Week 2 Report

Ben Chen

Dept of Computer Science and Engineering, SUSTech

September 19, 2024

TOC

Title	Conference	Institute	Authors	Idea
A Security RISC: Microarchitectural Attacks on Hardware RISC-V CPUs	Oakland '23	CISPA	Lukas Gerlach Daniel Weber Ruiyi Zhang Michael Schwarz	Cache+Time, Flush+Fault, CycleDrift on SiFive U74 & T-Head C906 with 6 case studies
(M)WAIT for It: Bridging the Gap between Microarchitectural and Architectural Side Channels	USENIX '23	CISPA	Ruiyi Zhang Taehyun Kim Daniel Weber Michael Schwarz	Exploiting `umonitor` and `umwait` to enhance spectre attack without timer

A Security RISC[1]

Systematic analysis of microarchitectural components:

- Hardware Timer: `rdcycle` and `rdtime` instructions on both CPUs, a higher resolution in retire instruction counter on U74 via `rdinstret` or `csrr` instructions.

A Security RISC[1]

Systematic analysis of microarchitectural components:

- Hardware Timer: rdcycle and rdtime instructions on both CPUs, a higher resolution in retire instruction counter on U74 via rdinstret or csrr instructions.
- Cache: Capacity and replacement policy (deterministic), FIFO on C906 and PLRU on U74; Cache maintainance instructions: fence.i by ISA and dcache.civa by C906 ⇒ efficient cache eviction.

A Security RISC[1]

Systematic analysis of microarchitectural components:

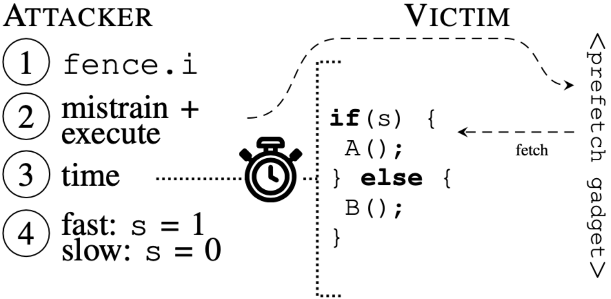
- Hardware Timer: `rdcycle` and `rdtime` instructions on both CPUs, a higher resolution in retire instruction counter on U74 via `rdinstret` or `csrr` instructions.
- Cache: Capacity and replacement policy (deterministic), FIFO on C906 and PLRU on U74; Cache maintainance instructions: `fence.i` by ISA and `dcache.civa` by C906 ⇒ efficient cache eviction.
- TLB: SV39, two separate 10-entry fully-associative TLBs for data and instructions, effectively evictable.

A Security RISC[1]

Systematic analysis of microarchitectural components:

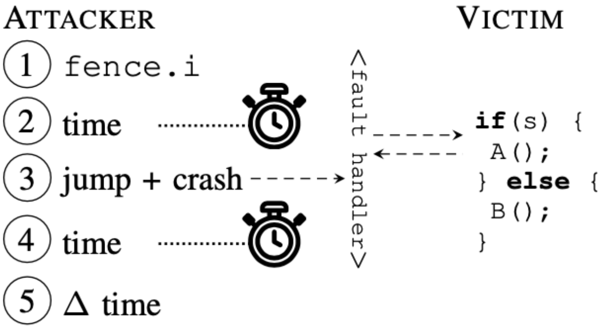
- Hardware Timer: `rdcycle` and `rdtime` instructions on both CPUs, a higher resolution in retire instruction counter on U74 via `rdinstret` or `csrr` instructions.
- Cache: Capacity and replacement policy (deterministic), FIFO on C906 and PLRU on U74; Cache maintainance instructions: `fence.i` by ISA and `dcache.civa` by C906 $\Rightarrow$ efficient cache eviction.
- TLB: SV39, two separate 10-entry fully-associative TLBs for data and instructions, effectively evictable.
- BRU: In-order pipeline but speculative prefetching, with BHT, BJT, and RAS.

A Security RISC[1]



flush i$ ⇒ mistrain to load A but not B ⇒ timing difference

A Security RISC[1]



```
if(s) {
 A();
} else {
 B();
}
```

flush i$ ⇒ jump to victim's code and trigger a fault ⇒ timing difference to determine whether `A` or `B` is in cache

- `addi a0, zero, zero` in attacker's code
- then jump to `ld xx, 0(a0)` in victim's code

A Security RISC[1]

Monitor number of retired instructions with a certain cycles:

Platform	SBI_EXT_BASE_GET_MVENDORID	SBI_EXT_0_1_CONSOLE_PUTCHAR	padded square-and-multiply
U74	963 cycles	85507 cycles	14(-3) instructions
C906	613 cycles	85109 cycles	18(-2) instructions

A Security RISC[1]

Case studies:

