CISPA
HELMHOLTZ CENTER FOR
INFORMATION SECURITY

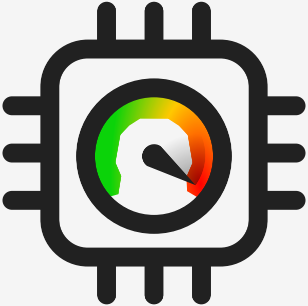
FetchBench

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

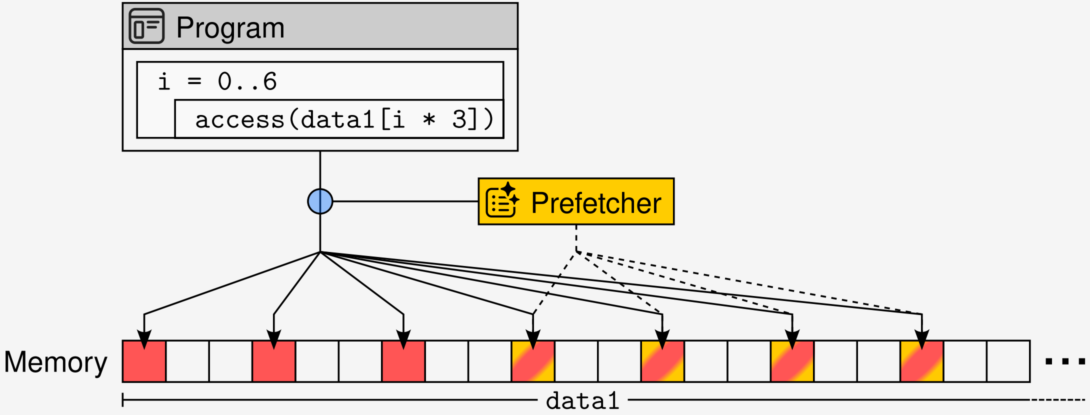
ACM CCS 2023 · November 27, 2023

Motivation

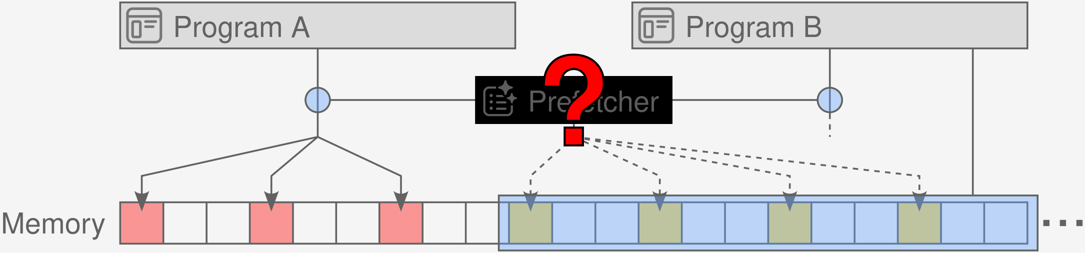


Hardware Prefetching

Recap: Stride Prefetching



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.

[2] Sanchez Vicarte et al., “Augury: Using Data Memory-Dependent Prefetchers to Leak Data at Rest”, S&P ’22.

[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?

