Design of Digital Circuits
Lecture 2: Mysteries in Comp Arch

Prof. Onur Mutlu

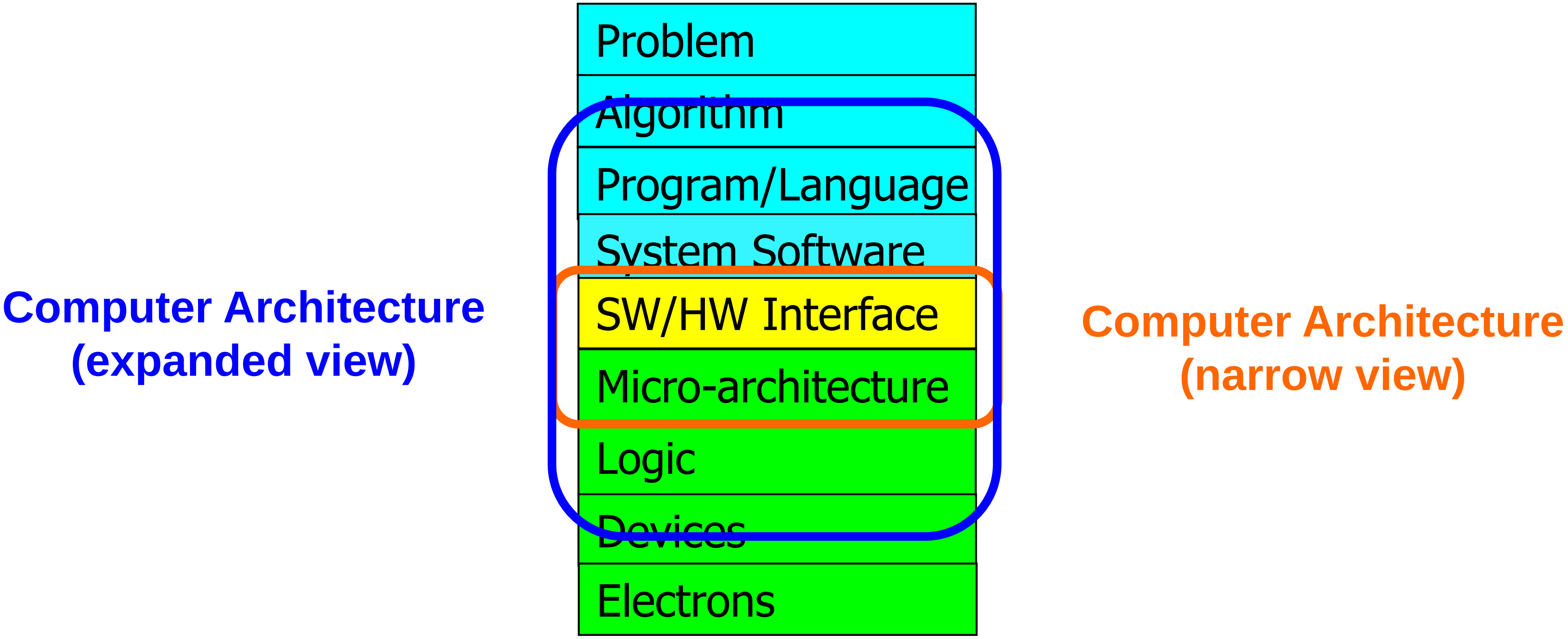
ETH Zurich

Spring 2018

23 February 2018

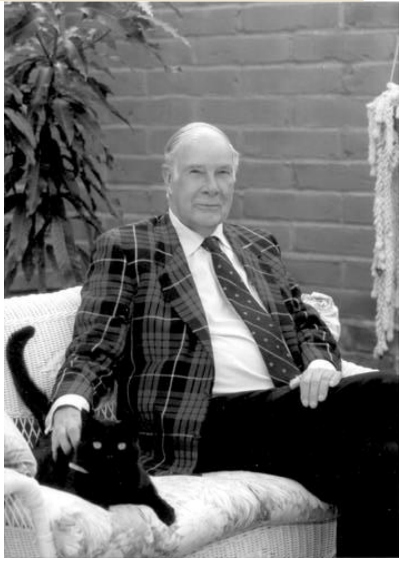
How Do Problems Get Solved by Electrons?

Recall: The Transformation Hierarchy



Recall: Levels of Transformation

"The purpose of computing is [to gain] insight" (*Richard Hamming*)
We gain and generate insight by solving problems
How do we ensure problems are solved by electrons?

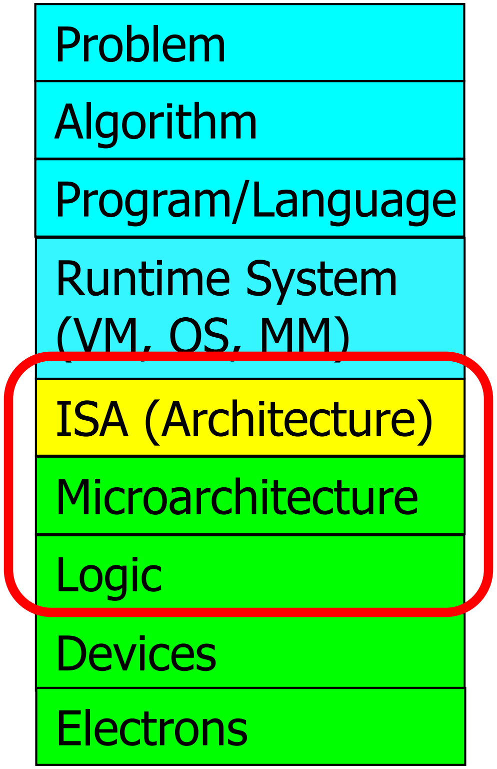


Algorithm

Step-by-step procedure that is **guaranteed to terminate** where **each step is precisely stated** and **can be carried out by a computer**

- **Finiteness**
- **Definiteness**
- **Effective computability**

Many algorithms for the same problem



ISA
(Instruction Set Architecture)

Interface/contract between SW and HW.

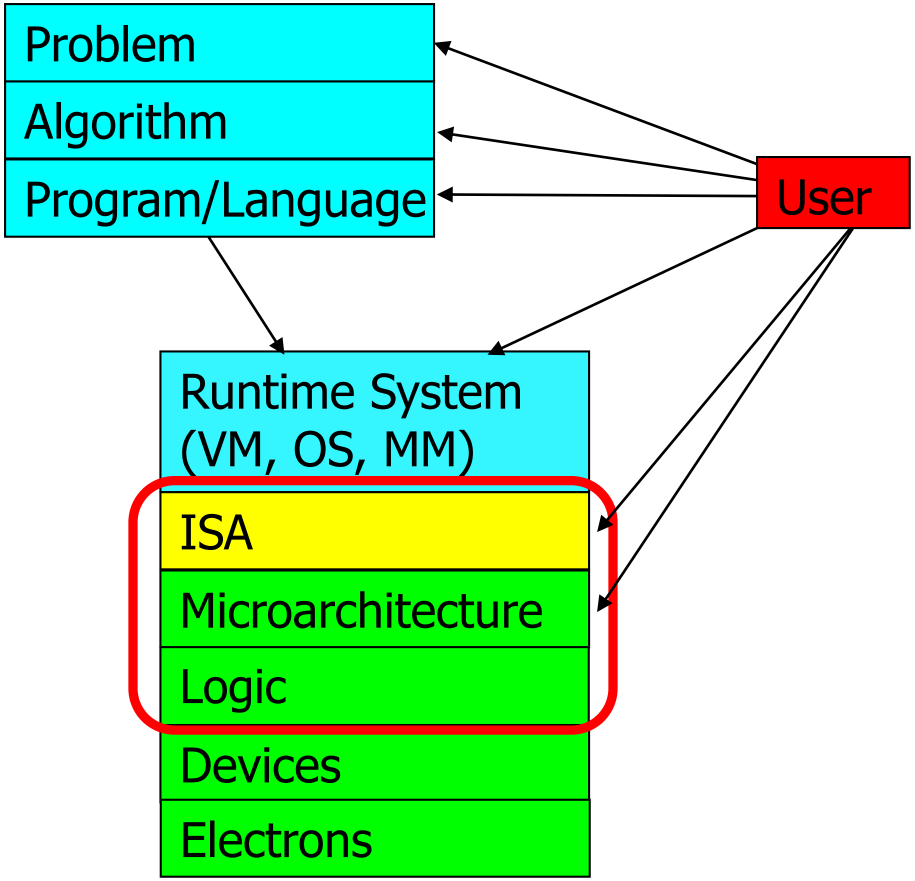
What the programmer assumes hardware will satisfy.

Microarchitecture
An implementation of the ISA

Digital logic circuits
Building blocks of micro-arch (e.g., gates)

Levels of Transformation, Revisited

- A user-centric view: computer designed for users



- The entire stack should be optimized for user
  - User varies across platforms and across use cases

The Power of Abstraction

- Levels of transformation create abstractions
  - Abstraction: A higher level only needs to know about the interface to the lower level, not how the lower level is implemented
  - E.g., high-level language programmer does not really need to know what the ISA is and how a computer executes instructions

- Abstraction improves productivity
  - No need to worry about decisions made in underlying levels
  - E.g., programming in Java vs. C vs. assembly vs. binary vs. by specifying control signals of each transistor every cycle

- Then, why would you want to know what goes on underneath or above?

Crossing the Abstraction Layers

- As long as everything goes well, not knowing what happens underneath (or above) is not a problem.

- What if
  - The program you wrote is running slow?
  - The program you wrote does not run correctly?
  - The program you wrote consumes too much energy?
  - Your system just shut down and you have no idea why?
  - Someone just compromised your system and you have no idea how?

- What if
  - The hardware you designed is too hard to program?
  - The hardware you designed is too slow because it does not provide the right primitives to the software?

- What if
  - You want to design a much more efficient and higher performance system?

Crossing the Abstraction Layers

- Two goals of this course (especially the second half) are
  - to understand how a processor works underneath the software layer and how decisions made in hardware affect the software/programmer
  - to enable you to be comfortable in making design and optimization decisions that cross the boundaries of different layers and system components

Some Example “Mysteries”

Four Mysteries: Familiar with Any?

- Meltdown & Spectre (2017-2018)
- Rowhammer (2012-2014)
- Memory Performance Attacks (2006-2007)
- Memories Forget: Refresh (2011-2012)

Mystery #1: Meltdown & Spectre

What Are These?

Source: J. Masters, Redhat, FOSDEM 2018 keynote talk.

Meltdown and Spectre Attacks

- Someone can steal secret data from the system even though
  - your program and data are perfectly correct and
  - your hardware behaves according to the specification and
  - there are no software vulnerabilities/bugs

Meltdown and Spectre

- Hardware security vulnerabilities that essentially effect almost all computer chips that were manufactured in the past two decades

- They exploit “speculative execution”
  - A technique employed in modern processors for high performance

- Speculative execution: Doing something before you know that it is needed
  - We do it all the time in life, to save time
    - Guess what will happen and act based on that guess
  - Processors do it, too, to run programs fast
    - They guess and execute code before they know it should be executed

Speculative Execution (I)

- Modern processors "speculatively execute" code to improve performance:

```
if (account-balance <= 0) {
    // do something
} else if (account-balance < 1M) {
    // do something else
} else {
    // do something else
}
```

Guess what code will be executed and execute it speculatively

- Improves performance, if it takes a long time to access account-balance

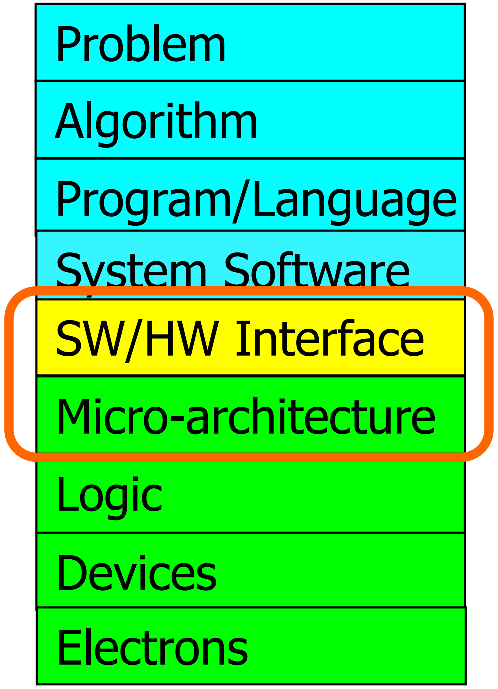
If the guess was wrong, flush the wrong instructions and execute the correct code

Speculative Execution is Invisible to the User

ISA
(Instruction Set Architecture)

Interface/contract between SW and HW.