- Square and Multiply in MbedTLS: Flush+Fault, Cache+Time

A Security RISC[1]

Case studies:

- Square and Multiply in MbedTLS: Flush+Fault, Cache+Time
- Breaking KASLR: CycleDrift, timming difference in page-table walk

A Security RISC[1]

Case studies:

- Square and Multiply in MbedTLS: Flush+Fault, Cache+Time
- Breaking KASLR: CycleDrift, timming difference in page-table walk
- Zigzagger Bypass: CycleDrift

A Security RISC[1]

Case studies:

- Square and Multiply in MbedTLS: Flush+Fault, Cache+Time
- Breaking KASLR: CycleDrift, timming difference in page-table walk
- Zigzagger Bypass: CycleDrift
- Leaking Contents of a Drop-Box Folder: CycleDrift,

A Security RISC[1]

Case studies:

- Square and Multiply in MbedTLS: Flush+Fault, Cache+Time
- Breaking KASLR: CycleDrift, timming difference in page-table walk
- Zigzagger Bypass: CycleDrift
- Leaking Contents of a Drop-Box Folder: CycleDrift,
- Interrupt Detection: CycleDrift

A Security RISC[1]

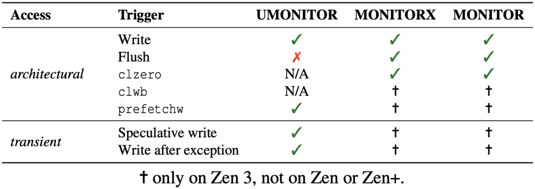
Case studies:

- Square and Multiply in MbedTLS: Flush+Fault, Cache+Time
- Breaking KASLR: CycleDrift, timming difference in page-table walk
- Zigzagger Bypass: CycleDrift
- Leaking Contents of a Drop-Box Folder: CycleDrift,
- Interrupt Detection: CycleDrift
- OpenSSL 1.0.1 AES T-Table: Cache attacks

(M)WAIT For it[2]

Motivation:

- Intel introduces `umonitor` and `umwait` in Alder Lake to optimize idle-loop.



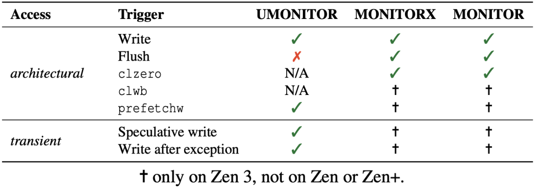
Access	Trigger	UMONITOR	MONITORX	MONITOR
architectural	Write	✓	✓	✓
	Flush	✗	✓	✓
	`clzero`	N/A	✓	✓
	`clwb`	N/A	†	†
	`prefetchw`	✓	†	†
transient	Speculative write	✓	†	†
	Write after exception	✓	†	†

† only on Zen 3, not on Zen or Zen+.

(M)WAIT For it[2]

Motivation:

- Intel introduces `umonitor` and `umwait` in Alder Lake to optimize idle-loop.
- AMD has similar instructions `monitorx` and `mwaitx`.



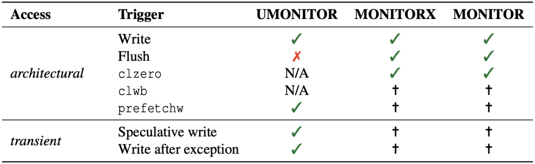
Access	Trigger	UMONITOR	MONITORX	MONITOR
architectural	Write	✓	✓	✓
	Flush	✗	✓	✓
	`clzero`	N/A	✓	✓
	`clwb`	N/A	†	†
	`prefetchw`	✓	†	†
transient	Speculative write	✓	†	†
	Write after exception	✓	†	†

† only on Zen 3, not on Zen or Zen+.

(M)WAIT For it[2]

Motivation:

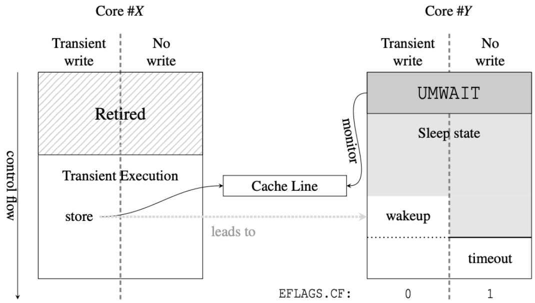
- Intel introduces `umonitor` and `umwait` in Alder Lake to optimize idle-loop.
- AMD has similar instructions `monitorx` and `mwaitx`.
- `umonitor` tells monitor component to start monitoring a memory region, `umwait` waits until the region is modified (either in cache or memory), interrupt arrives or a timeout.



