CAN’t Touch This
Software-only Mitigation against Rowhammer Attacks targeting Kernel Memory

Ferdinand Brasser
David Gens
Christopher Liebchen
Ahmad-Reza Sadeghi

Intel Collaborative Research Institute for Secure Computing
Technische Universität Darmstadt

Lucas Davi

University of Duisburg-Essen
Big Picture: Rowhammer Attacks

Software
Big Picture: Rowhammer Attacks

Software
Big Picture: Rowhammer Attacks

Software
Big Picture: Rowhammer Attacks

Software

Access Control
Big Picture: Rowhammer Attacks

Software

Access Control
Big Picture: Rowhammer Attacks

Software
Access Control

CPU
Memory Chip

Attacker
Big Picture: Rowhammer Attacks

Software

Access Control

CPU

Memory
Big Picture: Our Approach

Software

Access Control

Spatial Isolation
Big Picture: Our Approach

Software

Access Control

Spatial Isolation
Big Picture: Our Approach

Software
Access Control
Spatial Isolation

Can't Touch This
Dynamic Random Access Memory (DRAM)

1. DIMM*
2. Rank
3. Chip (contains Banks)

*) Dual Inline Memory Module
DRAM: Storing a Single Bit

Memory Cell

Select

Transistor

Capacitor

Data

Read Bit
DRAM: Storing a Single Bit

Memory Cell

Select

Transistor

Capacitor

Data

Read Bit

1
DRAM: Storing a Single Bit

![Diagram of a DRAM memory cell showing a capacitor, a transistor, and connections to select and data lines. The cell stores a '1'.]
DRAM: Storing a Single Bit

Select

Transistor

Capacitor

Data

Memory Cell

1

Read Bit
DRAM: Organization inside a Bank

Row 1:
Column 1: [Memory Cell]
Column 2: [Memory Cell]
Column 3: [Memory Cell]
Column 4: [Memory Cell]
Column 5: [Memory Cell]
Column 6: [Memory Cell]

Row 2:

Row 3:

Row 4:

Row Buffer:

Memory Cell

CPU
DRAM: Read Access

Row 1:
Row 2: 1 0 1 1 0 0
Row 3:
Row 4:
Row Buffer:
DRAM: Read Access

High voltage on access

Row 1: [empty]
Row 2: [1 0 1 1 0 0]
Row 3: [empty]
Row 4: [empty]
Row Buffer: [empty]
DRAM: Read Access

High voltage on access

Column 1: Column 2: Column 3: Column 4: Column 5: Column 6:

Row 1: Row 2: 1 0 1 1 0 0 Row 3: Row 4: Row Buffer:
DRAM: Read Access

High voltage on access

Row 1:
Row 2: 1 0 1 1 0 0
Row 3:
Row 4:
Row Buffer: 1 0 1 1 0 0

Column 1: Column 2: Column 3: Column 4: Column 5: Column 6:
DRAM: Read Access

Row 1:
Row 2: 101100
Row 3:
Row 4:
Row Buffer: 101100

High voltage on access
DRAM: Read Access

High voltage on access

Row 1:

Row 2: 1011100

Row 3:

Row 4:

Row Buffer: 1011100

Refresh (write-back)
Testing methodology introduced by Kim et al. [ISCA 2014]

X and Y need to be on the same bank but in different rows; general pattern: $Y = X + 8\text{MB}$

```assembly
<test-rows>:
    mov  eax, (X)
    mov  ebx, (Y)
    clflush (X)
    clflush (Y)
    jmp  test-rows
```

- Read from Memory at position X and store in EAX
- Read from Memory at position Y and store in EBX
- Evict X and Y from the cache
- Repeat procedure (lots of times)
Single-Sided Rowhammer

Row 1:  
Victim Row 2:  
Aggressor X Row 3:  
Victim Row 4:  
Row 5:  
Victim Row 6:  
Aggressor Y Row 7:  
Row Buffer:  

Repeatedly activating Row 3 and 7
Single-Sided Rowhammer

Row 1:

Victim Row 2:

Aggressor X Row 3:

Victim Row 4:

Row 5:

Victim Row 6:

Aggressor Y Row 7:

Row Buffer:

Repeatedly activating Row 3 and 7
Many Bit Flips Observed

Source: Kim et al., ISCA 2014
Once it’s bad, it gets worse.
Double-Sided Rowhammer

Aggressor X: Row 2: 1 0 1 0 1 0 1 0
Victim: Row 3: 0 0 0 0 0 0 0
Aggressor Y: Row 4: 1 0 1 0 1 0 1 0
Row 5:
Row 6:
Row 7:
Row Buffer: 1 0 1 0 1 0 1 0