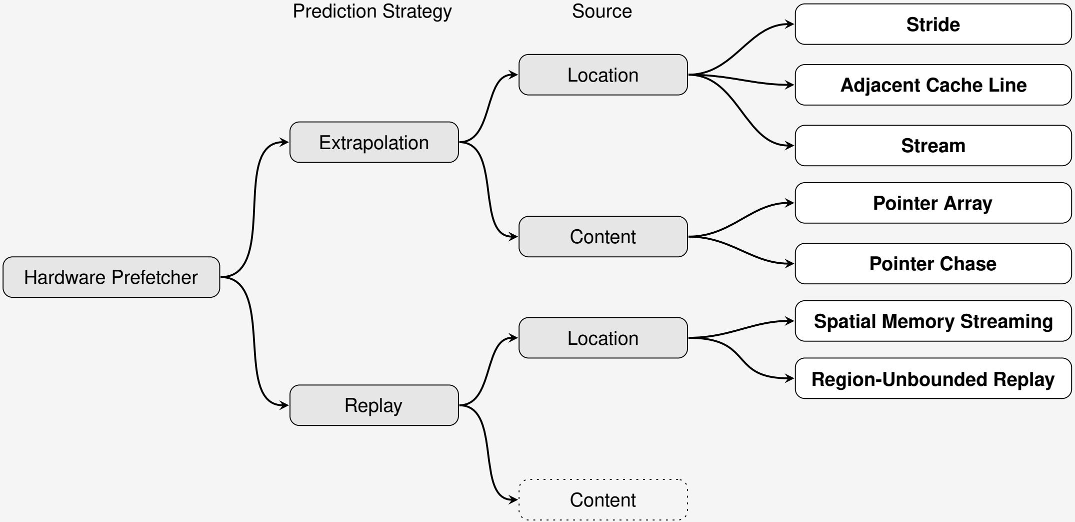
Research Questions

How to identify and characterize prefetchers?

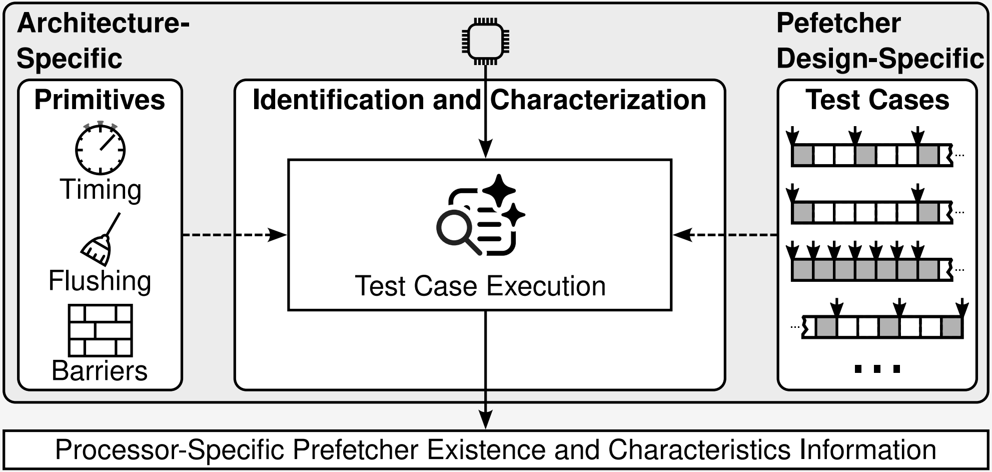
What prefetchers are commonly implemented?

What are the security implications?

Systematization of Prefetching Approaches: Our Taxonomy



Our Framework: FetchBench



Architecture-Specific

Primitives

Timing

Flushing

Barriers

Identification and Characterization

Test Case Execution

Pefetcher Design-Specific

Test Cases

...

Processor-Specific Prefetcher Existence and Characteristics Information

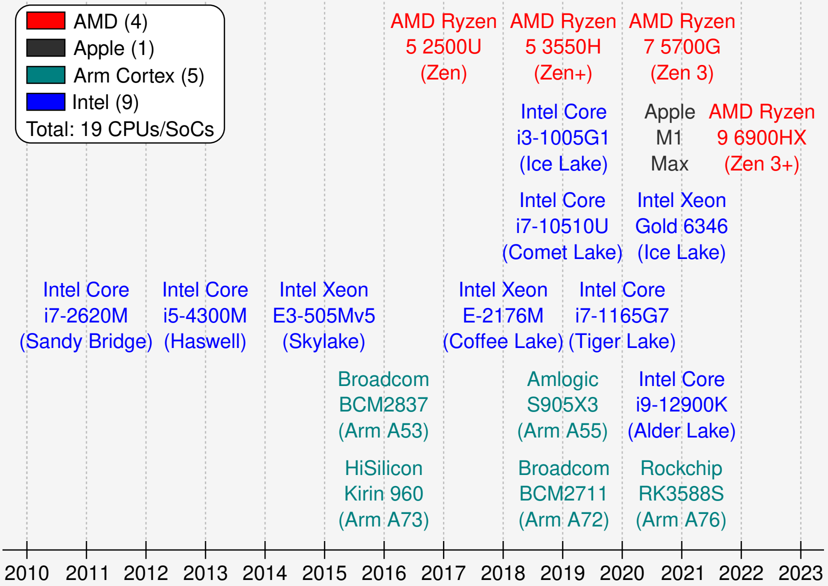
Research Questions

How to identify and characterize prefetchers?

