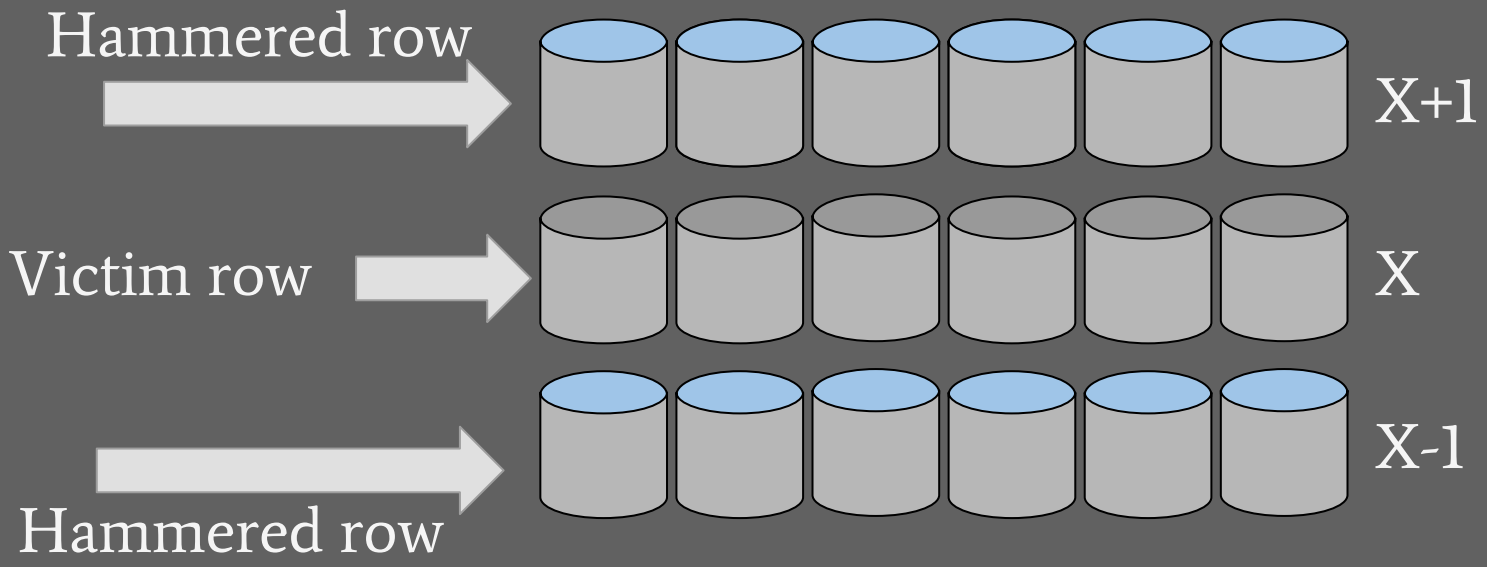
$RH_{th} = (LC) / (\alpha) \cdot (\text{Max operations})$

