Microarchitectural Vulnerabilities - Assessment and Mitigation
Maria MUSHTAQ, Safe and Secure Hardware (SSH), Communications and Electronics (COMelec)

**About me!**

**Distinctions:**
- Portrait of Woman -International face of university @ UBS, France
- Recipient of HiPEAC International Mobility –Yale University, USA
- Recipient of ACM Young Researcher Award –ETH Zurich, Switzerland
- Recipient of Ministry of Defense scholarship
- Recipient of CNRS-Excellent Postdoc grant
- Admissible candidate by the academic jury of CNRS, 2020-2021
Outline

- Information Security Perspective
- Detection Framework
- Mitigation Framework
- Conclusions & Future Perspectives
In May 2020, the NSA discovered that Russian hackers were stealing sensitive data from American organizations, through a bug in a popular email server.

In December 2020, one of the biggest media organizations in Germany, the Frankfurter Allgemeine, fell victim to a ransomware attack. Roughly 6,000 computers were infected, which halted work at the company’s editorial offices, as well as some of its major printing houses.

In June 2020, malware was used to commit cyber attacks against 9 Indian human rights activists. Their keystrokes were logged, audio was recorded, and their personal credentials were stolen.

2018 was the worst year for cyber attacks in America, with 30 incidents in that year alone.

Information Security Perspective

- A shared concern by many application domains
Information Security Perspective

Computing Stack & Privilege Levels

- Information leakage is possible *even under safe* software!
  - Software is often encrypted by mathematically strong encryption techniques [RSA, AES, ECC etc.]

- Underlying *hardware is vulnerable*
  - Micro-architectural features leak information on the state of program’s execution
Information Security Perspective

Encryption Operation

- Data/Instruction Timing
- Power
- Data/Instruction Access
- Electromagnetic Radiation
- Other Side-Channels
- Acoustic Emanation

Input
Key + Plain Text

Output
Cipher Text
Information Security Perspective

- Shared Memory Architecture – An Abstract View!
Information Security Perspective

- Leakage through Shared Memory

Key-dependent memory accesses create timing (Side-Channel) Information!
Information Security Perspective

Leakage Through Computational Optimizations

Branch Prediction
**Information Security Perspective**

**Intel’s x86—the biggest casualty of security vulnerabilities!**
Information Security Perspective

- Threat Model – Why CSCAs are interesting?
  - CSCAs are Non-invasive, Passive & High-resolution
  - CSCAs do not respect privileges
    - They are Cross CPU, Cross Core, Cross VM
  - All CSCAs [and other attacks too] work the same way:
    - Manipulate cache to a known state
    - Wait for the victim to perform its activity
    - Examine what has changed

- Hard to detect as they are part of the hardware design!

---

Mushtaq et al., Winter is here! A Decade of Cache-based Software Side-Channel Attacks & Mitigation Techniques. Published at Elsevier Information Systems 2020.
Information Security Perspective

- **Prime+Probe Attack**

  - **Prime Phase**: Attacker primes the cache.
  - **Victim Access**: Victim’s execution.
  - **Probe Phase**: Attacker probes the cache state.
Information Security Perspective

State-of-the-Art on Defenses
Information Security Perspective

State-of-the-Art on Defenses

What is Missing?

- Mitigations are Vulnerability Specific
- Mitigations are not System-wide
- Mitigations are Performance Heavy

The Way Forward:

- Attack surface is not completely known yet – rather expanding!
- Security paradigm is shifting
- Secure-by-Design — attacks are not feasible in the first place
- Secure at Run-time — attacks can happen, but their impact & value is contained
Research Perspective

The Big Picture

Detection-based Mitigation

Attack Vector Development & Analysis

Run-time Detection

Run-time Mitigation

Use-case Cache Side-Channel Attacks

Mushtaq et al., Improving Confidentiality Against Cache-based SCAs. Published at Conference of ACM WomENCourage-2017, Barcelona, Spain.
Detection Framework

- Cache SCAs affect or alter cache behavior!