What prefetchers are commonly implemented?

What are the security implications?

CPUs Under Test



- 🟥 AMD (4)
- ⬛ Apple (1)
- 🟩 Arm Cortex (5)
- 🟦 Intel (9)

Total: 19 CPUs/SoCs

Intel Core i7-2620M (Sandy Bridge)

Intel Core i5-4300M (Haswell)

Intel Xeon E3-505Mv5 (Skylake)

Broadcom BCM2837 (Arm A53)

HiSilicon Kirin 960 (Arm A73)

AMD Ryzen 5 2500U (Zen)

Intel Xeon E-2176M (Coffee Lake)

AMD Ryzen 5 3550H (Zen+)

Intel Core i3-1005G1 (Ice Lake)

Intel Core i7-10510U (Comet Lake)

Amlogic S905X3 (Arm A55)

Broadcom BCM2711 (Arm A72)

Intel Core i7-1165G7 (Tiger Lake)

AMD Ryzen 7 5700G (Zen 3)

Apple M1 Max

Intel Xeon Gold 6346 (Ice Lake)

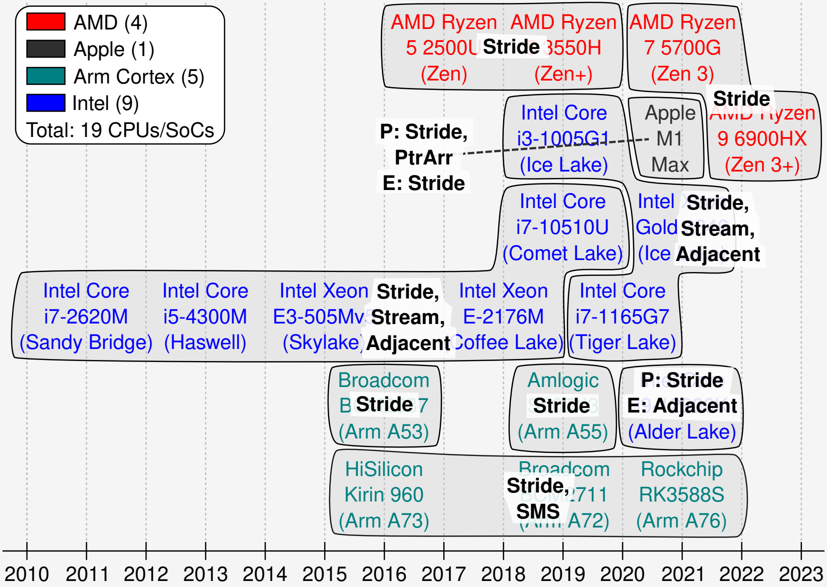
Intel Core i9-12900K (Alder Lake)

Rockchip RK3588S (Arm A76)

AMD Ryzen 9 6900HX (Zen 3+)

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



Preview: More Details in the Paper

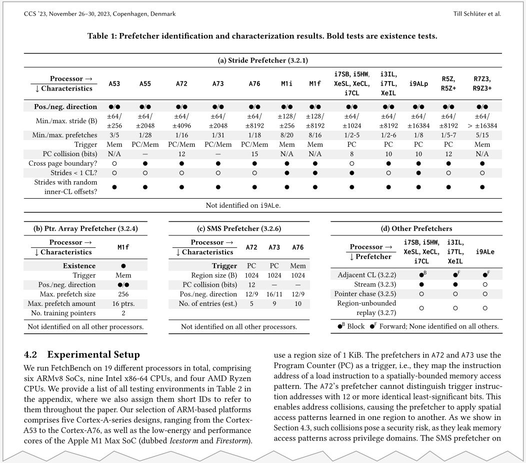


Table 1: Prefetcher identification and characterization results. Bold tests are existence tests.