$RH_{th}@64ms = (LC) / (10) \cdot (1.3M)$

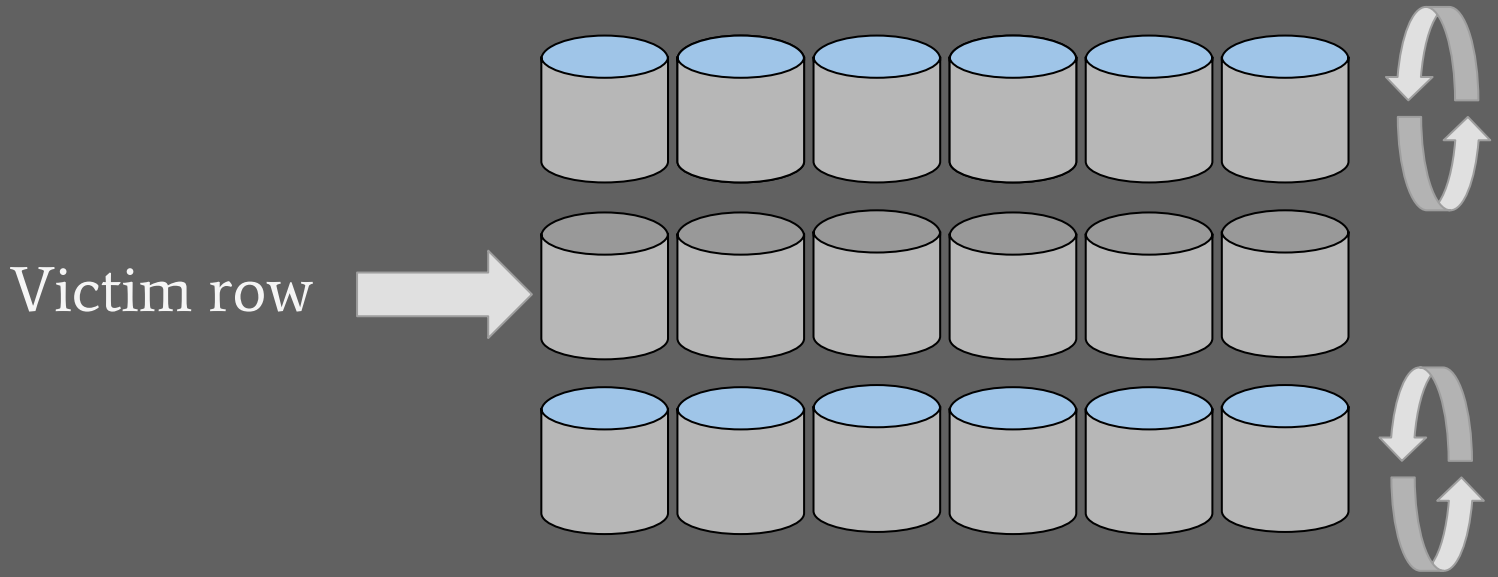
$RH_{th}@64ms = 1 / 10 \cdot 1.3M = 130{,}000$