What the programmer assumes hardware will satisfy.



Microarchitecture
An implementation of the ISA

Programmer assumes their code will be executed in sequential order

Microarchitecture executes instructions in a different order, speculatively – but reports the results as expected by the programmer

Meltdown and Spectre

- Someone can steal secret data from the system even though
  - your program and data are perfectly correct and
  - your hardware behaves according to the specification and
  - there are no software vulnerabilities/bugs

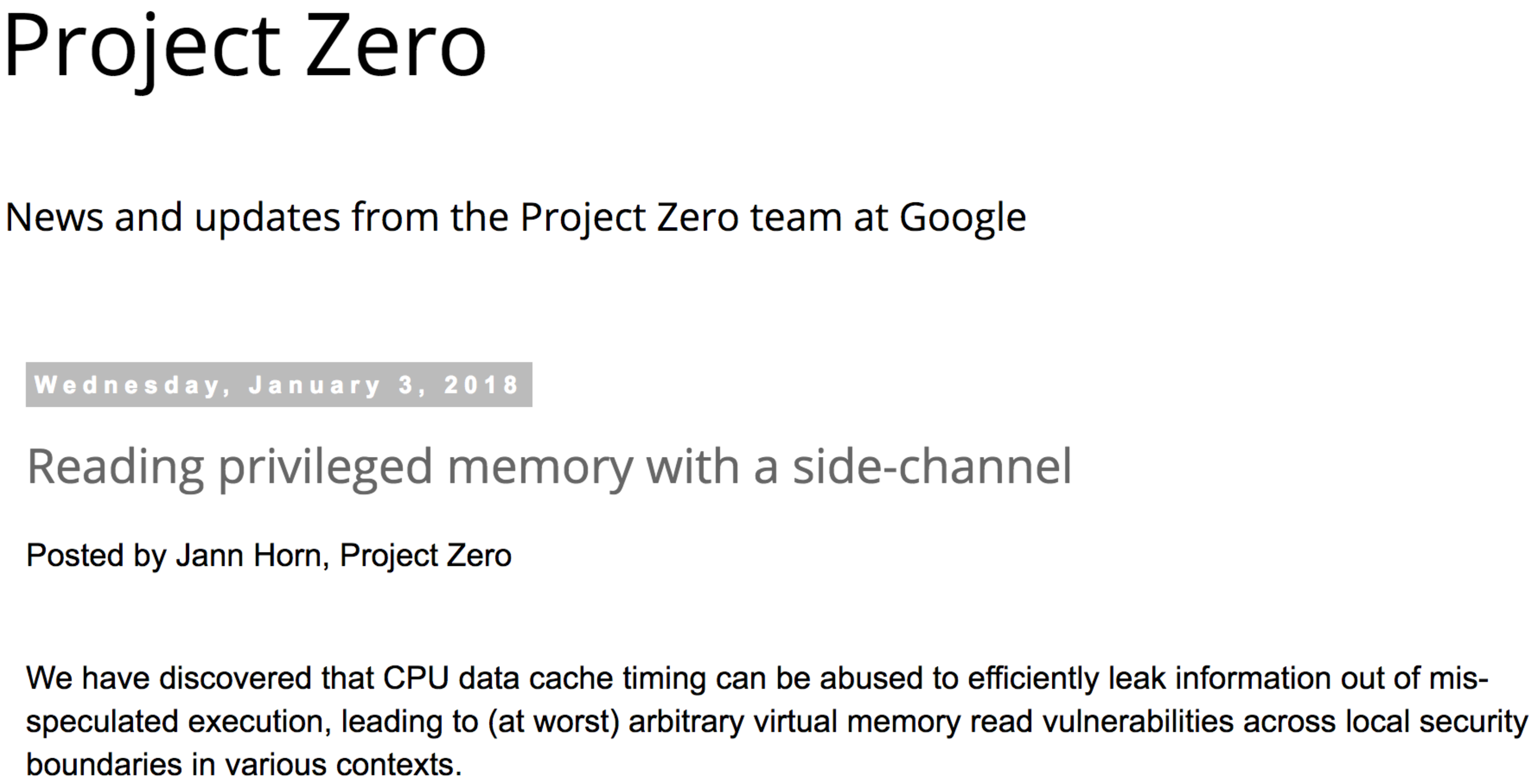
- Why?
  - Speculative execution leaves traces of secret data in the processor’s cache (internal storage)
    - It brings data that is not supposed to be brought/accessed if there was no speculative execution
  - A malicious program can inspect the contents of the cache to “infer” secret data that it is not supposed to access
  - A malicious program can actually force another program to speculatively execute code that leaves traces of secret data

Processor Cache as a Side Channel

- Speculative execution leaves traces of data in processor cache
  - **Architecturally correct behavior w.r.t. specification**
  - However, this leads to a side channel: a channel through which someone sophisticated can extract information

- Processor cache leaks information by storing speculatively-accessed data
  - A clever attacker can probe the cache and infer the secret data values
    - by measuring how long it takes to access the data
  - A clever attacker can force a program to speculatively execute code and leave traces of secret data in the cache

More on Meltdown/Spectre Side Channels



Project Zero

News and updates from the Project Zero team at Google

Wednesday, January 3, 2018

Reading privileged memory with a side-channel

Posted by Jann Horn, Project Zero

We have discovered that CPU data cache timing can be abused to efficiently leak information out of mis-speculated execution, leading to (at worst) arbitrary virtual memory read vulnerabilities across local security boundaries in various contexts.

Source: https://googleprojectzero.blogspot.ch/2018/01/reading-privileged-memory-with-side.html

Three Questions

- Can you figure out why someone stole your secret data if you do not know how the processor executes a program?

- Can you fix the problem without knowing what is happening "underneath", i.e., inside the microarchitecture?

- Can you fix the problem well/fundamentally without knowing both software and hardware design?

- Can you construct this attack or similar attacks without knowing what is happening underneath?

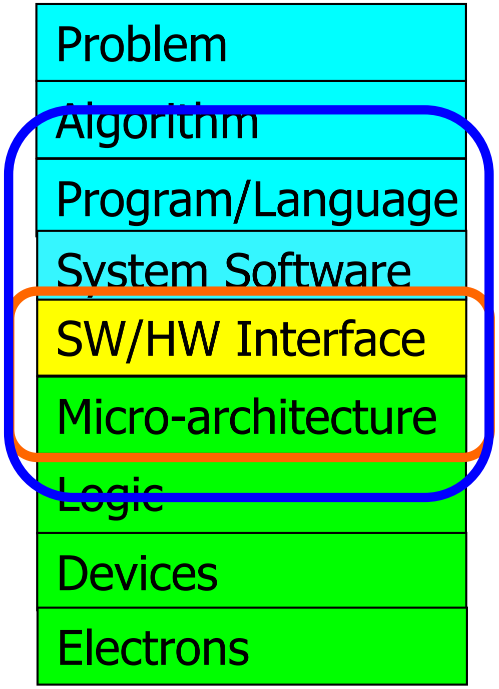
Three Other Questions

- What are the causes of Meltdown and Spectre?

- How can we prevent them (while keeping the performance benefits of speculative execution)?
  - Software changes?
  - Operating system changes?
  - Instruction set architecture changes?
  - Microarchitecture/hardware changes?
  - Changes at multiple layers, done cooperatively?
  - …

- How do we design high-performance processors that do not leak information via side channels?

Meltdown/Spectre Span Across the Hierarchy



Problem

Algorithm

Program/Language

System Software

SW/HW Interface

Micro-architecture

Logic

Devices

Electrons

Computer Architecture
(expanded view)

Meltdown/Spectre problem
and solution space

Computer Architecture
(narrow view)

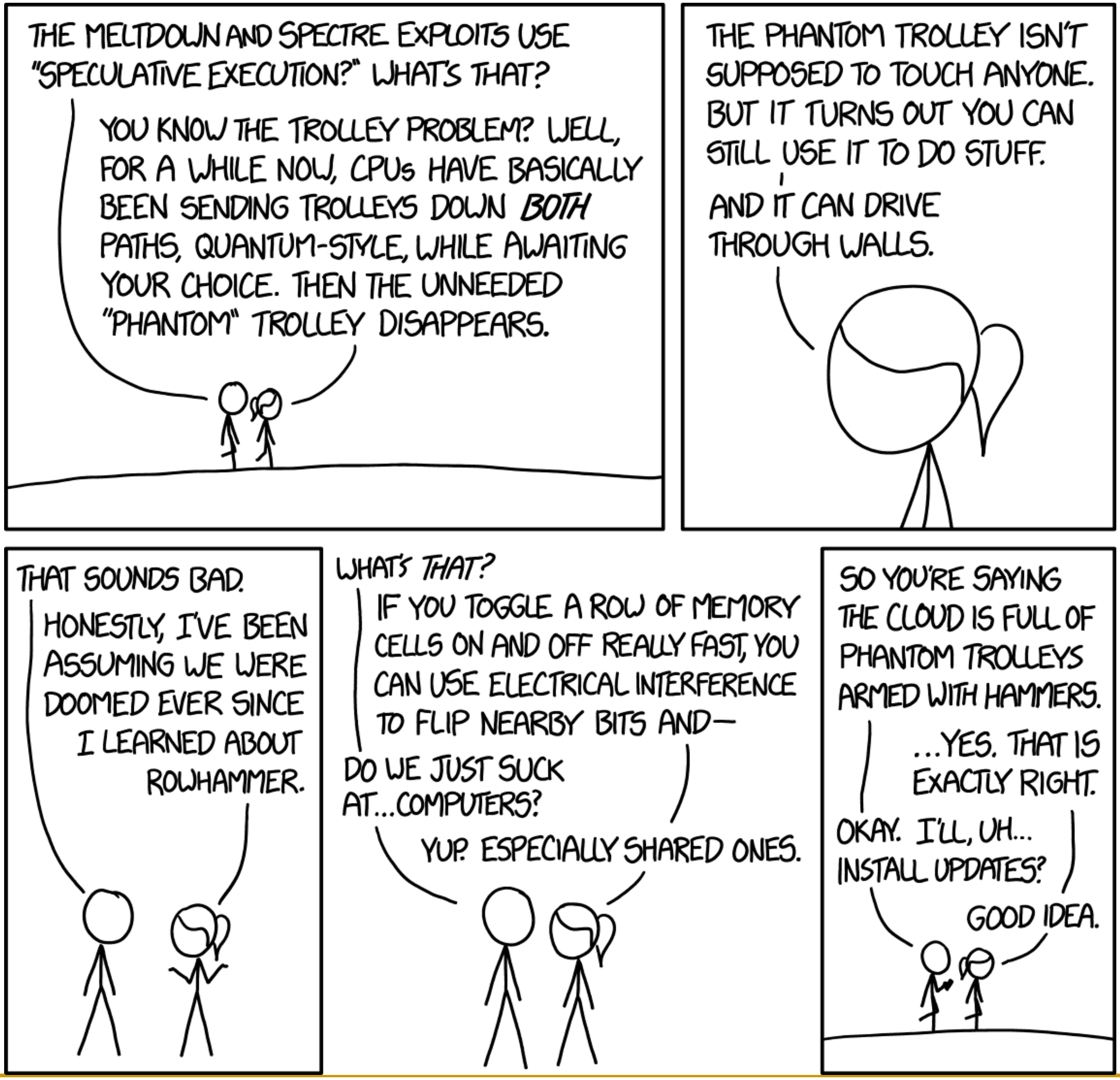
Takeaway

Breaking the abstraction layers (between components and transformation hierarchy levels)

and knowing what is underneath

enables you to **understand** and **solve** problems

… and Also Understand/Critique Cartoons!