(a) Stride Prefetcher (3.2.1)

Processor → ↓ Characteristics	A53	A55	A72	A73	A76	M1i	M1f	i7SB, i5HW, XeSL, XeCL, i7CL	i3IL, i7TL, XeIL	i9ALp	R5Z, R5Z+	R7Z3, R9Z3+
Pos./neg. direction	●/●	●/●	●/●	●/●	●/●	●/●	●/●	●/●	●/●	●/●	●/●	●/●
Min./max. stride (B)	±64/ ±256	±64/ ±2048	±64/ ±4096	±64/ ±2048	±64/ ±8192	±128/ ±256	±128/ ±8192	±64/ ±1024	±64/ ±8192	±64/ ±16384	±64/ ±8192	±64/ > ±16384
Min./max. prefetches	3/5	1/28	1/16	1/31	1/18	8/20	8/16	1/2-5	1/2-6	1/8	1/5-7	5/15
Trigger	Mem	PC/Mem	PC/Mem	PC/Mem	PC/Mem	Mem	Mem	PC	PC	PC	PC	Mem
PC collision (bits)	N/A	—	12	—	15	N/A	N/A	8	10	10	12	N/A
Cross page boundary?	○	●	●	●	●	●	●	○	●	●	●	●
Strides < 1 CL?	○	○	○	○	○	●	●	●	○	●	○	○
Strides with random inner-CL offsets?	●	●	●	●	●	●	●	●	●	●	●	●

Not identified on i9ALe.

(b) Ptr. Array Prefetcher (3.2.4)

Processor → ↓ Characteristics	M1f
Existence	●
Trigger	Mem
Pos./neg. direction	●/●
Max. prefetch size	256
Max. prefetch amount	16 ptrs.
No. training pointers	2

Not identified on all other processors.

(c) SMS Prefetcher (3.2.6)

Processor → ↓ Characteristics	A72	A73	A76
Trigger	PC	PC	Mem
Region size (B)	1024	1024	1024
PC collision (bits)	12	—	—
Pos./neg. direction	12/9	16/11	12/9
No. of entries (est.)	5	9	10

Not identified on all other processors.

(d) Other Prefetchers

Processor → ↓ Prefetcher	i7SB, i5HW, XeSL, XeCL, i7CL	i3IL, i7TL, XeIL	i9ALe
Adjacent CL (3.2.2)	●B	●F	●F
Stream (3.2.3)	●	●	○
Pointer chase (3.2.5)	○	○	○
Region-unbounded replay (3.2.7)	○	○	○

●B Block ●F Forward; None identified on all others.

4.2 Experimental Setup

We run FetchBench on 19 different processors in total, comprising six ARMv8 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 to the Cortex-A76, as well as the low-energy and performance cores of the Apple M1 Max SoC (dubbed *Icestorm* and *Firestorm*).

use a region size of 1 KiB. The prefetchers in A72 and A73 use the Program Counter (PC) as a trigger, i.e., they map the instruction address of a load instruction to a spatially-bounded memory access pattern. The A72's prefetcher cannot distinguish trigger instruction addresses with 12 or more identical least-significant bits. This enables address collisions, causing the prefetcher to apply spatial access patterns learned in one region to another. As we show in Section 4.3, such collisions pose a security risk, as they leak memory access patterns across privilege domains. The SMS prefetcher on

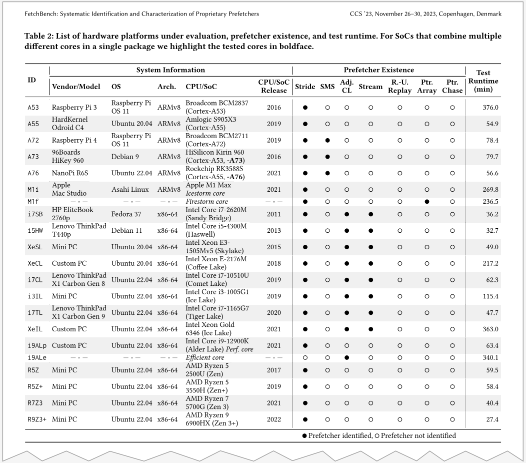


Table 2: List of hardware platforms under evaluation, prefetcher existence, and test runtime. For SoCs that combine multiple different cores in a single package we highlight the tested cores in boldface.