Access	**Trigger**	**UMONITOR**	**MONITORX**	**MONITOR**
architectural	Write	✓	✓	✓
	Flush	✗	✓	✓
	`clzero`	N/A	✓	✓
	`clwb`	N/A	†	†
	`prefetchw`	✓	†	†
transient	Speculative write	✓	†	†
	Write after exception	✓	†	†

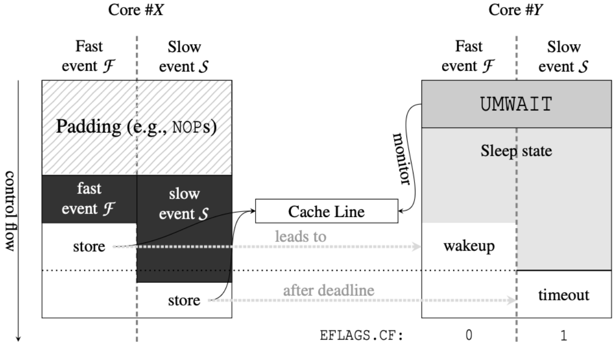
† only on Zen 3, not on Zen or Zen+.

(M)WAIT For it[2]



- Carry flag (CF) is set to 0 if waken up by writing cache/memory, to 1 if waken up due to timeout.
- Leaks architectural state about whether the victim transient writes to the target cache line

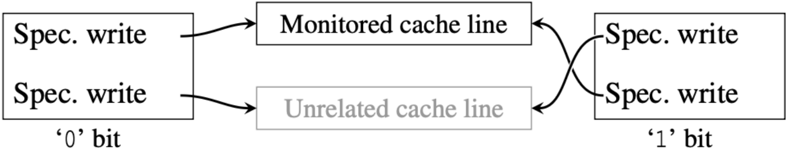
(M)WAIT For it[2]



Set a threshold to determine if the targeted memory region is present in cache line ⇒ enable a spectre attack without timer.

(M)WAIT For it[2]

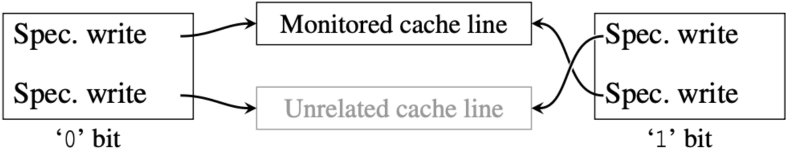
Storytelling: build a covert channel to transmit data without a timer.



- Carry flag as Manchester-encoded bit to transmit data

(M)WAIT For it[2]

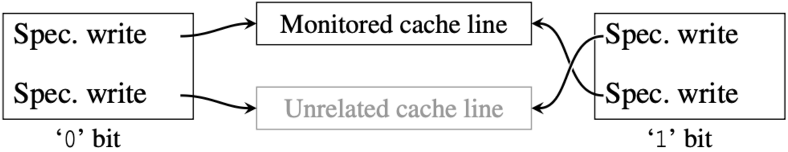
Storytelling: build a covert channel to transmit data without a timer.



- Carry flag as Manchester-encoded bit to transmit data
- An rising edge in CF indicates a 1, a falling edge indicates a 0

(M)WAIT For it[2]

Storytelling: build a covert channel to transmit data without a timer.



- Carry flag as Manchester-encoded bit to transmit data
- An rising edge in CF indicates a 1, a falling edge indicates a 0
- Without synchronization, the channel achieves 697 bit/s

(M)WAIT For it[2]

Case studies:

- Spectral: Enhance Spectre PHT with TWM, by monitoring if the target cache line is written without probing the cache line.

(M)WAIT For it[2]

Case studies:

- Spectral: Enhance Spectre PHT with TWM, by monitoring if the target cache line is written without probing the cache line.
- Timerless Cache Attacks on OpenSSL 1.0.1 T-Table: Substitute Prime+Probe with TLT.

(M)WAIT For it[2]

Case studies:

- Spectral: Enhance Spectre PHT with TWM, by monitoring if the target cache line is written without probing the cache line.
- Timerless Cache Attacks on OpenSSL 1.0.1 T-Table: Substitute Prime+Probe with TLT.
- Network Fingerprints: Utilize `umwait`'s waken up by external interrupt to fingerprint the network interrupts with a time bucket.

References I

[1] Lukas Gerlach et al. "A Security RISC: Microarchitectural Attacks on Hardware RISC-V CPUs". In: *2023 IEEE Symposium on Security and Privacy (SP)*. 2023, pp. 2321–2338. DOI: 10.1109/SP46215.2023.10179399.

[2] Ruiyi Zhang et al. "(M)WAIT for It: Bridging the Gap between Microarchitectural and Architectural Side Channels". In: *32nd USENIX Security Symposium (USENIX Security 23)*. Anaheim, CA: USENIX Association, Aug. 2023, pp. 7267–7284. ISBN: 978-1-939133-37-3. URL: https://www.usenix.org/conference/usenixsecurity23/presentation/zhang-ruiyi.