THE MELTDOWN AND SPECTRE EXPLOITS USE "SPECULATIVE EXECUTION?" WHAT'S THAT?
YOU KNOW THE TROLLEY PROBLEM? WELL, FOR A WHILE NOW, CPUs HAVE BASICALLY BEEN SENDING TROLLEYS DOWN *BOTH* PATHS, QUANTUM-STYLE, WHILE AWAITING YOUR CHOICE. THEN THE UNNEEDED "PHANTOM" TROLLEY DISAPPEARS.
THE PHANTOM TROLLEY ISN'T SUPPOSED TO TOUCH ANYONE. BUT IT TURNS OUT YOU CAN STILL USE IT TO DO STUFF.
AND IT CAN DRIVE THROUGH WALLS.
THAT SOUNDS BAD.
HONESTLY, I'VE BEEN ASSUMING WE WERE DOOMED EVER SINCE I LEARNED ABOUT ROWHAMMER.
WHAT'S *THAT?*
IF YOU TOGGLE A ROW OF MEMORY CELLS ON AND OFF REALLY FAST, YOU CAN USE ELECTRICAL INTERFERENCE TO FLIP NEARBY BITS AND—
DO WE JUST SUCK AT...COMPUTERS?
YUP. ESPECIALLY SHARED ONES.
SO YOU'RE SAYING THE CLOUD IS FULL OF PHANTOM TROLLEYS ARMED WITH HAMMERS.
...YES. THAT IS EXACTLY RIGHT.
OKAY. I'LL, UH... INSTALL UPDATES?
GOOD IDEA.

Source: https://xkcd.com/1938/

An Important Note: Design Goal and Mindset

- Design goal of a system determines the design mindset and evaluation metrics

- Meltdown and Spectre are there because the design goal of cutting-edge processors (employed everywhere in our lives)
  - has mainly been focused on high performance and low energy (relatively recently)
  - has not included security (or information leakage) as an important constraint

- Incorporating security as a first-class constraint and "metric" into (hardware) design and education is critical in today's world

Design Mindset

Security is about preventing unforeseen consequences

Source: https://s-media-cache-ak0.pinimg.com/originals/48/09/54/4809543a9c7700246a0cf8acdae27abf.jpg

Two Other Goals of This Course

- Enable you to think critically

- Enable you to think broadly

To Learn and Discover Further

- High-level Video by RedHat
  - https://www.youtube.com/watch?v=syAdX44pokE

- A bit lower-level, comprehensive explanation by Y. Vigfusson
  - https://www.youtube.com/watch?v=mgAN4w7LH2o

- Keep attending lectures and taking in all the material

- Come talk with me in the future
  - I have many bachelor's & master's projects on hardware security
  - "Fundamentally secure computing architectures" is a key direction of scientific investigation and design

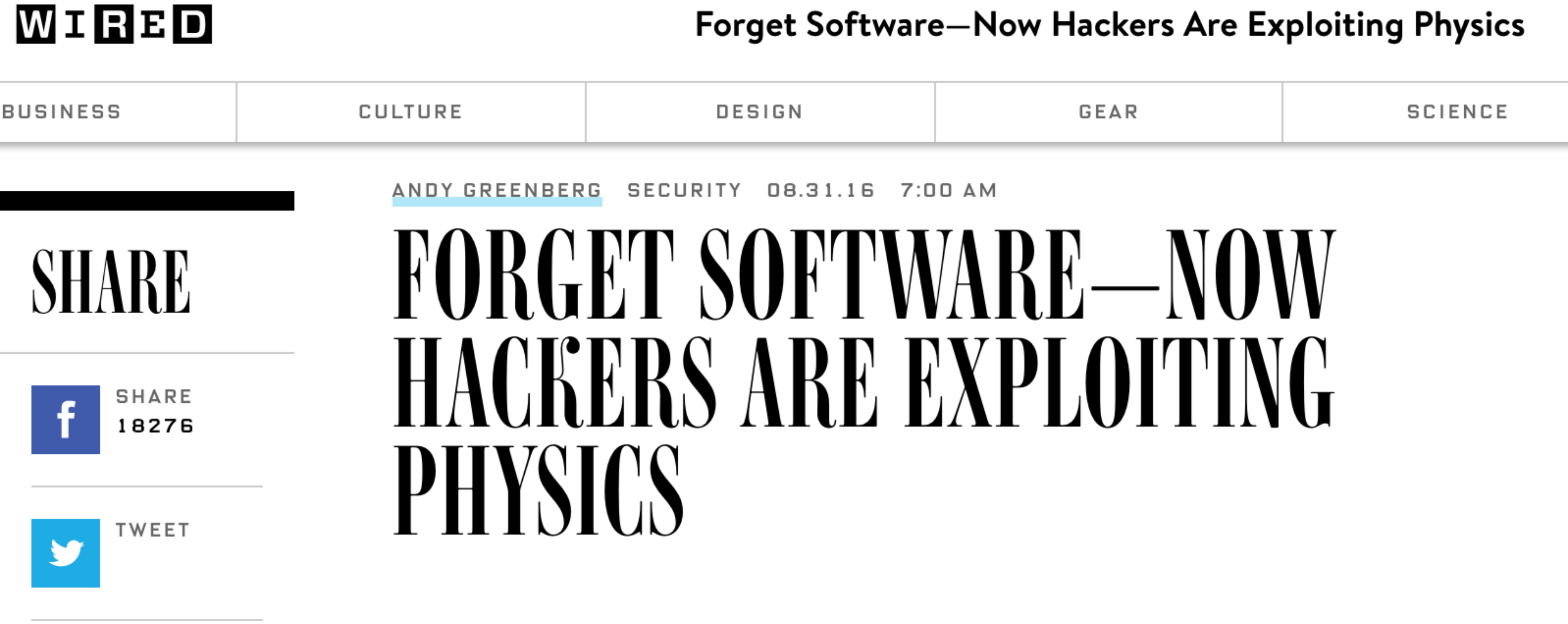
Another Example “Mystery”

Mystery #2: RowHammer

RowHammer: Another Mystery?

- DRAM Row Hammer (or, DRAM Disturbance Errors)

- How a simple hardware failure mechanism can create a widespread system security vulnerability



WIRED

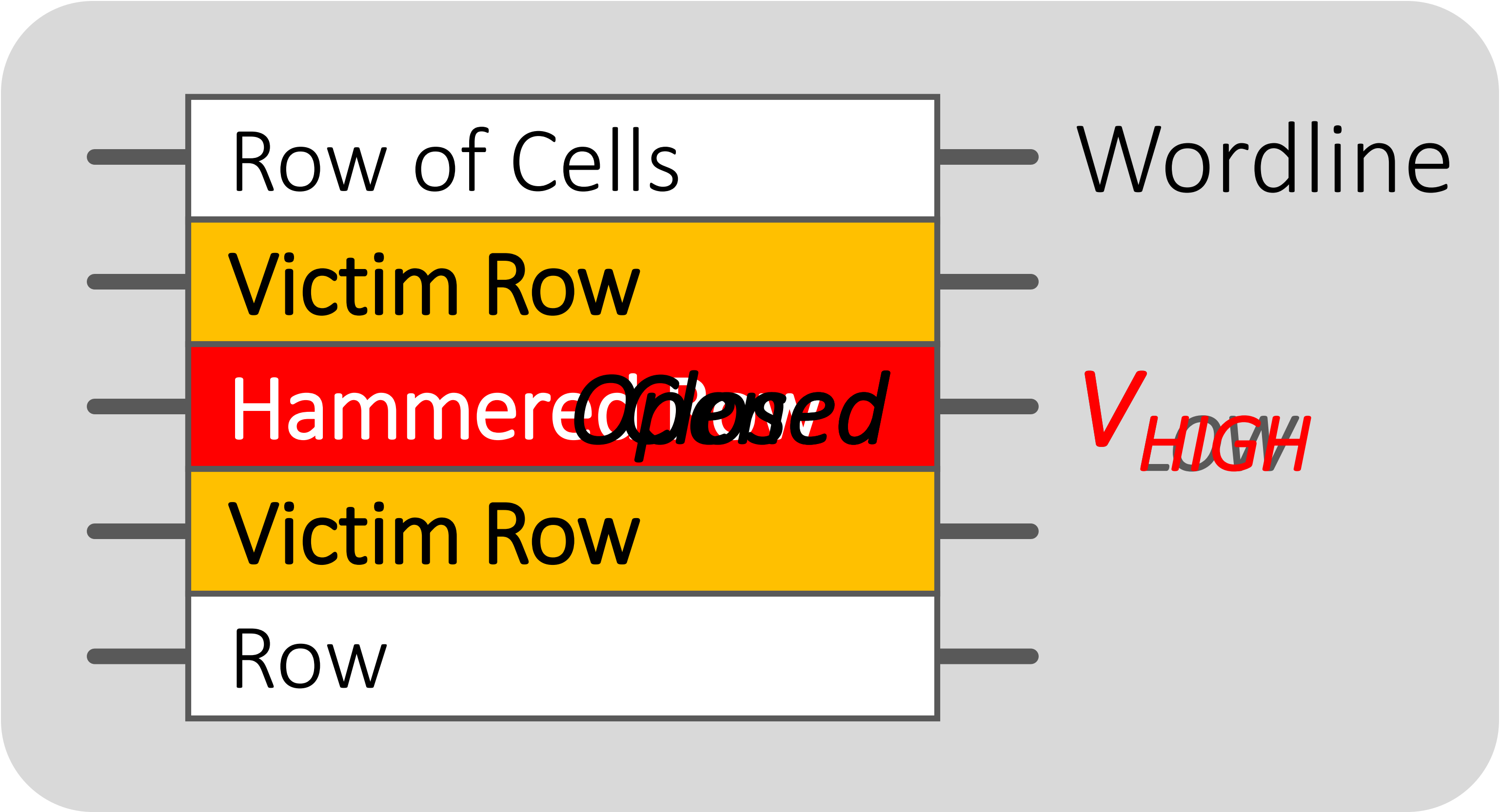
Forget Software—Now Hackers Are Exploiting Physics

BUSINESS | CULTURE | DESIGN | GEAR | SCIENCE

ANDY GREENBERG SECURITY 08.31.16 7:00 AM

FORGET SOFTWARE—NOW HACKERS ARE EXPLOITING PHYSICS

Modern DRAM is Prone to Disturbance Errors

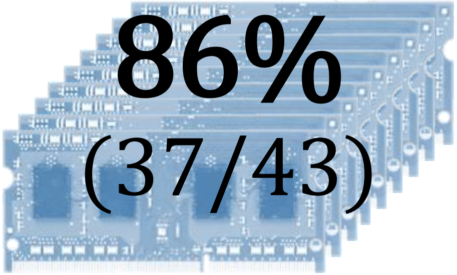


Repeatedly opening and closing a row enough times within a refresh interval induces disturbance errors in adjacent rows in most real DRAM chips you can buy today

Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors, (Kim et al., ISCA 2014)

Most DRAM Modules Are Vulnerable

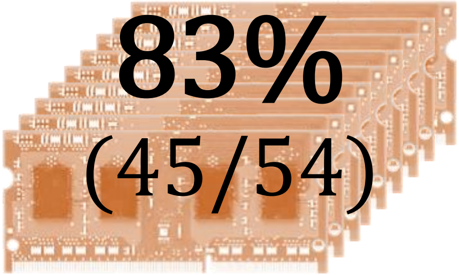
A company



86%
(37/43)

