FetchBench

Systematic Identification and Characterization of Proprietary Prefetchers

Till Schlüter, Amit Choudhari, Lorenz Hetterich, Leon Trampert, Hamed Nemati, Ahmad Ibrahim, Michael Schwarz, Christian Rossow, Nils Ole Tippenhauer

ACM CCS 2023 · November 27, 2023
Motivation

Hardware Prefetching
Recap: Stride Prefetching

```
Program

i = 0..6
access(data1[i * 3])

Prefetcher

Memory
data1
```

Why Do We Care?

- Prefetchers can enable for security-critical side channels\(^1,2,3\)
- Implementations are proprietary, undocumented, diverse
- In order to get a better picture of overall security, we need to understand prefetchers better

---

\(^1\) Shin et al., “Unveiling Hardware-Based Data Prefetcher, a Hidden Source of Information Leakage”, CCS ’18.
\(^3\) Chen, Pei, and Carlson, “AfterImage: Leaking Control Flow Data and Tracking Load Operations via the Hardware Prefetcher”, ASPLOS ’23.
Research Questions

- How to identify and characterize prefetchers?
- What prefetchers are commonly implemented?
- What are the security implications?
Research Questions

- How to identify and characterize prefetchers?
- What prefetchers are commonly implemented?
- What are the security implications?
Systematization of Prefetching Approaches: Our Taxonomy

Hardware Prefetcher

Prediction Strategy
- Extrapolation
- Replay

Source
- Location
  - Stride
  - Adjacent Cache Line
- Content
  - Stream
  - Pointer Array
  - Pointer Chase
- Location
  - Spatial Memory Streaming
  - Region-Unbounded Replay

Content
Our Framework: FetchBench

**Identification and Characterization**
- Processor-Specific Prefetcher Existence and Characteristics Information
- Architecture-Specific
  - Timing
  - Flushing
  - Barriers
- Pefetcher Design-Specific
- Test Cases
- Test Case Execution

Result: Processor-Specific Prefetcher Existence and Characteristics Information
Research Questions

How to identify and characterize prefetchers?

What prefetchers are commonly implemented?

What are the security implications?
Motivation

Identification and Characterization of Prefetchers

Identification and Characterization Results

Security Implications

Conclusion

CPUs Under Test

- **AMD (4)**
  - AMD Ryzen 5 2500U (Zen)
  - AMD Ryzen 5 3550H (Zen+)
  - AMD Ryzen 7 5700G (Zen 3)
- **Apple (1)**
  - Apple M1 Max
- **Arm Cortex (5)**
  - Amlogic S905X3 (Arm A55)
  - HiSilicon Kirin 960 (Arm A73)
  - Broadcom BCM2837 (Arm A53)
  - Broadcom BCM2711 (Arm A72)
- **Intel (9)**
  - Intel Core i7-1165G7 (Tiger Lake)
  - Intel Core i7-10510U (Comet Lake)
  - Intel Xeon E-2176M (Coffee Lake)
  - Intel Core i5-4300M (Haswell)
  - Intel Xeon E3-505Mv5 (Skylake)
  - Intel Xeon Gold 6346 (Ice Lake)
  - Intel Core i3-1005G1 (Ice Lake)
  - Intel Core i7-2620M (Sandy Bridge)
  - Intel Xeon E3-1231v5 (Haswell)
  - Intel Xeon E-2176M (Coffee Lake)
  - Intel Core i5-4300M (Haswell)
  - Intel Xeon Gold 6346 (Ice Lake)
  - Intel Core i7-1165G7 (Tiger Lake)

Total: 19 CPUs/SoCs
Notable Findings

- At least 1, at most 3 data prefetchers
- **Most common:** Stride
- **New:** SMS (Spatial Memory Streaming)
- Complexity grows with CPU complexity and over time
4.2 Experimental Setup

We ran FetchBench on 19 different processors in total, comprising six ARM64 SoCs, nine Intel x86-64 CPUs, and four AMD Ryzen CPUs. We provide a list of all testing environments in Table 2 in the appendix, where we also assign them short IDs to refer to them throughout the paper. Our selection of ARM-based platforms comprises five Cortex-A-series designs, ranging from the Cortex-A53 in the Cortex-A77, as well as the low-energy and performance variants of the Apple M1 Max SoC (dubbed Western Force) and Firestorm.
Research Questions

- How to identify and characterize prefetchers?
- What prefetchers are commonly implemented?
- What are the security implications?
Exploiting Spatial Memory Streaming (SMS)

Motivation
Identification and Characterization of Prefetchers
Identification and Characterization Results
Security Implications
Conclusion

Program A

2 4 1 3
LD LD LD LD

Region A

Program B

PF PF LF PF

Region B

Memory
Case Studies

**Userspace → Userspace**

- **AES key**
- **plaintexts**
- **Prefetcher**
- **Lookup Tables (private)**
- **Attacker**
- **triggers**
- **Memory**