- The Victim's Perspective
## Detection Framework

### Performance Counters as features

<table>
<thead>
<tr>
<th>#</th>
<th>Scope</th>
<th>Hardware Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cache Level 1</td>
<td>Data Cache Misses (L1-DCM)</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>Instruction Cache misses (L1-ICM)</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>Total cache misses (L1-TCM)</td>
</tr>
<tr>
<td>4</td>
<td>Cache Level 2</td>
<td>Instruction cache accesses (L2-ICA)</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>Instruction Cache misses (L2-ICM)</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>Total Cache accesses (L2-TCA)</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>Total cache misses (L2-TCM)</td>
</tr>
<tr>
<td>8</td>
<td>Cache Level 3</td>
<td>Instruction cache accesses (L3-ICA)</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>Total Cache accesses (L3-TCA)</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>Total cache misses (L3-TCM)</td>
</tr>
<tr>
<td>11</td>
<td>System-wide</td>
<td>Branch Miss Prediction (BR_MSP)</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>Total CPU Cycles (TOT_CYC)</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>Total Page Faults (Page-Faults)</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>Total Number of Instructions (TOT_INS)</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>Total Branch Instructions (BR_INS)</td>
</tr>
</tbody>
</table>
Detection Framework

- Performance Counters – Prime+Probe Attack

Detection Framework

Detection-based Mitigation

Attack Vector Development & Analysis

Run-time Detection

Run-time Mitigation

Use-case Cache Side-Channel Attacks
Detection Framework

- Performance Counters

Detection Framework

- Performance Counters

Machine Learning Can Help!

## Detection Framework

### Machine Learning Models

<table>
<thead>
<tr>
<th>#</th>
<th>Machine Learning Model</th>
<th>Type of Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Linear Regression (LR)</td>
<td>Linear</td>
</tr>
<tr>
<td>2</td>
<td>Linear Discriminant Analysis (LDA)</td>
<td>Linear</td>
</tr>
<tr>
<td>3</td>
<td>Linear Support Vector Machine (SVM)</td>
<td>Linear</td>
</tr>
<tr>
<td>4</td>
<td>Quadratic Discriminant Analysis (QDA)</td>
<td>Linear</td>
</tr>
<tr>
<td>5</td>
<td>Nearest Centroid</td>
<td>Linear</td>
</tr>
<tr>
<td>6</td>
<td>Naïve Bayes</td>
<td>Linear</td>
</tr>
<tr>
<td>7</td>
<td>K-Nearest Neighbors (KNN)</td>
<td>Non-Linear</td>
</tr>
<tr>
<td>8</td>
<td>Perceptron</td>
<td>Non-Linear</td>
</tr>
<tr>
<td>9</td>
<td>Decision Tree</td>
<td>Non-Linear</td>
</tr>
<tr>
<td>10</td>
<td>Dummy</td>
<td>Non-Linear</td>
</tr>
<tr>
<td>11</td>
<td>Random Forest (RF)</td>
<td>Non-Linear</td>
</tr>
<tr>
<td>12</td>
<td>Convolutional Neural Networks (CNNs)</td>
<td>Non-Linear</td>
</tr>
</tbody>
</table>

[https://scikit-learn.org/0.17/modules/classes.html](https://scikit-learn.org/0.17/modules/classes.html)
Detection Framework

- Cache SCA Detection

Detection Framework

Proposed Framework – The Big Picture

Detection-based Mitigation

- Attack Vector Development & Analysis
- Run-time Detection
- Run-time Mitigation

Use-case Cache Side-Channel Attacks
## Use-case Attacks

<table>
<thead>
<tr>
<th>No.</th>
<th>Use-cases</th>
<th>Cryptosystem</th>
<th>OpenSSL Version</th>
<th>Key Recovery</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Flush+Reload</td>
<td>RSA</td>
<td>0.9.7l</td>
<td>Full Key</td>
</tr>
<tr>
<td>2</td>
<td>Flush+Reload</td>
<td>AES</td>
<td>0.9.7l/1.0.1f</td>
<td>Half Key</td>
</tr>
<tr>
<td>3</td>
<td>Flush+Reload</td>
<td>AES</td>
<td></td>
<td>Full Key</td>
</tr>
<tr>
<td>4</td>
<td>Flush+Flush</td>
<td>AES</td>
<td>0.9.7l/1.0.1f</td>
<td>Half Key</td>
</tr>
<tr>
<td>5</td>
<td>Flush+Flush</td>
<td>AES</td>
<td></td>
<td>Full Key</td>
</tr>
<tr>
<td>6</td>
<td>Prime+Probe</td>
<td>AES</td>
<td></td>
<td>Half Key</td>
</tr>
<tr>
<td>7</td>
<td>Prime+Probe</td>
<td>AES</td>
<td></td>
<td>Full Key</td>
</tr>
<tr>
<td>8</td>
<td>Spectre</td>
<td>Not crypto-specific</td>
<td>Linux Kernel 4.13.037</td>
<td>Full message exploitation</td>
</tr>
<tr>
<td>9</td>
<td>Meltdown</td>
<td>Not crypto-specific</td>
<td>Linux Kernel 4.13.037</td>
<td>Full message exploitation</td>
</tr>
</tbody>
</table>