Up to
1.0×10^7
errors

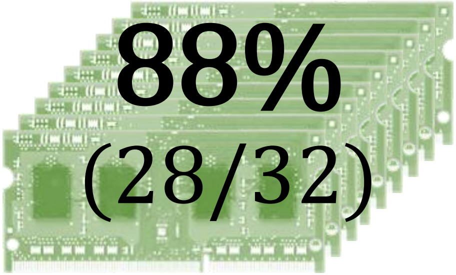
B company



83%
(45/54)

Up to
2.7×10^6
errors

C company

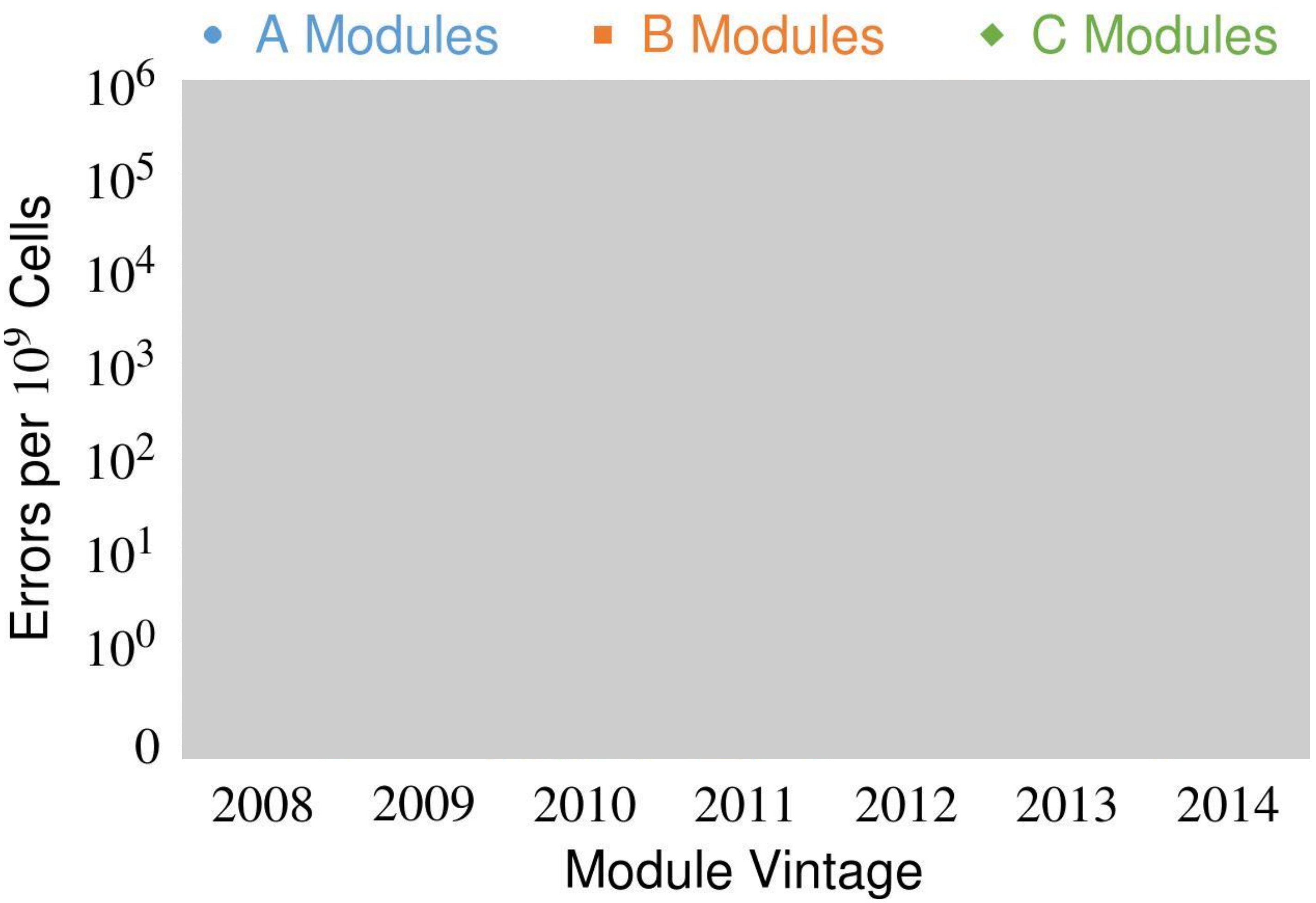


88%
(28/32)

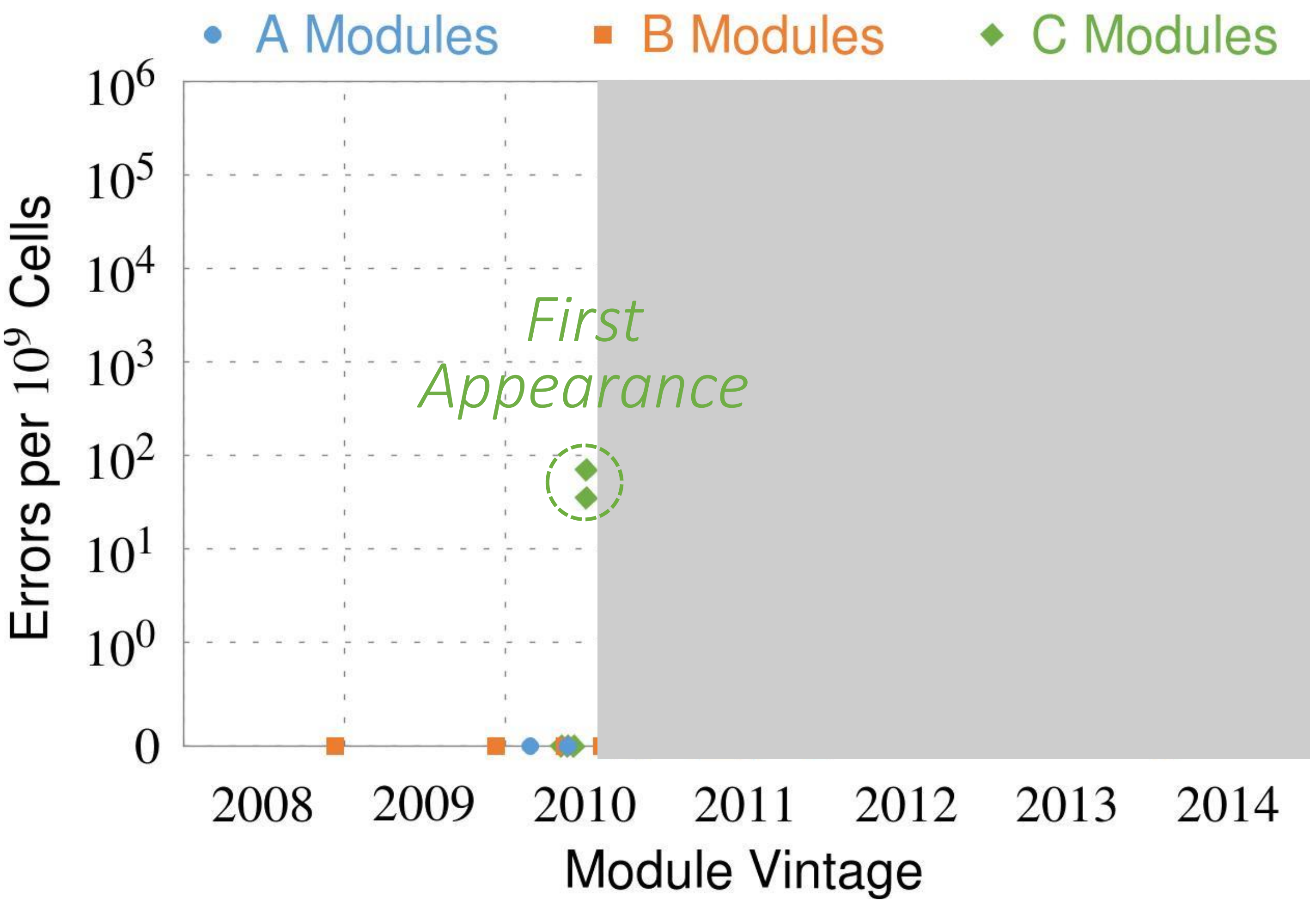
Up to
3.3×10^5
errors

Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors, (Kim et al., ISCA 2014)

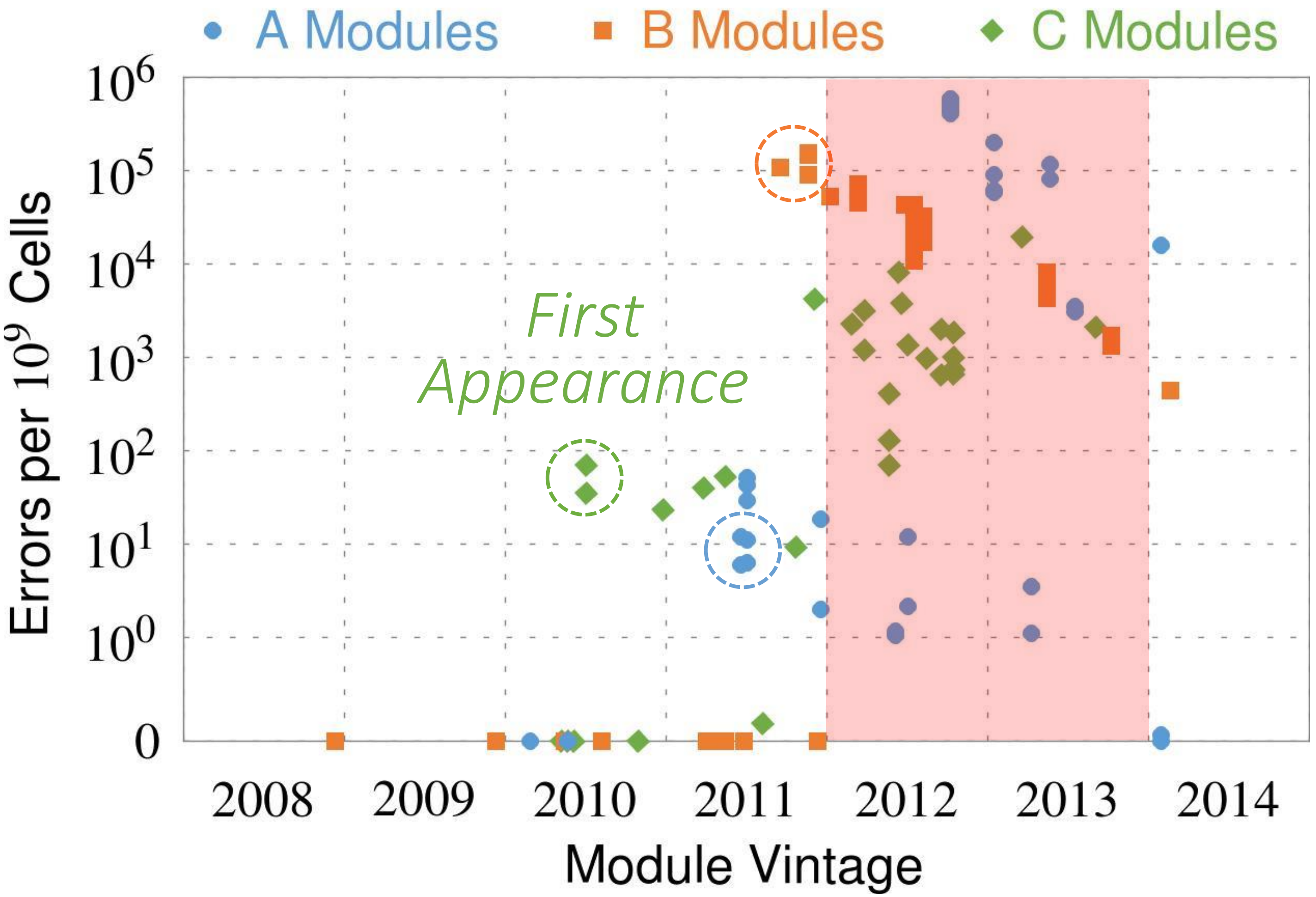
Recent DRAM Is More Vulnerable



Recent DRAM Is More Vulnerable



Recent DRAM Is More Vulnerable



All modules from 2012–2013 are vulnerable

Why Is This Happening?