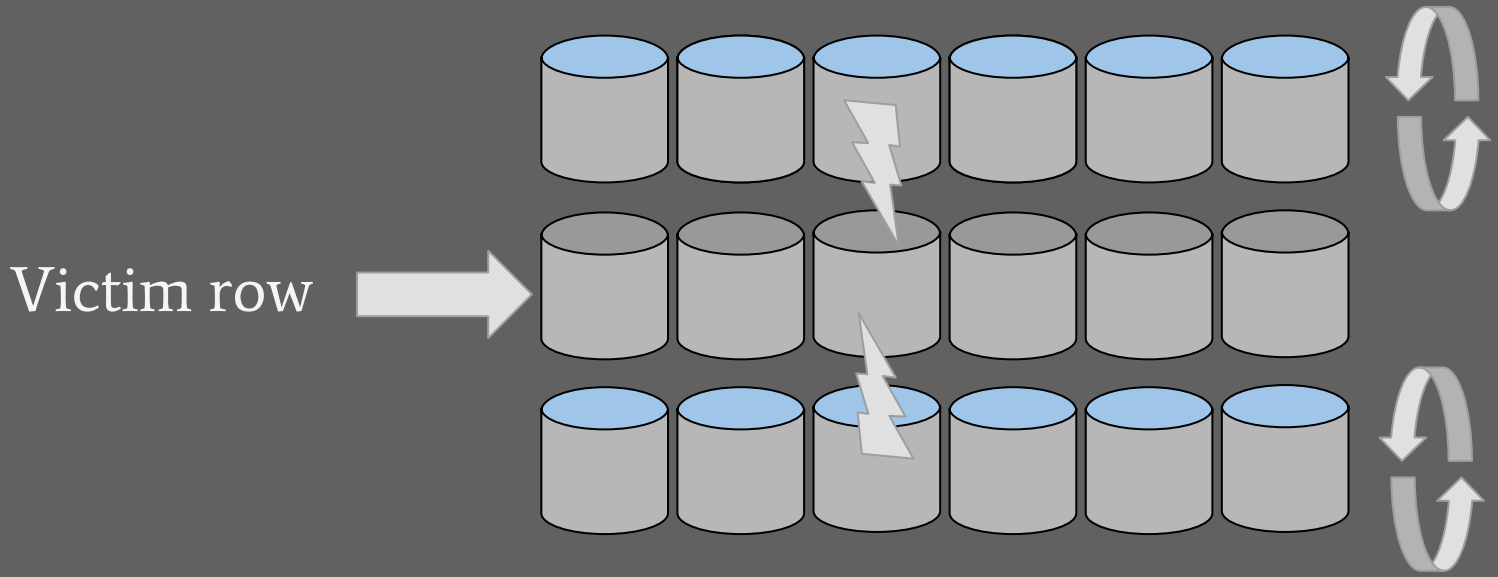
Rowhammering



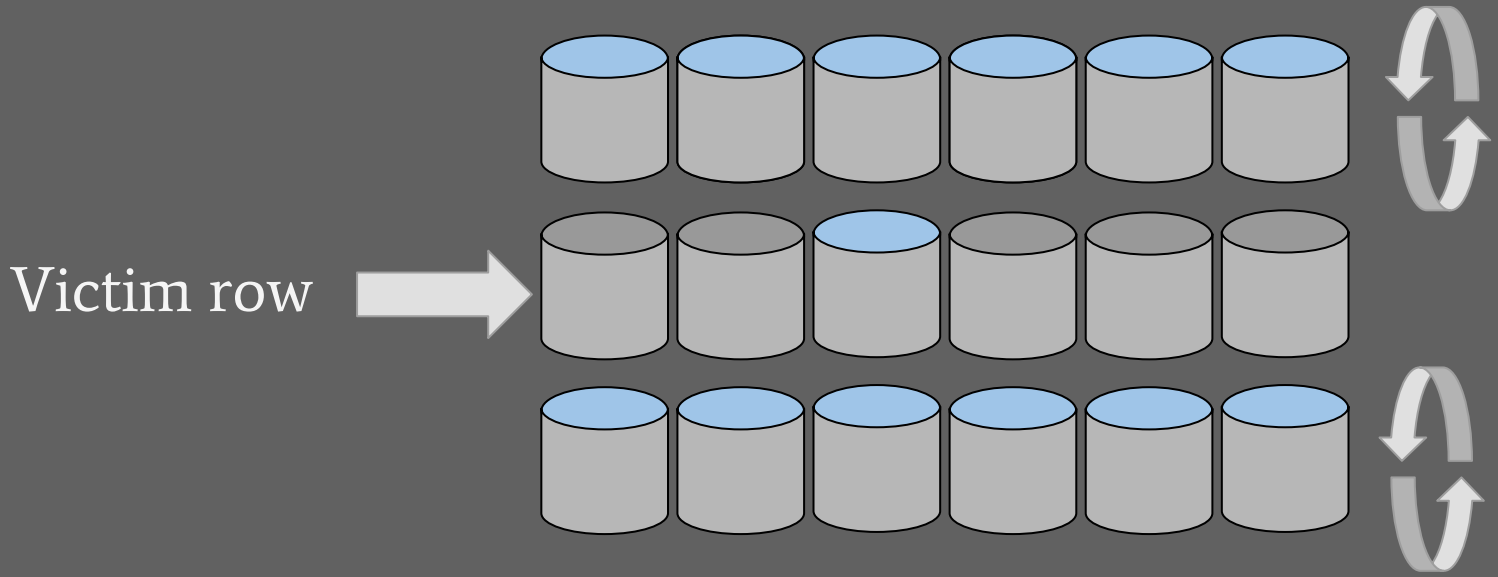
Rowhammering



Rowhammering



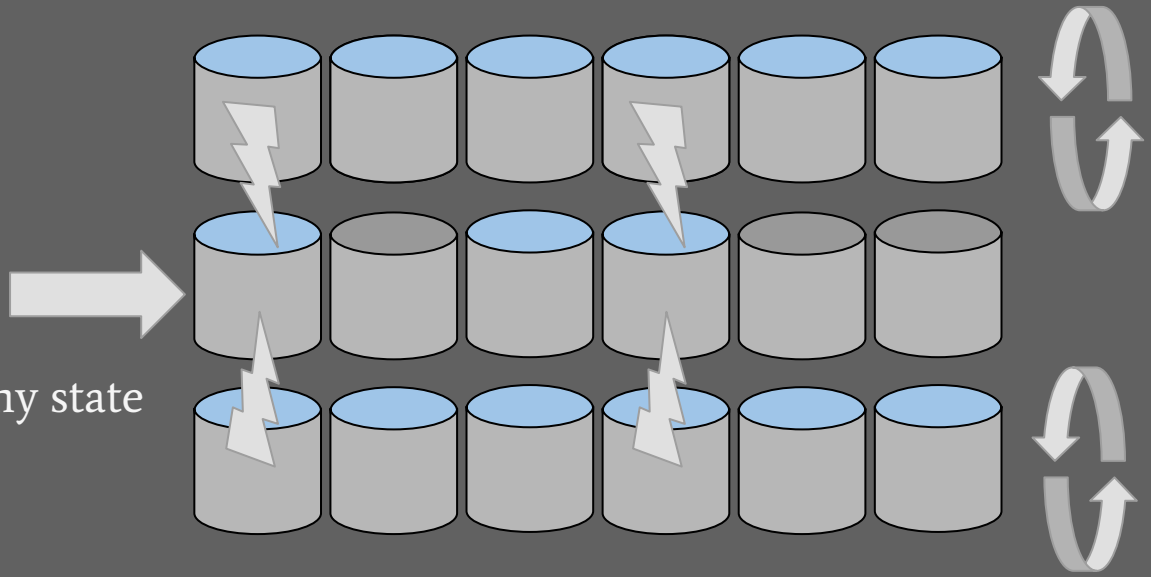
Rowhammering



Rowhammering

Victim row

Can be manipulated into any state



Rowhammering

What this means:

- Careful manipulations can provide unauthorized access

01001001 00100000 01000001 01001101
00100000 01001001 00100000 01001111
01010100 00001010

I. AM. ROOT.

Rowhammering

What this means:

- Careful manipulations can provide unauthorized access
  - Google Project Zero has done this

01001001 00100000 01000001 01001101
00100000 01001001 00100000 01010010 01001111
01010100 00001010

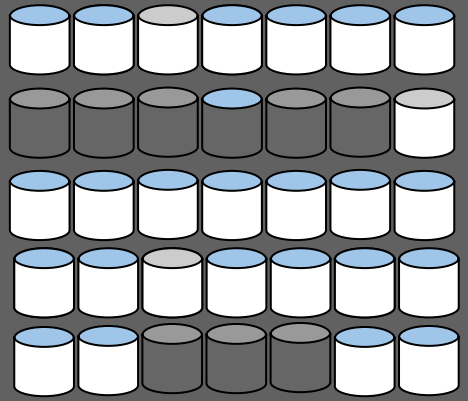
I. AM. ROOT.

