Micro-Architectural Attacks on Cyber-Physical Systems

07/24/2019
Heechul Yun
University of Kansas
Modern Cyber-Physical Systems

• Cyber Physical Systems (CPS)
  – Cyber (Computer) + Physical (Plant)

• Real-time
  – Control physical process in real-time

• Safety-critical
  – Can harm people/things

• Intelligent
  – Can function autonomously
Modern System-on-a-Chip (SoC)

- Integrate multiple cores, GPU, accelerators
- Good performance, size, weight, power
- Introduce new challenges in real-time, security
Micro-Architectural Attacks

• Micro-architectural hardware components
  – E.g., cache, tlb, DRAM, OoO engine, MSHRs, ...

• Can affect execution timing
  – E.g., delay critical real-time tasks

• Can leak secret
  – E.g., Meltdown, Spectre

• Can alter data
  – E.g., RowHammer
1. Denial-of-Service Attacks

- Attacker’s goal: increase the victim’s task **execution time**
- The attacker is on different core/memory/cache partition
- The attacker can only execute non-privileged code.

Non-Blocking Cache

- We identified cache internal structures that are potential DoS attack vectors

Miss Status Holding Registers\(^1\)
- Track outstanding cache misses.

Writeback Buffer\(^2\)
- Holds evicted dirty lines (writebacks).
- Prevents cache refills from waiting.

\(^2\) M. G. Bechtel and H. Yun. “Denial-of-Service Attacks on Shared Cache in Multicore: Analysis and Prevention.” In RTAS, 2019
Cache DoS Attacks

```
for (i = 0; i < mem_size; i += LINE_SIZE)
{
    sum += ptr[i];
}

Read Attacker
(target MSHRs)
```

```
for (i = 0; i < mem_size; i += LINE_SIZE)
{
    ptr[i] = 0xff;
}

Write Attacker
(target WBBBuffer)
```

- Denial-of-Service (DoS) attacks targeting internal hardware structures of a shared cache.
  - Block the cache ➔ delay the victim’s execution time

Effects of Cache DoS Attacks

- Observed worst-case: >300X (times) slowdown
  - On popular in-order multicore processors
  - Due to contention in cache write-back buffer

DeepPicar

- A **low cost**, small scale replication of NVIDIA’s DAVE-2
- Uses the exact same DNN
- Runs on a Raspberry Pi 3 in **real-time**

<table>
<thead>
<tr>
<th>Item</th>
<th>Cost ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raspberry Pi 3 Model B</td>
<td>35</td>
</tr>
<tr>
<td>New Bright 1:24 scale RC car</td>
<td>10</td>
</tr>
<tr>
<td>Playstation Eye camera</td>
<td>7</td>
</tr>
<tr>
<td>Pololu DRV8835 motor hat</td>
<td>8</td>
</tr>
<tr>
<td>External battery pack &amp; misc.</td>
<td>10</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>70</strong></td>
</tr>
</tbody>
</table>

output: steering angle
- fc4: fully-connected layer
- fc3: fully-connected layer
- fc2: fully-connected layer
- fc1: fully-connected layer
- conv5: 64@1x18 convolutional layer
- conv4: 64@3x2 convolutional layer
- conv3: 48@5x22 convolutional layer
- conv2: 36@14x47 convolutional layer
- conv1: 24@31x98 convolutional layer
- input: 200x66 RGB pixels

https://github.com/mbechtel2/DeepPicar-v2
Experiment Setup

- DNN control task of DeepPicar (real-world RT)
- IsolBench BwWrite benchmark (synthetic RT)
- Parboil benchmarks (real-world BE)

<table>
<thead>
<tr>
<th>Task</th>
<th>WCET (C ms)</th>
<th>Period (P ms)</th>
<th># Threads</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{dnn}^{rt}$</td>
<td>34</td>
<td>100</td>
<td>2</td>
</tr>
<tr>
<td>$t_{bww}^{rt}$</td>
<td>220</td>
<td>340</td>
<td>2</td>
</tr>
<tr>
<td>$t_{cutcp}^{be}$</td>
<td>$\infty$</td>
<td>N/A</td>
<td>4</td>
</tr>
<tr>
<td>$t_{lbm}^{be}$</td>
<td>$\infty$</td>
<td>N/A</td>
<td>4</td>
</tr>
</tbody>
</table>

Effect of Co-Scheduling

https://youtu.be/Jm6KSDqlqiU
2. Speculative Execution Attacks

- Attacks exploiting microarchitectural side-effects of executing speculative (transient) instructions
- Many variants