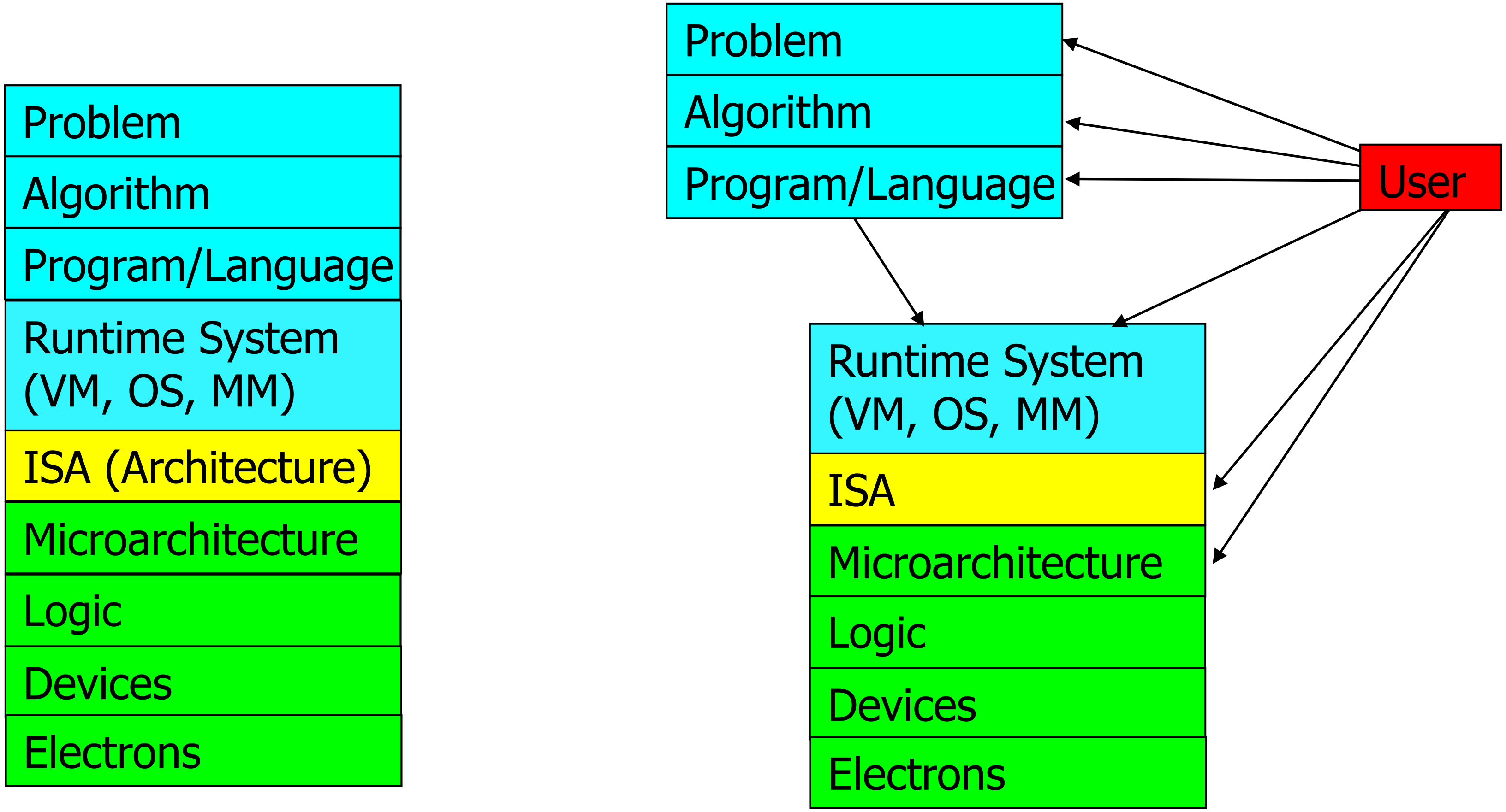
- DRAM cells are too close to each other!
  - They are not electrically isolated from each other

- Access to one cell affects the value in nearby cells
  - due to electrical interference between
    - the cells
    - 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

Higher-Level Implications

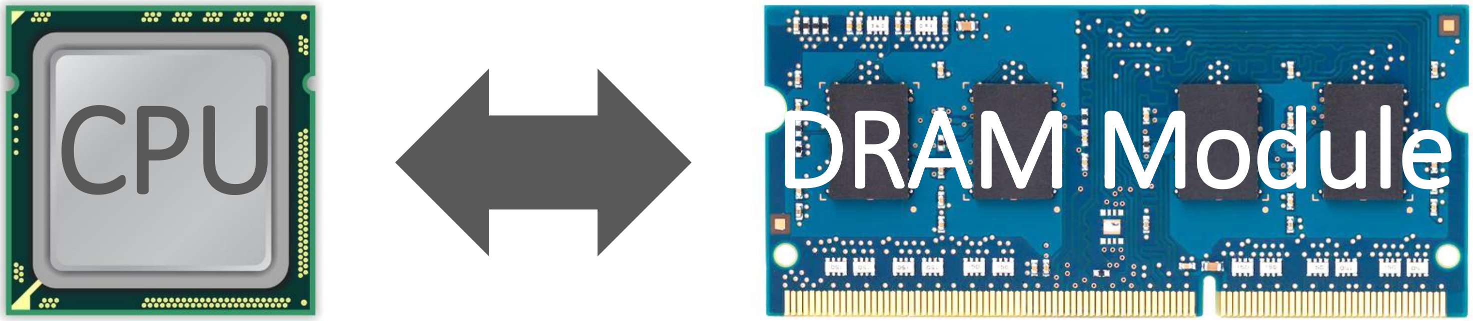
- This simple circuit-level failure mechanism has enormous implications on upper layers of the transformation hierarchy



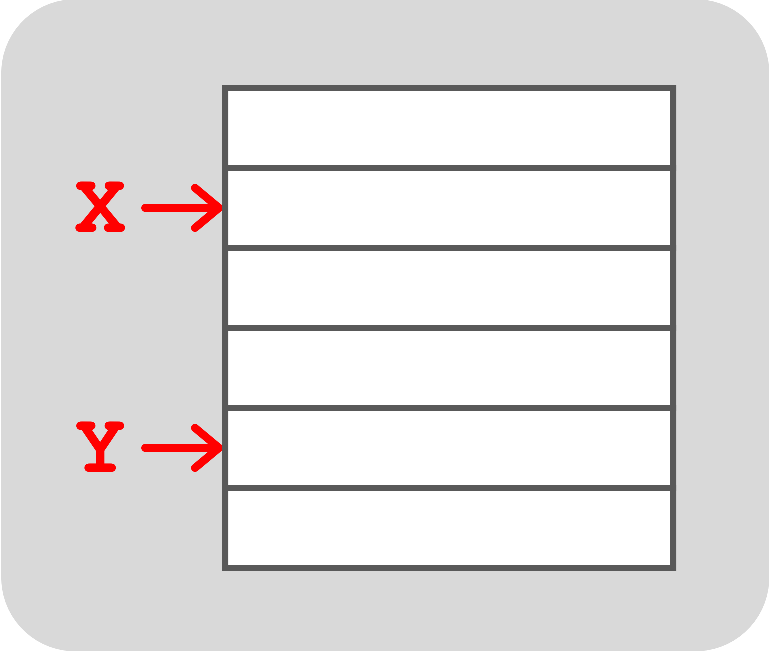
Problem
Algorithm
Program/Language
Runtime System (VM, OS, MM)
ISA (Architecture)
Microarchitecture
Logic
Devices
Electrons

Problem
Algorithm
Program/Language
User
Runtime System (VM, OS, MM)
ISA
Microarchitecture
Logic
Devices
Electrons

A Simple Program Can Induce Many Errors

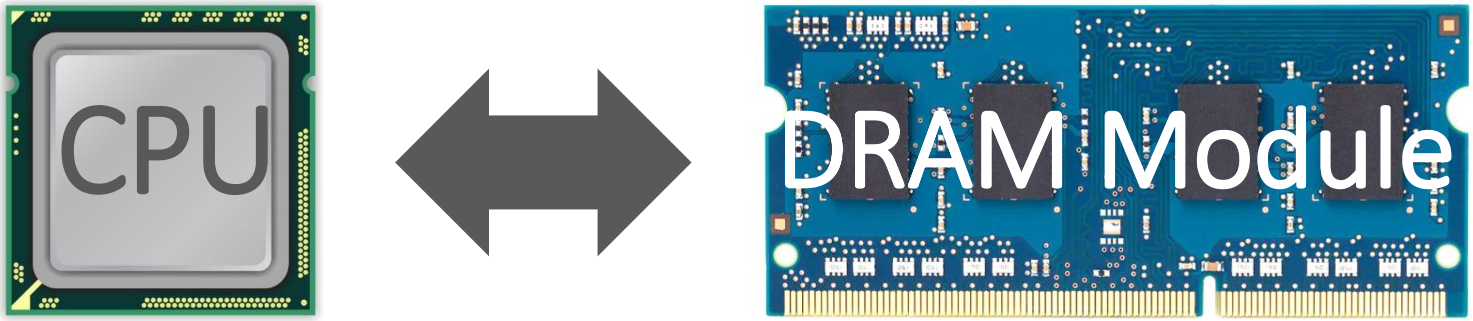


```
loop:
   mov (X), %eax
   mov (Y), %ebx
   clflush (X)
   clflush (Y)
   mfence
   jmp loop
```



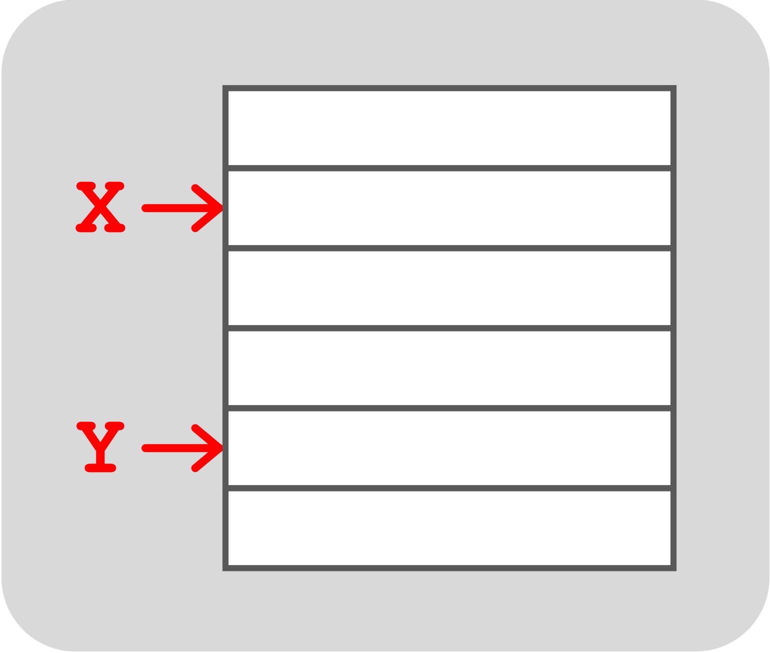
Download from: https://github.com/CMU-SAFARI/rowhammer