Google Project Zero

- Sandbox (Chrome)
  - Allows low-level code
  - Used for efficiency
  - Deemed “safe” or “unsafe”
  - Run if safe, otherwise halt

Google Project Zero

- "Escaping" the sandbox (Chrome)
- A memory block is allocated (white)
- Fragmentation happens
- Victim rows are in between sandbox rows
- Privilege is stored somewhere
- Random rows are hammered until ROOT privilege is granted



Not sandbox =

sandbox =

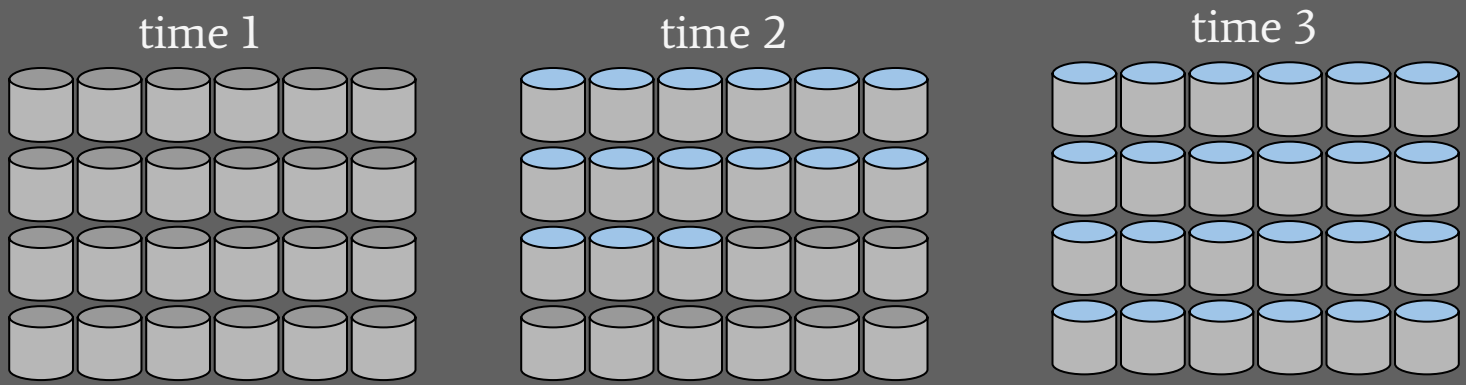
Google Project Zero

- Rowhammer test (GitHub) on Morris machines
  - GitHub test on checks for bit flips
  - Newer machines were more vulnerable (faster memory)
- Google mentions Rowhammering may be possible without the sandbox in JavaScript
- Three months later...