**Secure World → Normal World**

- **EL3 Secure Monitor**
- **EL2 Hypervisor**
- **EL1 OS Kernel**
- **EL0 Trusted OS**

- **Prefetcher**
- **CVE-2023-33936**
**Motivation**

Identification and Characterization of Prefetchers

Identification and Characterization Results

Security Implications

**Conclusion**

### Taxonomy & FetchBench

- **Hardware Prefetcher**
- **Replay**
- **Extrapolation**
- **Location**
- **Content**

**Spatial Memory Streaming**

Region-Unbounded Replay

Pointer Chase

Stream

Adjacent Cache Line

Stride

### CPU Characterization

<table>
<thead>
<tr>
<th>Year</th>
<th>Processor</th>
<th>Model</th>
<th>Architecture</th>
<th>Technology</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010</td>
<td>Amlogic</td>
<td>S905X3</td>
<td>Arm A55</td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td>Broadcom</td>
<td>BCM2837</td>
<td>Arm A53</td>
<td></td>
</tr>
<tr>
<td>2012</td>
<td>Broadcom</td>
<td>BCM2711</td>
<td>Arm A72</td>
<td></td>
</tr>
<tr>
<td>2013</td>
<td>HiSilicon</td>
<td>Kirin 960</td>
<td>Arm A73</td>
<td></td>
</tr>
<tr>
<td>2014</td>
<td>Rockchip</td>
<td>RK3588S</td>
<td>Arm A76</td>
<td></td>
</tr>
<tr>
<td>2015</td>
<td>Apple</td>
<td>M1</td>
<td>Arm</td>
<td></td>
</tr>
<tr>
<td>2016</td>
<td>Intel Core</td>
<td>i7-2620M</td>
<td>Sandy Bridge</td>
<td></td>
</tr>
<tr>
<td>2017</td>
<td>Intel Xeon</td>
<td>E3-505Mv5</td>
<td>Skylake</td>
<td></td>
</tr>
<tr>
<td>2018</td>
<td>Intel Core</td>
<td>i5-4300M</td>
<td>Haswell</td>
<td></td>
</tr>
<tr>
<td>2019</td>
<td>Intel Core</td>
<td>i7-10510U</td>
<td>Comet Lake</td>
<td></td>
</tr>
<tr>
<td>2020</td>
<td>Intel Core</td>
<td>i9-12900K</td>
<td>Alder Lake</td>
<td></td>
</tr>
<tr>
<td>2021</td>
<td>AMD Ryzen</td>
<td>5 2500G</td>
<td>Zen</td>
<td></td>
</tr>
<tr>
<td>2022</td>
<td>AMD Ryzen</td>
<td>5 3550H</td>
<td>Zen+</td>
<td></td>
</tr>
<tr>
<td>2023</td>
<td>AMD Ryzen</td>
<td>7 5700G</td>
<td>Zen 3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AMD Ryzen</td>
<td>9 6900HX</td>
<td>Zen 3+</td>
<td></td>
</tr>
</tbody>
</table>

**Stride,** **SMS**

**P:** Stride, **E:** Adjacent, **Stream,** **Adjacent**

<table>
<thead>
<tr>
<th>Year</th>
<th>Processor</th>
<th>Model</th>
<th>Architecture</th>
<th>Technology</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010</td>
<td>AMD Ryzen</td>
<td>5 2500G</td>
<td>Zen</td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td>AMD Ryzen</td>
<td>5 3550H</td>
<td>Zen+</td>
<td></td>
</tr>
<tr>
<td>2012</td>
<td>AMD Ryzen</td>
<td>7 5700G</td>
<td>Zen 3</td>
<td></td>
</tr>
<tr>
<td>2013</td>
<td>AMD Ryzen</td>
<td>9 6900HX</td>
<td>Zen 3+</td>
<td></td>
</tr>
</tbody>
</table>

**Contact me:** Till Schlüter

- [till.schlueter@cispa.de](mailto:till.schlueter@cispa.de)
- tschlueter.com

(List of all resources related to this paper)

© Till Schlüter et al. 2023 | FetchBench: Systematic Identification and Characterization of Proprietary Prefetchers | ACM CCS 2023 - 2023-11-27 | 16/16
AES Case Study: Synchronization

Diagram showing the synchronization process with the following steps:

1. attacker/sleeper
2. victim
3. attacker/main
4. attacker/sleeper
5. victim
6. attacker/sleeper
7. victim

Time line with events labeled 1 to 7, showing the sequence of operations involving victim and attacker processes.
Discussion

Limitations

- We only find prefetcher designs that we have tests for
- We only consider prefetching on data loads

Countermeasures

- Segmentation of the prefetcher’s state
- Avoid collisions in the prefetcher’s state
- Constant-time programming
References