A Simple Program Can Induce Many Errors



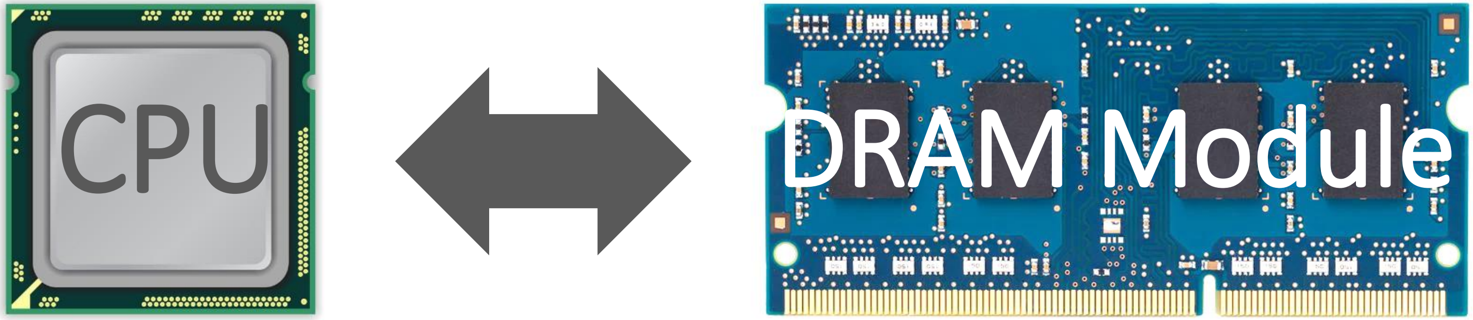
1. Avoid ***cache hits***
   - Flush **X** from cache

2. Avoid ***row hits*** to **X**
   - Read **Y** in another row

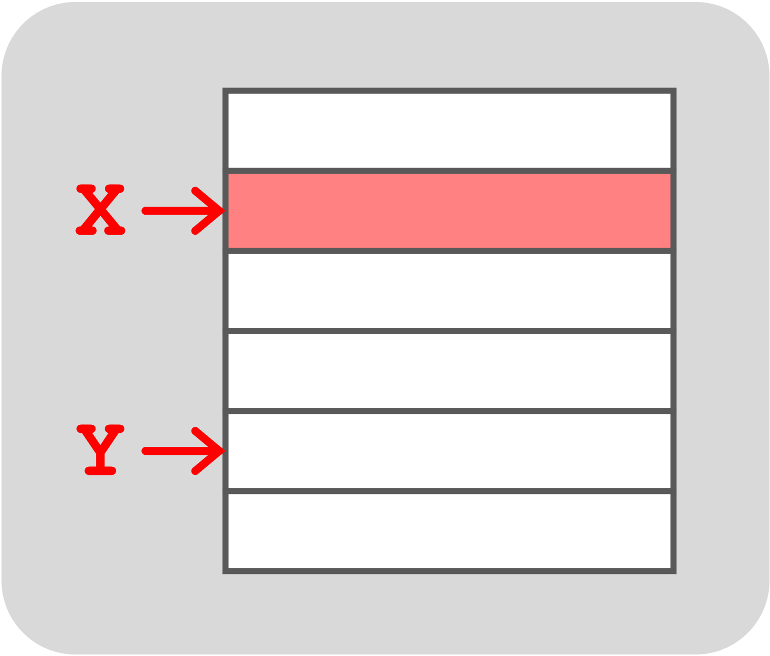


Download from: https://github.com/CMU-SAFARI/rowhammer

A Simple Program Can Induce Many Errors

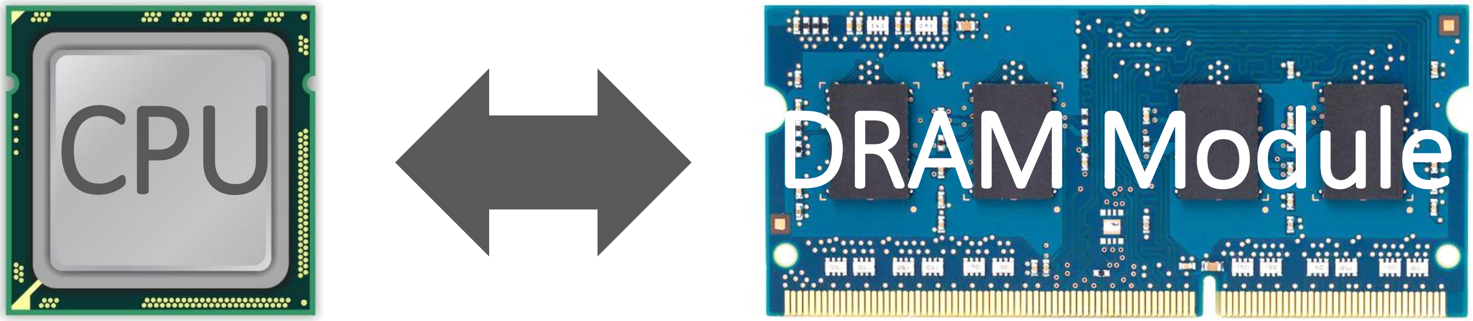


```
loop:
   mov  (X), %eax
   mov  (Y), %ebx
   clflush (X)
   clflush (Y)
   mfence
   jmp loop
```

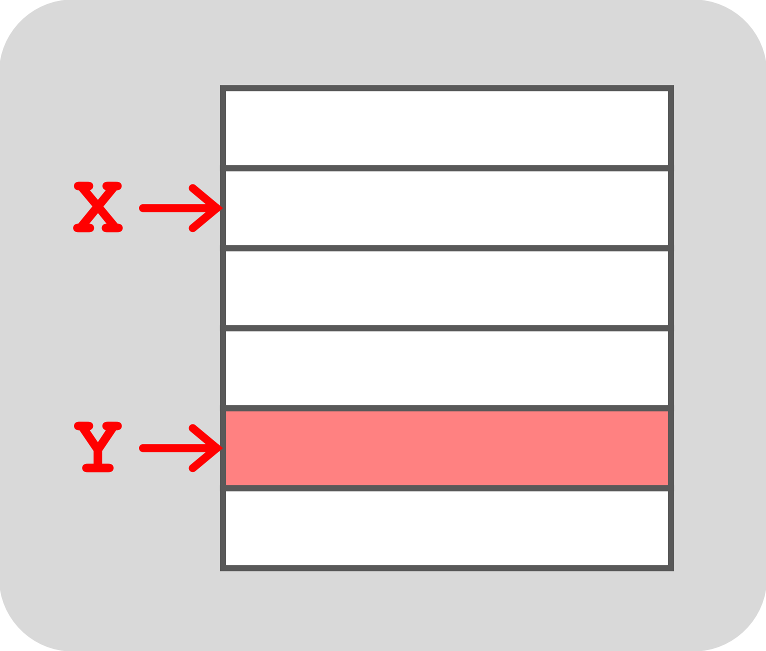


Download from: https://github.com/CMU-SAFARI/rowhammer

A Simple Program Can Induce Many Errors

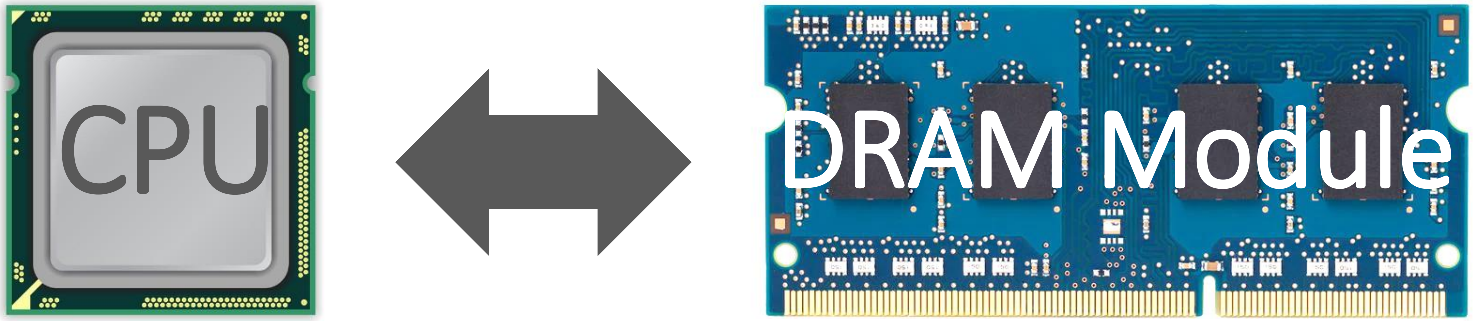


```
loop:
   mov  (X), %eax
   mov  (Y), %ebx
   clflush (X)
   clflush (Y)
   mfence
   jmp loop
```

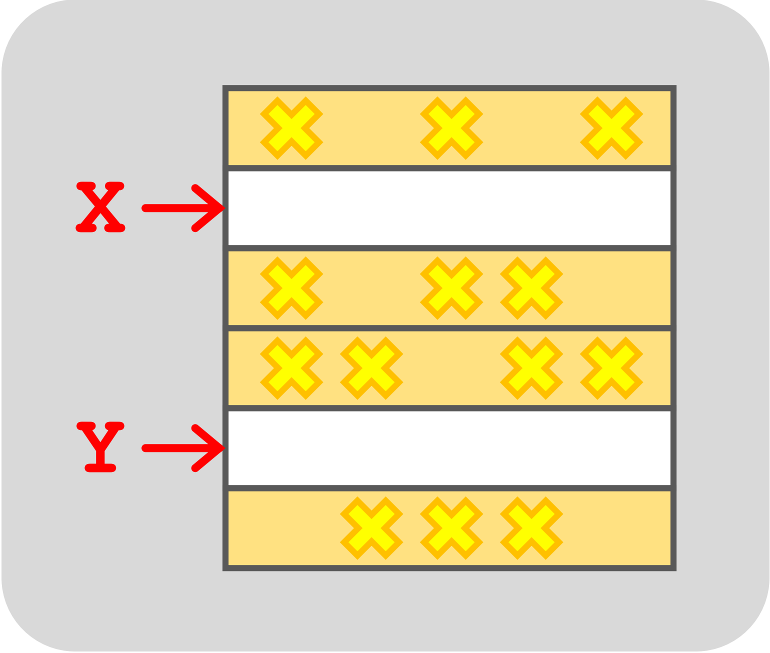


Download from: https://github.com/CMU-SAFARI/rowhammer

A Simple Program Can Induce Many Errors



```
loop:
   mov (X), %eax
   mov (Y), %ebx
   clflush (X)
   clflush (Y)
   mfence
   jmp loop
```



Download from: https://github.com/CMU-SAFARI/rowhammer

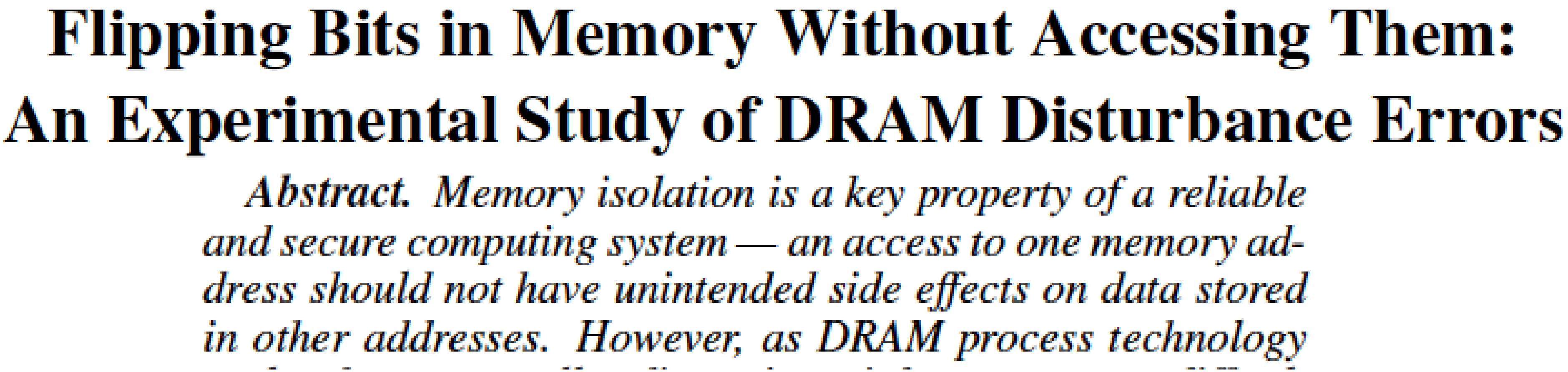
Observed Errors in Real Systems

CPU Architecture	Errors	Access-Rate
Intel Haswell (2013)	22.9K	12.3M/sec
Intel Ivy Bridge (2012)	20.7K	11.7M/sec
Intel Sandy Bridge (2011)	16.1K	11.6M/sec
AMD Piledriver (2012)	59	6.1M/sec

A real reliability & security issue

Kim+, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors," ISCA 2014.