ID	System Information					Prefetcher Existence							Test Runtime (min)
	Vendor/Model	OS	Arch.	CPU/SoC	CPU/SoC Release	Stride	SMS	Adj. CL	Stream	R.-U. Replay	Ptr. Array	Ptr. Chase	
A53	Raspberry Pi 3	Raspberry Pi OS 11	ARMv8	Broadcom BCM2837 (Cortex-A53)	2016	●	○	○	○	○	○	○	376.0
A55	HardKernel Odroid C4	Ubuntu 20.04	ARMv8	Amlogic S905X3 (Cortex-A55)	2019	●	○	○	○	○	○	○	54.9
A72	Raspberry Pi 4	Raspberry Pi OS 11	ARMv8	Broadcom BCM2711 (Cortex-A72)	2019	●	●	○	○	○	○	○	78.4
A73	96Boards HiKey 960	Debian 9	ARMv8	HiSilicon Kirin 960 (Cortex-A53, **-A73**)	2016	●	●	○	○	○	○	○	79.7
A76	NanoPi R6S	Ubuntu 22.04	ARMv8	Rockchip RK3588S (Cortex-A55, **-A76**)	2021	●	●	○	○	○	○	○	56.6
M1i	Apple Mac Studio	Asahi Linux	ARMv8	Apple M1 Max *Icestorm core*	2021	●	○	○	○	○	○	○	269.8
M1f	— " —	— " —	— " —	*Firestorm core*	— " —	●	○	○	○	○	●	○	236.5
i7SB	HP EliteBook 2760p	Fedora 37	x86-64	Intel Core i7-2620M (Sandy Bridge)	2011	●	○	●	●	○	○	○	36.2
i5HW	Lenovo ThinkPad T440p	Debian 11	x86-64	Intel Core i5-4300M (Haswell)	2013	●	○	●	●	○	○	○	32.7
XeSL	Mini PC	Ubuntu 20.04	x86-64	Intel Xeon E3-1505Mv5 (Skylake)	2015	●	○	●	●	○	○	○	49.0
XeCL	Custom PC	Ubuntu 20.04	x86-64	Intel Xeon E-2176M (Coffee Lake)	2018	●	○	●	●	○	○	○	217.2
i7CL	Lenovo ThinkPad X1 Carbon Gen 8	Ubuntu 22.04	x86-64	Intel Core i7-10510U (Comet Lake)	2019	●	○	●	●	○	○	○	62.3
i3IL	Mini PC	Ubuntu 22.04	x86-64	Intel Core i3-1005G1 (Ice Lake)	2019	●	○	●	●	○	○	○	115.4
i7TL	Lenovo ThinkPad X1 Carbon Gen 9	Ubuntu 22.04	x86-64	Intel Core i7-1165G7 (Tiger Lake)	2020	●	○	●	●	○	○	○	47.7
XeIL	Custom PC	Ubuntu 22.04	x86-64	Intel Xeon Gold 6346 (Ice Lake)	2021	●	○	●	●	○	○	○	363.0
i9ALp	Custom PC	Ubuntu 22.04	x86-64	Intel Core i9-12900K (Alder Lake) *Perf. core*	2021	●	○	○	○	○	○	○	63.4
i9ALe	— " —	— " —	— " —	*Efficient core*	— " —	○	○	●	○	○	○	○	340.1
R5Z	Mini PC	Ubuntu 22.04	x86-64	AMD Ryzen 5 2500U (Zen)	2017	●	○	○	○	○	○	○	59.5
R5Z+	Mini PC	Ubuntu 22.04	x86-64	AMD Ryzen 5 3550H (Zen+)	2019	●	○	○	○	○	○	○	58.4
R7Z3	Mini PC	Ubuntu 22.04	x86-64	AMD Ryzen 7 5700G (Zen 3)	2021	●	○	○	○	○	○	○	40.4
R9Z3+	Mini PC	Ubuntu 22.04	x86-64	AMD Ryzen 9 6900HX (Zen 3+)	2022	●	○	○	○	○	○	○	27.4

● Prefetcher identified, ○ Prefetcher not identified

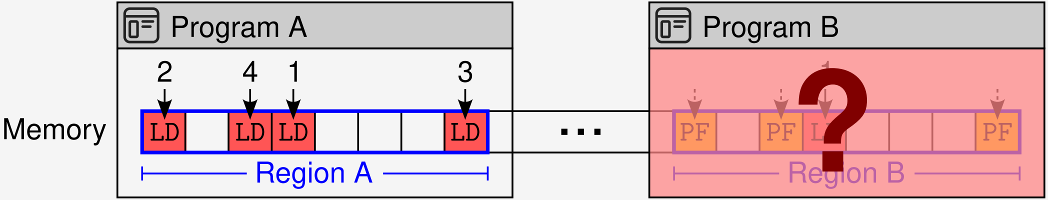
Research Questions

How to identify and characterize prefetchers?