Open source repository of our work: [https://github.com/ECLab-ITU/Cache-Side-Channel-Attacks](https://github.com/ECLab-ITU/Cache-Side-Channel-Attacks)
### Detection Framework

#### F+R Attack on RSA Cryptosystem

<table>
<thead>
<tr>
<th>Model</th>
<th>Loads</th>
<th>Accuracy (%)</th>
<th>Speed (%)</th>
<th>FP (%)</th>
<th>FN (%)</th>
<th>Overhead (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA</td>
<td>ZL</td>
<td>99.5</td>
<td>0.9</td>
<td>.498</td>
<td>.002</td>
<td>0.9</td>
</tr>
<tr>
<td></td>
<td>ML</td>
<td>99.5</td>
<td>0.9</td>
<td>0.49</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HL</td>
<td>99.4</td>
<td>0.9</td>
<td>.527</td>
<td>.073</td>
<td></td>
</tr>
<tr>
<td>LR</td>
<td>ZL</td>
<td>99.5</td>
<td>0.9</td>
<td>0.5</td>
<td>0</td>
<td>1.6</td>
</tr>
<tr>
<td></td>
<td>ML</td>
<td>99.5</td>
<td>0.9</td>
<td>.494</td>
<td>.006</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HL</td>
<td>99.5</td>
<td>0.9</td>
<td>.462</td>
<td>.038</td>
<td></td>
</tr>
<tr>
<td>SVM</td>
<td>ZL</td>
<td>98.8</td>
<td>0.9</td>
<td>0.4</td>
<td>.78</td>
<td>1.3</td>
</tr>
<tr>
<td></td>
<td>ML</td>
<td>90</td>
<td>0.9</td>
<td>0.17</td>
<td>9.83</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HL</td>
<td>95.8</td>
<td>0.9</td>
<td>3.21</td>
<td>.99</td>
<td></td>
</tr>
<tr>
<td>QDA</td>
<td>ZL</td>
<td>99.5</td>
<td>0.9</td>
<td>0.5</td>
<td>0</td>
<td>0.6</td>
</tr>
<tr>
<td></td>
<td>ML</td>
<td>99.5</td>
<td>0.9</td>
<td>.494</td>
<td>.006</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HL</td>
<td>99.4</td>
<td>0.9</td>
<td>0.57</td>
<td>.03</td>
<td></td>
</tr>
</tbody>
</table>

Mushtaq et al., NIGHTS-WATCH: A Cache-Based Side-Channel Intrusion Detector using Hardware Performance Counters. Published at ISCA-HASP, Los Angeles, CA, USA, 2018.

Mushtaq et al., Sherlock Holmes of Cache Side-Channel Attacks in Intel’s x86 Architecture. Accepted at IEEE Conference on Communications and Network Security (CNS), Washington, USA, 2019
Detection Framework

- Computational Attacks
• Two CPU vulnerabilities discovered in 2018!
• Both exploit performance enhancement techniques
Detection Framework

- Meltdown
  - Vulnerability: Permission check for address is done in parallel & out-of-order to the load instruction!

![Diagram of Detection Framework](image)

- Access to Kernel memory
  - Access Permission Check: If permission check fails, interrupt the O-o-O execution.
  - Race Condition: Execution results are discarded.

Figure: Courtesy Daniel Gruss et al.
Detection Framework

- Meltdown Detection
  - Selected HPCs & SPCs

<table>
<thead>
<tr>
<th>Scope of event</th>
<th>Hardware event</th>
<th>Feature ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>L3 cache</td>
<td>Total cache misses</td>
<td>L3_TCM</td>
</tr>
<tr>
<td>L3 cache</td>
<td>Total cache accesses</td>
<td>L3_TCA</td>
</tr>
<tr>
<td>System wide</td>
<td>Total page faults</td>
<td>page_faults</td>
</tr>
<tr>
<td>System wide</td>
<td>Total number of instructions</td>
<td>TOT_INS</td>
</tr>
</tbody>
</table>

![Graph showing total instructions and page faults over time](image)
Detection Framework