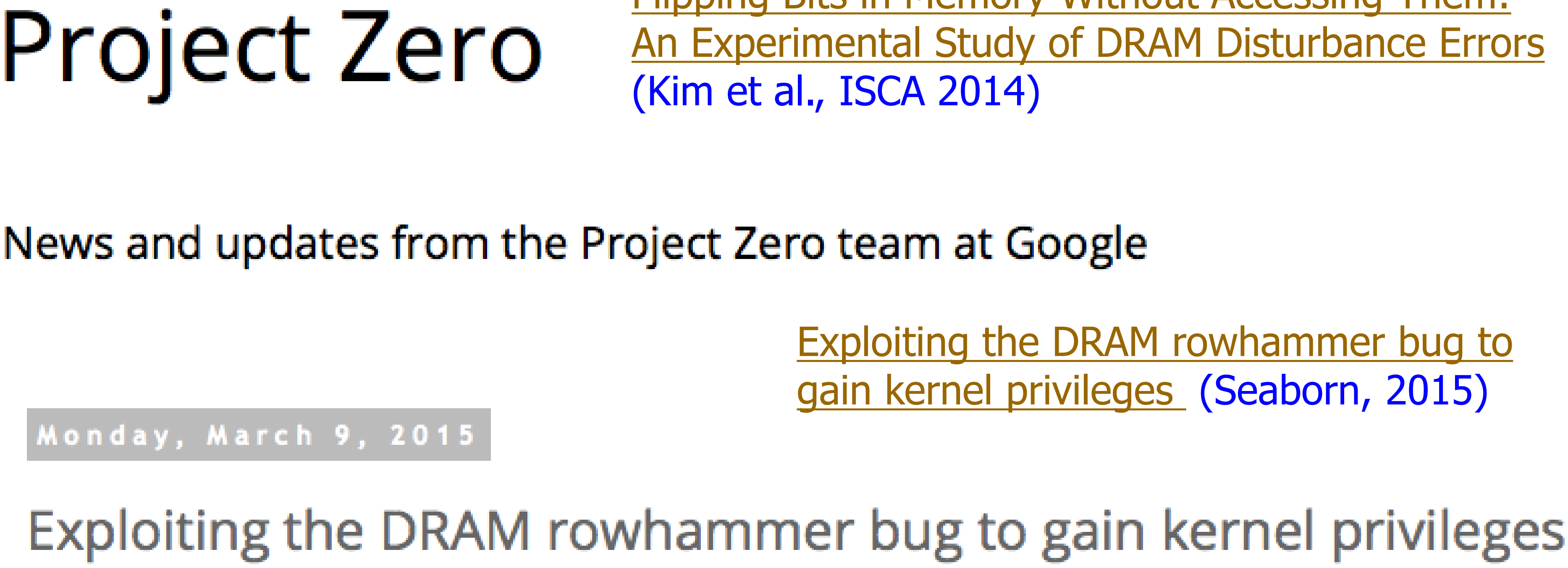
One Can Take Over an Otherwise-Secure System



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

Abstract. Memory isolation is a key property of a reliable and secure computing system — an access to one memory address should not have unintended side effects on data stored in other addresses. However, as DRAM process technology

Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors (Kim et al., ISCA 2014)



Project Zero

News and updates from the Project Zero team at Google

Monday, March 9, 2015

Exploiting the DRAM rowhammer bug to gain kernel privileges

Exploiting the DRAM rowhammer bug to gain kernel privileges (Seaborn, 2015)

RowHammer Security Attack Example

- "Rowhammer" is a problem with some recent DRAM devices in which repeatedly accessing a row of memory can cause bit flips in adjacent rows (Kim et al., ISCA 2014).
  - Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors (Kim et al., ISCA 2014)

- We tested a selection of laptops and found that a subset of them exhibited the problem.
- We built two working privilege escalation exploits that use this effect.
  - Exploiting the DRAM rowhammer bug to gain kernel privileges (Seaborn, 2015)

- One exploit uses rowhammer-induced bit flips to gain kernel privileges on x86-64 Linux when run as an unprivileged userland process.
- When run on a machine vulnerable to the rowhammer problem, the process was able to induce bit flips in page table entries (PTEs).
- It was able to use this to gain write access to its own page table, and hence gain read-write access to all of physical memory.

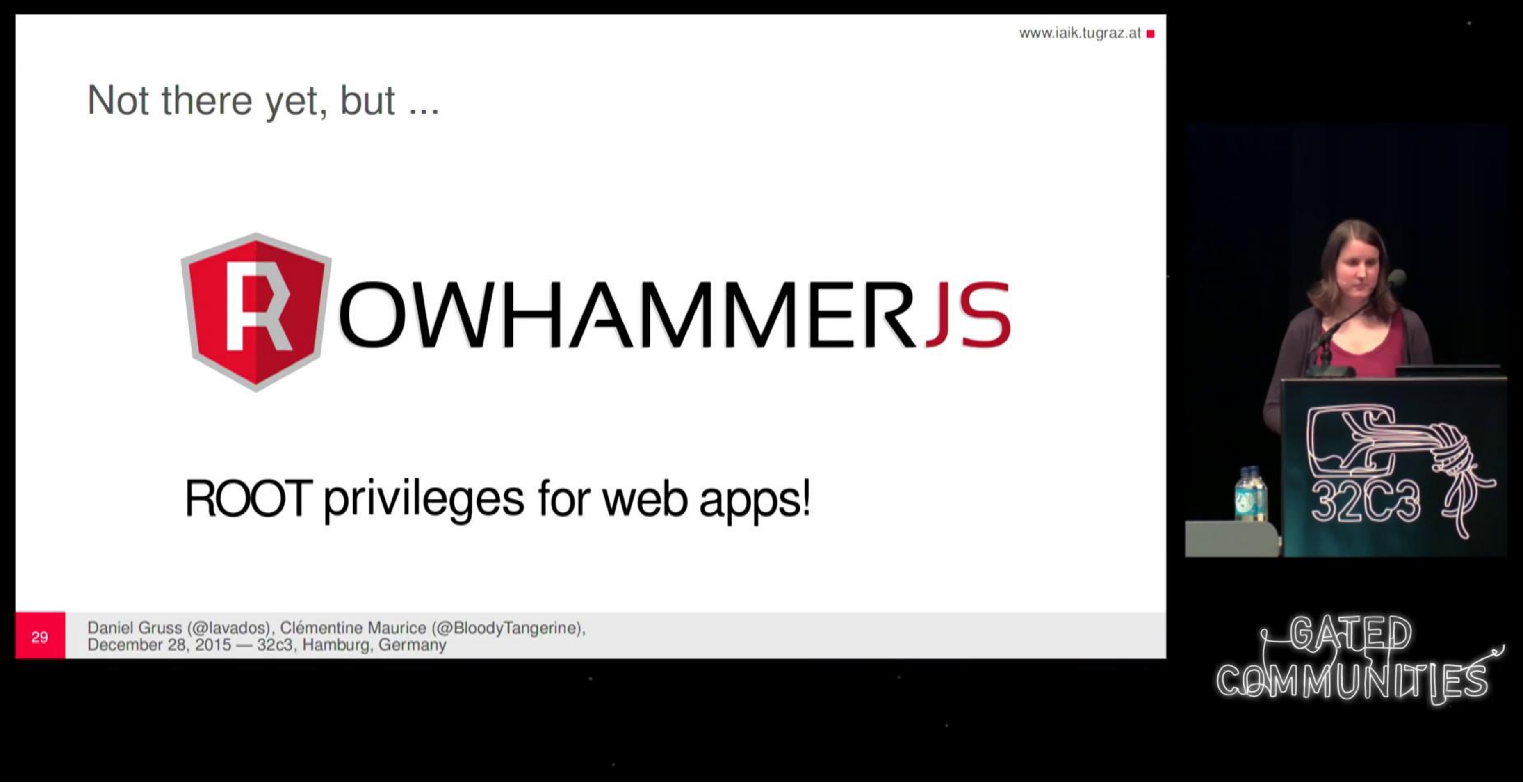
Exploiting the DRAM rowhammer bug to gain kernel privileges (Seaborn, 2015)

Security Implications

It’s like breaking into an apartment by repeatedly slamming a neighbor’s door until the vibrations open the door you were after

More Security Implications

"We can gain unrestricted access to systems of website visitors."



www.iaik.tugraz.at

Not there yet, but ...

ROWHAMMERJS

ROOT privileges for web apps!

29

Daniel Gruss (@lavados), Clémentine Maurice (@BloodyTangerine),
December 28, 2015 — 32c3, Hamburg, Germany

Rowhammer.js: A Remote Software-Induced Fault Attack in JavaScript (DIMVA'16)

Source: https://lab.dsst.io/32c3-slides/7197.html

More Security Implications

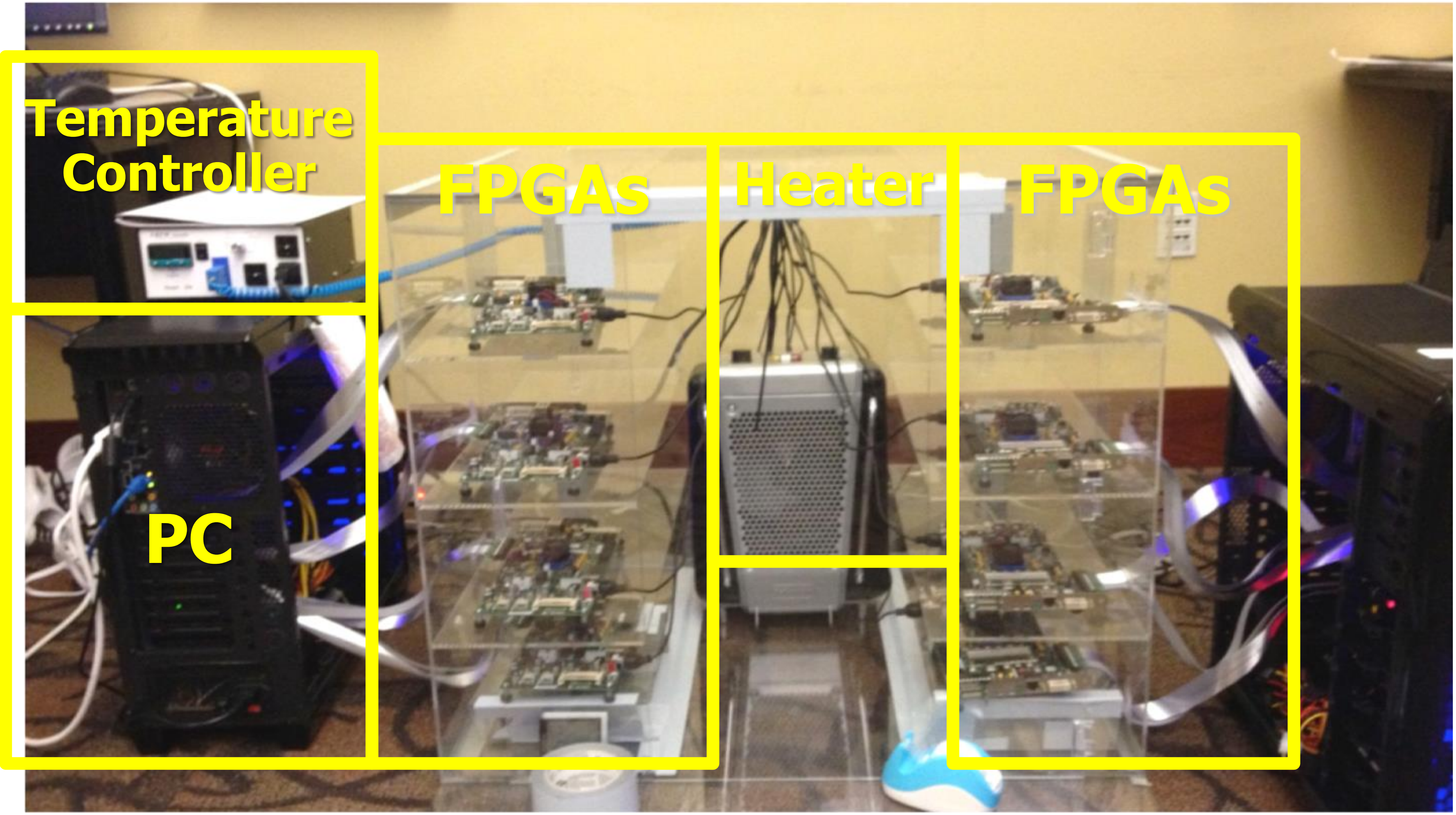
"Can gain control of a smart phone deterministically"