Repeatedly activating Row 2 and 4
Double-Sided Rowhammer

Repeatedly activating Row 2 and 4
How Dangerous are Bit Flips?
Rowhammer Timeline and Attacks

2009
- DRAM Errors in the Wild
  Schroeder et al. (SIGMETRIC)

2012
- Row hammer refresh command
  Intel Corporation (US Patent)

2014
- Flipping Bits in Memory
  Kim et al. (ISCA)

2015
- Exploiting the DRAM rowhammer bug
  Seaborn and Dullien (Black Hat US)

2016
- Drammer
  van der Veen et al. (CCS)
- Improved Rowhammer Attacks
  Qiao and Seaborn (HOST)
- Flip Feng Shui
  Razavi et al. (USENIX Sec.)
- Rowhammer.js
  Gruss et al. (DIMVA)
- One Bit Flips, One Cloud Flops
  Xiao et al. (USENIX Sec.)
Related Work: First Defenses

- **Prohibit CLFLUSH**
  - Ineffective [Qiao and Seaborn, HOST 2016]

- **Increase Refresh Interval**
  - Ineffective [Aweke et al. ASPLOS 2016]

- **Software Access Control**

- **ANVIL [Aweke et al. ASPLOS 2016]**
  - Heuristic approach (overhead & false positives)

- **Probabilistic Adjacent Row Activation [Kim et al. ISCA 2014]**
  - Modifies Hardware (costly & legacy problem)
Reviewing Attacker Assumptions

1. Vulnerable Cells
2. Co-location
Our Initial Approach:

Blacklisting
Deactivate Vulnerable Physical Memory
Initial Tests with Blacklisting

Locate Vulnerable Memory

Phy Addr 1
Phy Addr 2
...

Blacklist of vulnerable memory

Offline Analysis

Physical Memory

Kernel

Memory Allocator

List of Available Memory

Modified Bootloader

For more details check our technical report at https://arxiv.org/abs/1611.08396
Problems of Blacklisting

• Coverage

• Progression of vulnerable cells over time

• Memory overhead for other systems than our test systems unclear

https://arxiv.org/abs/1611.08396
Our Generic Approach:

CATT
Spatially Isolate Physical Memory in Software
CATT: Contributions and Challenges

• First defense that enables spatial memory isolation

• Defines and manages different security domains

• Prototype Implementation
  • CATT for the Linux kernel
  • Tested using Real-World Setup
  • Extensive Performance and Security Evaluation
CATT: Design Idea

- Separate security domains \textit{physically}
CATT: Design Idea

• Separate security domains *physically*
CATT: Design Idea

- Separate security domains *physically*
CATT: Design Idea

- Separate security domains *physically*
CATT: Design Idea

- Separate security domains *physically*
  - Attacker can still flip bits
CATT: Design Idea

• Separate security domains *physically*
  • Attacker can still flip bits
  • But only within her security domain
CATT: DRAM-aware Memory Allocation

- Rowhammer exploits physical co-location
CATT: DRAM-aware Memory Allocation

- Rowhammer exploits physical co-location
CATT: DRAM-aware Memory Allocation

- Rowhammer exploits physical co-location

If we know the mapping, we know where a Page Frame will be located in DRAM!

Physical Address Space

Page Frame
Page Frame
Page Frame
...

Page Frame

Physical to DRAM Mapping (Hash of the physical address)

Physical Memory (DRAM)

If we know the mapping, we know where a Page Frame will be located in DRAM!
CATT: Implementation

• Prototype for the Linux kernel
  • Version 4.6
  • Completely transparent to applications

• Modifies physical page allocator
  • Associates page frames with security domain
  • Adds „kernel“ zone to buddy allocator
Evaluation
System Setup

S1
i7 – Ivy Bridge
8GB DDR3

S2
i5 – Sandy Bridge
8GB DDR3

S3
i5 – Sandy Bridge (Mobile)
8GB DDR3
Security

- Tested blacklisting against previously compiled list of target rows
  - Vulnerable rows are successfully blocked by the bootloader

- Tested CATT against existing Rowhammer kernel exploits [BH15 Seaborn and Dullien]
  - Without our patch: success within minutes
  - With our patch: ran 48+ hours without success
Performance

• SPEC CPU 2006: avg. -0.5% (max 0.29%)

• Phoronix: avg. 0.27% (max. 2.49%)

• LMBench: avg. 0.11% (max. 1.66%)

• Linux Test Project: same results as vanilla kernel (contains stress tests for scheduling, memory, and file accesses)
Conclusion

• Software vulnerabilities are still the predominant attack vector
  • Continuous arms race between attacks and defenses

• Hardware reliability issues lead to severe security consequences
  • Rowhammer corrupts memory without requiring software vulnerabilities

• Good news: Promising research results and insights
  • First software-only defenses against Rowhammer have been proposed to protect legacy systems
Questions?