What prefetchers are commonly implemented?

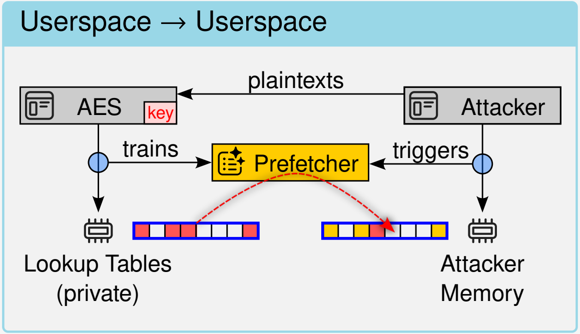
What are the security implications?

Exploiting Spatial Memory Streaming (SMS)

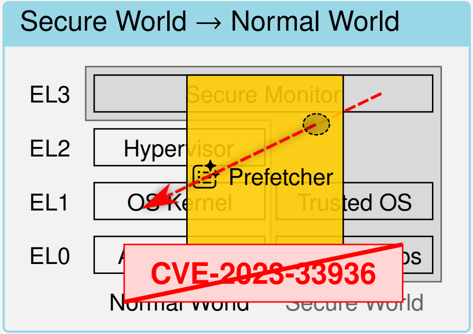


Case Studies

Userspace → Userspace

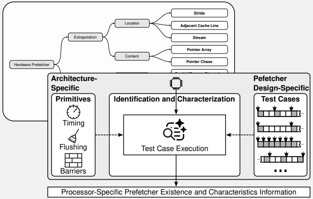


Secure World → Normal World

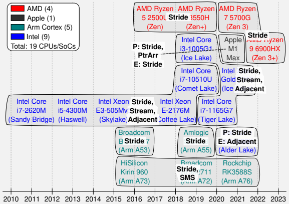


~~CVE-2023-33936~~

Taxonomy & FetchBench



CPU Characterization



Case Studies (SMS)

- Userspace → Userspace
- Secure World → Normal World

github.com/scy-phy/FetchBench

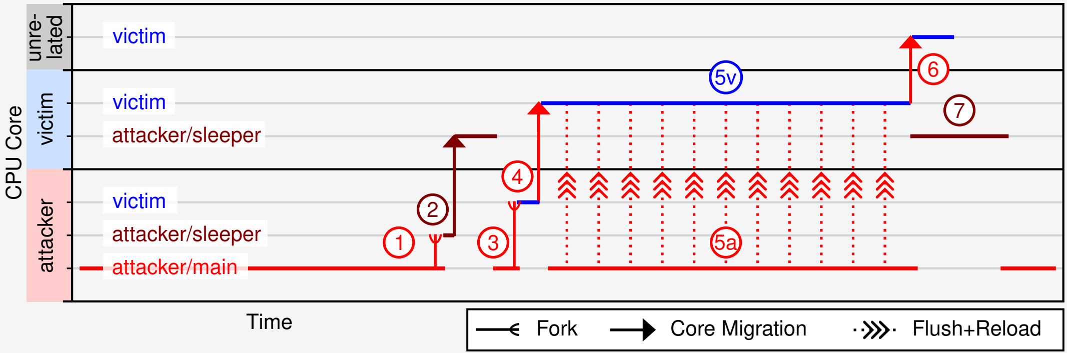
Contact me: Till Schlüter

till.schlueter@cispa.de

tschlueter.com

(List of all resources related to this paper)

AES Case Study: Synchronization



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

[1] Yun Chen, Lingfeng Pei, and Trevor E. Carlson. "AfterImage: Leaking Control Flow Data and Tracking Load Operations via the Hardware Prefetcher". In: ASPLOS '23. 2023. DOI: 10.1145/3575693.3575719.

[2] Jose Rodrigo Sanchez Vicarte et al. "Augury: Using Data Memory-Dependent Prefetchers to Leak Data at Rest". In: S&P '22. 2022. DOI: 10.1109/SP46214.2022.9833570.

[3] Youngjoo Shin et al. "Unveiling Hardware-Based Data Prefetcher, a Hidden Source of Information Leakage". In: CCS '18. 2018. DOI: 10.1145/3243734.3243736.