Drammer: Deterministic Rowhammer Attacks on Mobile Platforms, CCS'16

Source: https://fossbytes.com/drammer-rowhammer-attack-android-root-devices/

More Security Implications?

Where RowHammer Was Discovered…



Kim+, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors," ISCA 2014.

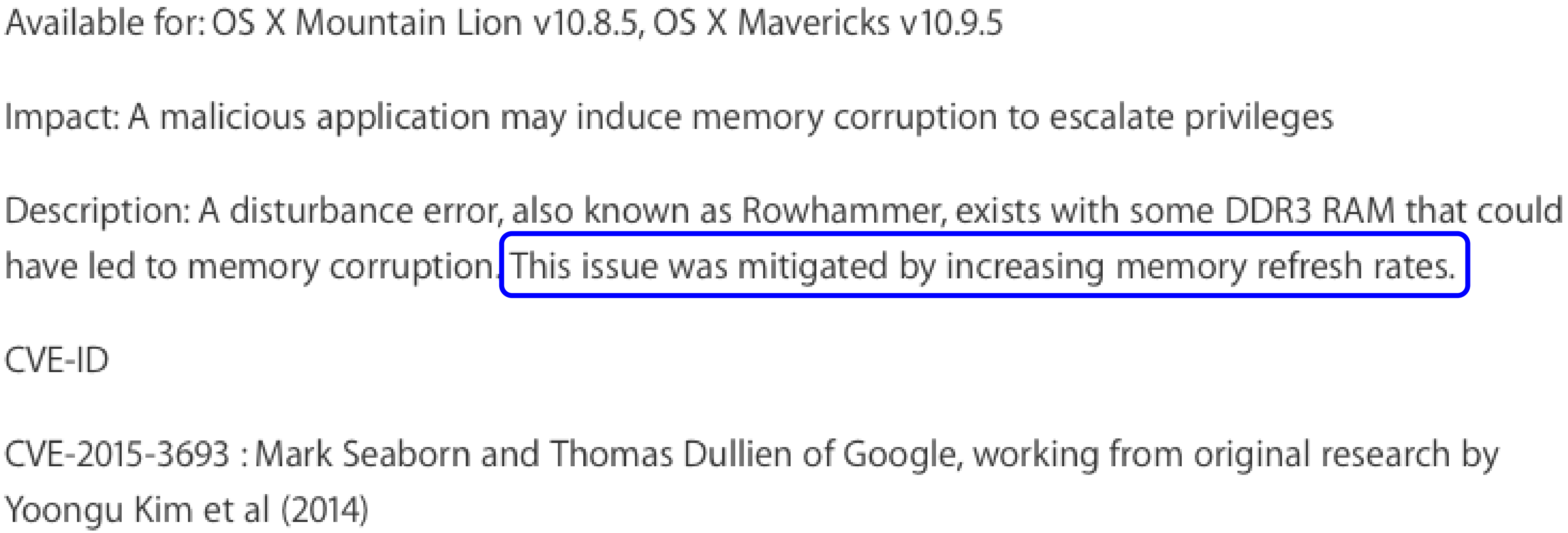
How Do We Fix The Problem?

Some Potential Solutions

• Make better DRAM chips	Cost
• Refresh frequently	Power, Performance
• Sophisticated Error Correction	Cost, Power
• Access counters	Cost, Power, Complexity

Apple’s Security Patch for RowHammer

- https://support.apple.com/en-gb/HT204934



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

A Cheaper Solution

- PARA: *Probabilistic Adjacent Row Activation*

- Key Idea
  - After closing a row, we activate (i.e., refresh) one of its neighbors with a low probability: $p = 0.005$

- Reliability Guarantee
  - When $p=0.005$, errors in one year: 9.4×10^{-14}
  - By adjusting the value of p, we can provide an arbitrarily strong protection against errors

Some Thoughts on RowHammer

- A simple hardware failure mechanism can create a widespread system security vulnerability

- How to find, exploit and fix the vulnerability requires a strong understanding across the transformation layers
  - And, a strong understanding of tools available to you

- Fixing needs to happen for two types of chips
  - Existing chips (already in the field)
  - Future chips
- Mechanisms for fixing are different between the two types

Aside: Byzantine Failures

- This class of failures is known as Byzantine failures

- Characterized by
  - Undetected erroneous computation
  - Opposite of "fail fast (with an error or no result)"

- "erroneous" can be "malicious" (intent is the only distinction)
- Very difficult to detect and confine Byzantine failures
- Do all you can to avoid them

- Lamport et al., "The Byzantine Generals Problem," ACM TOPLAS 1982.

Slide credit: Mahadev Satyanarayanan, CMU, 15-440, Spring 2015

Really Interested?

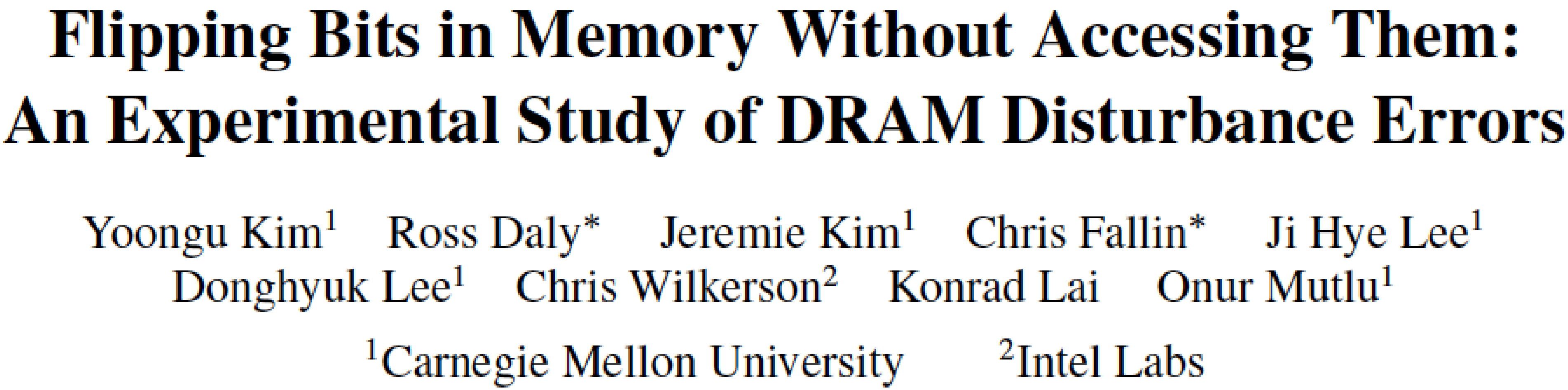
- Our first detailed study: Rowhammer analysis and solutions (June 2014)
  - Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu, **"Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors"** *Proceedings of the 41st International Symposium on Computer Architecture (**ISCA**)*, Minneapolis, MN, June 2014. [Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Source Code and Data]

- Our Source Code to Induce Errors in Modern DRAM Chips (June 2014)
  - https://github.com/CMU-SAFARI/rowhammer

- Google Project Zero's Attack to Take Over a System (March 2015)
  - Exploiting the DRAM rowhammer bug to gain kernel privileges (Seaborn+, 2015)
  - https://github.com/google/rowhammer-test
  - Double-sided Rowhammer

More on RowHammer Analysis

- Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu, **"Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors"** *Proceedings of the 41st International Symposium on Computer Architecture* (***ISCA***), Minneapolis, MN, June 2014. [Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Source Code and Data]



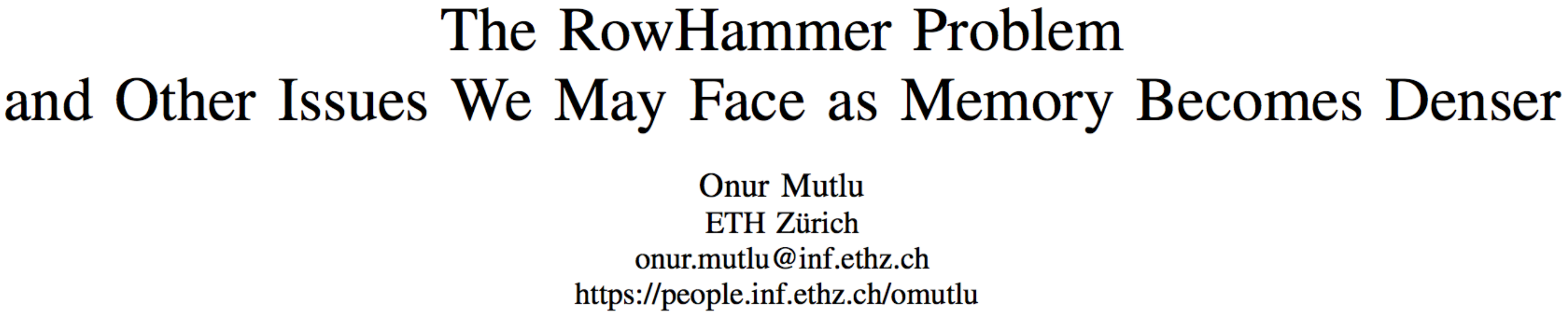
Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors

Yoongu Kim[1] Ross Daly* Jeremie Kim[1] Chris Fallin* Ji Hye Lee[1]
Donghyuk Lee[1] Chris Wilkerson[2] Konrad Lai Onur Mutlu[1]

[1]Carnegie Mellon University [2]Intel Labs

Future of Memory Reliability

- Onur Mutlu,
  "The RowHammer Problem and Other Issues We May Face as Memory Becomes Denser"
  *Invited Paper in Proceedings of the Design, Automation, and Test in Europe Conference (**DATE**)*, Lausanne, Switzerland, March 2017.
  [Slides (pptx) (pdf)]



The RowHammer Problem
and Other Issues We May Face as Memory Becomes Denser

Onur Mutlu
ETH Zürich
onur.mutlu@inf.ethz.ch
https://people.inf.ethz.ch/omutlu

https://people.inf.ethz.ch/omutlu/pub/rowhammer-and-other-memory-issues_date17.pdf

Takeaway

Breaking the abstraction layers (between components and transformation hierarchy levels)

and knowing what is underneath

enables you to **understand** and **solve** problems

Design of Digital Circuits
Lecture 2: Mysteries in Comp Arch

Prof. Onur Mutlu
ETH Zurich

Spring 2018

23 February 2018