Computer	Iterations	Time	bit flipped?
eva01	340	2838 sec	yes
multivac	326	359 sec	yes
falcon	3979	4092 sec	yes
tang	1873	>3hours	no
zytel	13638	>3hours	no
reliant	1824	>3hours	no
neoprene	1670	>3hours	no
mylar	7541	>3hours	no
cobar	6253	>3hours	no
acrylic	5819	>3hours	no
tedlar	6124	>3hours	no
rynite	120619	>3hours	no

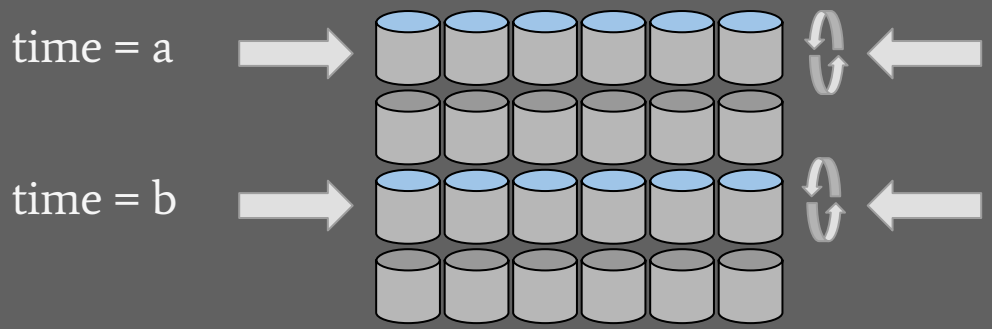
Rowhammer.js

- Rowhammering exploit in JavaScript
- Does not use low-level code
- Rowhammer.js measures computation time
- Allocates a block of memory
  - uses timing to determine offset



Rowhammer.js

- Every offset is hammered



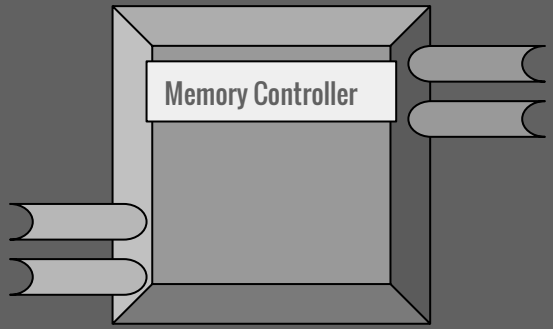
That's it,
No low-level code needed

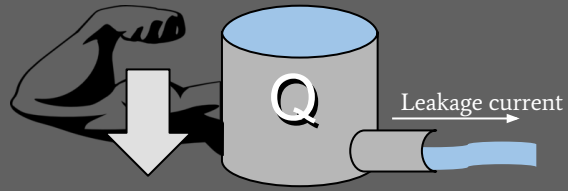
Outline

1. Introduction
2. Rowhammering
3. Vulnerabilities
4. Solutions
5. Conclusion

Vulnerabilities

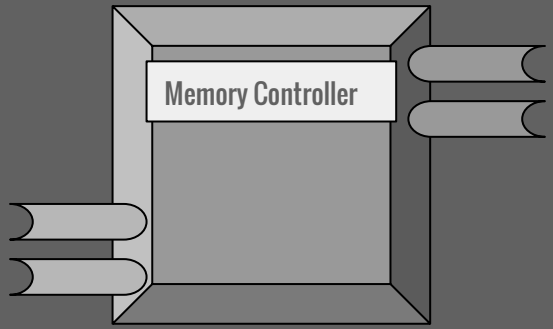
- The refresh rate plays a crucial role in Rowhammer prevention
  - The sandbox allows low level operations
    - Weak cells

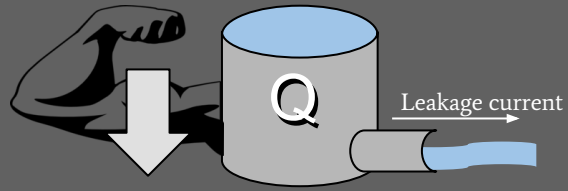




Vulnerabilities

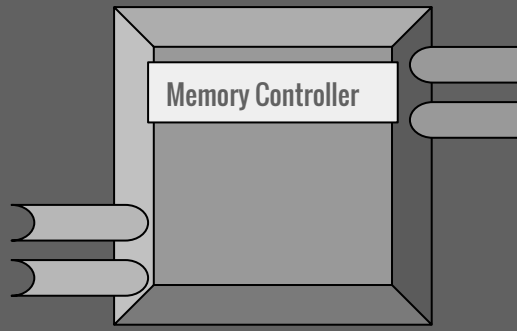
What can we do?