- Meltdown Detection

<table>
<thead>
<tr>
<th>Model</th>
<th>Load</th>
<th>Accuracy (%)</th>
<th>Speed (μs)</th>
<th>FP (%)</th>
<th>FN(%)</th>
<th>Overhead (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA</td>
<td>NL</td>
<td>99.99</td>
<td>10</td>
<td>0.01</td>
<td>0</td>
<td>1.91</td>
</tr>
<tr>
<td></td>
<td>AL</td>
<td>99.91</td>
<td>10</td>
<td>0.09</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FL</td>
<td>98.30</td>
<td>10</td>
<td>1.25</td>
<td>0.45</td>
<td></td>
</tr>
<tr>
<td>LR</td>
<td>NL</td>
<td>99.41</td>
<td>10</td>
<td>0.59</td>
<td>0</td>
<td>2.21</td>
</tr>
<tr>
<td></td>
<td>AL</td>
<td>97.45</td>
<td>10</td>
<td>1.95</td>
<td>0.60</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FL</td>
<td>96.00</td>
<td>10</td>
<td>3.40</td>
<td>1.60</td>
<td></td>
</tr>
<tr>
<td>SVM</td>
<td>NL</td>
<td>99.99</td>
<td>10</td>
<td>0.01</td>
<td>0</td>
<td>2.00</td>
</tr>
<tr>
<td></td>
<td>AL</td>
<td>99.40</td>
<td>10</td>
<td>0.60</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FL</td>
<td>98.35</td>
<td>10</td>
<td>1.39</td>
<td>0.26</td>
<td></td>
</tr>
</tbody>
</table>

Mushtaq et al., Transit-Guard: An OS-based Defense Mechanism Against Transient Execution Attacks, Published at IEEE European Test Symposium, 2021.
Mitigation Framework

- Proposed Framework –The Big Picture

Detection-based Mitigation

- Attack Vector Development & Analysis
- Run-time Detection
- Run-time Mitigation

Use-case Cache Side-Channel Attacks
Mitigation Framework

- Simultaneous Attacks, Detection and Mitigation
Mitigation Framework

- Detection-based Protection under Linux
Mitigation Framework

Detection-based Mitigation of F+R Attack on RSA

Mushtaq et al., The Kingsguard: OS-level mitigation against cache-channel attacks using run-time detection, Published at IEEE-Access, 2022.
Outline

• Information Security Perspective
• Detection Framework
• Mitigation Framework
• Conclusions & Future Perspectives
Conclusions –at large

- Side channel information leakage is powerful & attack surface is expanding

- Need-based protection has the potential to contain SCAs, both computational & storage, while retaining the performance benefits

- Detection is promising –can serve as the first line of defense in the absence of secure-by-design solutions

- Machine learning can help improving security –use of specialized ML models and deep learning
Future Perspectives

Information Retrieval Attacks

- Software on Hardware
- Software on Software
- Hardware on Hardware
- Hardware on Software

Trusted Computing Base (TCB) [Hardware/Software]

Protected Software

Targets for Attacker

Attack Surface

- Software Internal
- Software External
- Hardware Internal
- Hardware External
Future Perspectives

- Security has become a 1st class design constraint – computing must be seen beyond classics.

- Modern security challenges emerge from the way we compute today – radical changes at both the hardware & software levels are required.

- No computing platform is secure today and attack surface will expand further – tools are required to contain existing vulnerabilities and future systems must be predictable!
Research Activities

Automated Vulnerability Assessment
OS/Hypervisor Security
Re-defining ISA

Forcioli et al., Virtual Platform to Analyze Security of a System on Chip at Microarchitectural level, Published SILM, European Symposium on Security and Privacy Workshop, 2021
Hamza et al., Diminisher: A Linux Kernel based Countermeasure for TAA Vulnerability, European Symposium on Research in Computer Security, 2021
France et al., Vulnerability Assessment of the Rowhammer Attack Using Machine Learning and the gem5 Simulator, at ACM workshop on secure and trustworthy cyber-physical systems, 2021
France et al., Implementing Rowhammer Memory Corruption in the gem5 Simulator, at Workshop on Rapid System Prototyping (RSP), 2021
Khatib et al., Unsupervised Network Intrusion Detection System for AVTP in Automotive Ethernet Networks, at IEEE Intelligent Vehicles Symposium, 2022
Awareness Seminar

IP Paris & Telecom’s 1st International Winter School on Microarchitectural Security – 5-9th of December 2022

International Winter School on Microarchitectural Security 2022

The International Winter School on Microarchitectural Security (Mic-Sec) offers academic and industrial talks along with hands-on experience on attacks, software and hardware countermeasure techniques with a special focus on side-channel attacks. The Mic-Sec Winter School 2022 edition will take place at the FIAP Paris from the 5th to the 9th of December 2022 in Paris, France.

Thank You!

[Discussion]

Maria.Mushtaq@telecom-paris.fr
@Maria_Mushtaq_