No hardware support planned in near future

<table>
<thead>
<tr>
<th>Attack</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variant 1 (Spectre) [16]</td>
<td>Bounds Check Bypass</td>
</tr>
<tr>
<td>Variant 1.1 [15]</td>
<td>Bounds Check Bypass Store</td>
</tr>
<tr>
<td>Variant 1.2 [15]</td>
<td>Read-only Protection Bypass</td>
</tr>
<tr>
<td>Variant 2 (Spectre) [16]</td>
<td>Branch Target Injection</td>
</tr>
<tr>
<td>Variant 3 (Meltdown) [18]</td>
<td>Supervisor Protection Bypass</td>
</tr>
<tr>
<td>Variant 3a [12]</td>
<td>System Register Bypass</td>
</tr>
<tr>
<td>Lazy FP [24]</td>
<td>FPU Register Bypass</td>
</tr>
<tr>
<td>Variant 4 [9]</td>
<td>Speculative Store Bypass</td>
</tr>
<tr>
<td>ret2spec [20]</td>
<td>Return Stack Buffer</td>
</tr>
<tr>
<td>L1 Terminal Fault [11, 26]</td>
<td>Virtual Translation Bypass</td>
</tr>
</tbody>
</table>

Spectre Attack (Variant 1)

```c
if(x < array1_length){
    val = array1[x];
    tmp = array2[val*512];
}
```

• Assume \( x \) is under the attacker’s control
• Attacker trains the branch predictor to predict the branch is in-bound

Spectre Attack (Variant 1)

if(x < array1_length){
    val = array1[x];
    tmp = array2[val*512];
}

1. [ACCESS]

• Speculative execution of the first line accesses the secret (val)

Spectre Attack (Variant 1)

```java
if(x < array1_length) {
    val = array1[x];
    tmp = array2[val*512];
}
```

2. [TRANSMIT]

- Speculative execution of the second, secret dependent load **transmits** the secret to a microarchitectural state (e.g., cache)

if(x < array1_length){
    val = array1[x];
    tmp = array2[val*512];
}

3. [RECEIVE]

- Attacker receives the secret by measuring timing differences (cache hit vs. miss) among the elements in the probe array.
Cache Timing Channels

• Leak secret via timing differences
  – Fast (cache-hit): victim accessed it
  – Slow (cache-miss): victim didn’t access it.
• Methods: Flush+Reload, Prime+Probe, etc.

3. RowHammer Attacks

- Repeatedly opening and closing a DRAM row can induces **bit flips** in adjacent rows storing sensitive data (e.g., page table)

Credit: This slide is from Dr. Yoongu Kim’s presentation slides of the following paper: “Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors,” In ISCA, 2014
Isolation

• Traditionally about memory isolation
  – Prevent unauthorized access to memory
  – Hardware support: MPU, MMU

• What we need
  – Prevent influence between domains
  – Not only for real-time systems
  – But also for security

• What hardware architecture/OS do we need?

1 Q Ge, Y Yarom, T Chothia, G Heiser. “Time Protection: the Missing OS Abstraction”. In *EuroSys*, 2019
Real-Time **AND** Real-Fast

- Strong isolation **AND** high performance
How?

• Embrace complexity for high performance
  – Non-blocking cache, prefetcher, out-of-order execution engine, split-transaction bus, ...

• Cross-layer OS/HW collaborative approach
  – Need to re-think existing abstractions
  – Need new SW/HW contracts to reason and control all things that affect timing
Deterministic Memory

- Declare all or part of address space as deterministic memory
- DM-aware **end-to-end resource management**

**Application view (logical)**

**System-level view (physical)**

**Data-centric cross-layer approach for real-time**

SpectreGuard

• Step 1: Software tells OS what **data** is secret
• Step 2: OS updates the page table entries
• Step 3: Load of the secret data is identified by MMU
• Step 4: secret data forwarding is **delayed** until safe

---

Data-centric cross-layer approach for **security**

---

J. Fustos, F. Farshchi, H. Yun. “SpectreGuard: An Efficient Data-centric Defense Mechanism against Spectre Attacks.” In DAC, 2019
RISC-V + NVDLA SoC Platform

- Full-featured quad-core SoC with hardware DNN accelerator on Amazon FPGA cloud

Open-source hardware: big research opportunity!

F. Farshchi, Q. Huang, H. Yun. “Integrating NVIDIA Deep Learning Accelerator (NVDLA) with RISC-V SoC on FireSim.” In EMC², 2019
RT-Gang

- One parallel real-time task---a gang---at a time
  - Eliminate inter-task interference by construction
- Schedule best-effort tasks during slacks \textit{w/ throttling}

OS can do a lot more on COTS hardware

RT-Gang

https://youtu.be/pk0j063cUAs
Conclusion

• Micro-architectural attacks are a serious threat for intelligent CPS
  – Can leak secret (confidentiality)
  – Can alter data (integrity)
  – Can affect real-time performance (correctness)

• We need **better computing infrastructure** for safe, secure, and intelligent CPS
  – And we can **build** one
Thank You!

Acknowledgement:
This research is supported by NSA Science of Security initiative contract #H98230-18-D-0009 and NSF CNS 1718880, 1815959.
Recent Publications


Full List: [http://www.ittc.ku.edu/~heechul/pub.html](http://www.ittc.ku.edu/~heechul/pub.html)