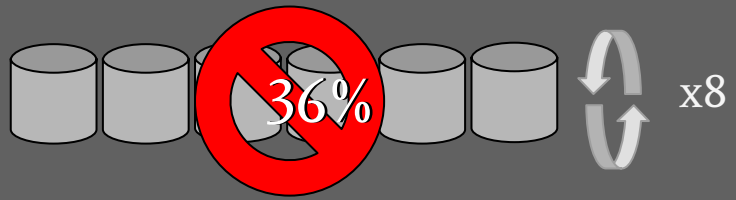




Solutions

1. Refresh [illegible] more frequently
   - i.e. redu[illegible]
   - Already b[illegible]
   - Curren[illegible] of the time
   - Woul[illegible] this: 36% of the time

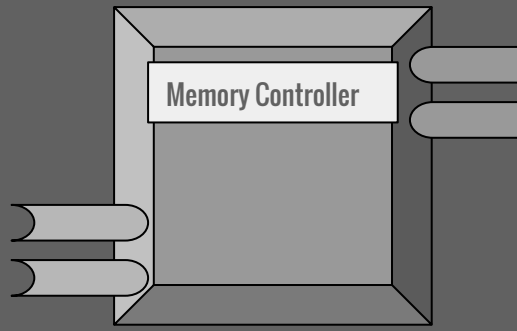


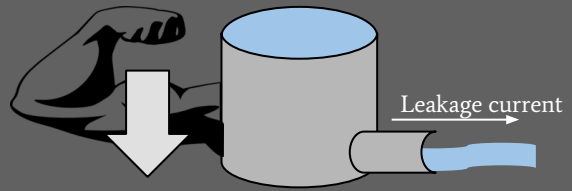


Solutions

2. Check fo victi rows

- Find wi nerable
- Relocate t
- Takes s hip
- There t b n room for all victims

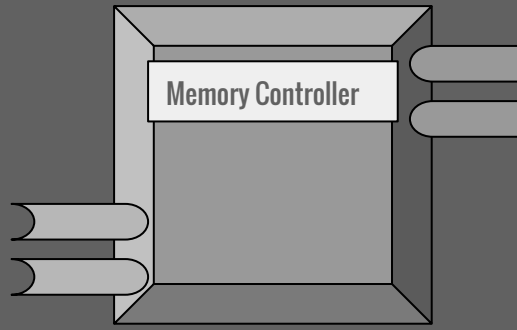




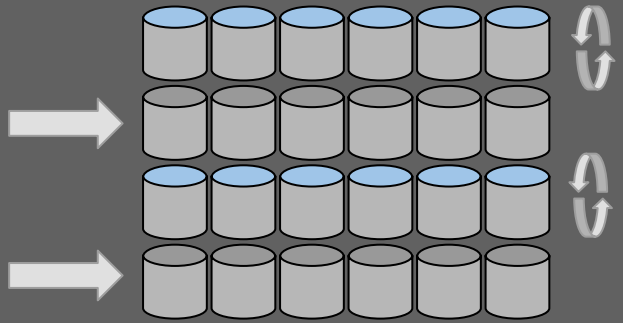
Solutions

3. Refresh intelligently

- PARA system
- Each read / write there is a slight chance to refresh adjacent rows



Victims refresh more frequently



Conclusions

- Memory will continue to get more dense
  - Which makes cells leakier
- Rowhammering is nascent
- Rowhammering can be run from JavaScript
  - JavaScript is extremely common, which means many computers to attack
- Until a solution to Rowhammering is shown to work, nothing can protect you from it

References

1. D.-H. Kim, P. Nair, and M. Qureshi. Architectural support for mitigating row hammering in dram memories. *Computer Architecture Letters*, 14(1):9–12, Jan 2015.
2. D. Gruss, C. Maurice, and S. Mangard. Rowhammer.js: a remote software-induced fault attack in javascript. *arXiv preprint arXiv:1507.06955,* 2015.
3. Google. Exploiting the DRAM rowhammer bug to gain kernel privileges, 2015.

Questions?