
black hat
MIDDLE EAST AND AFRICA
(M)Wait for it
Bridging the Gap between Microarchitectural and Architectural Side Channels
15 - 17 NOVEMBER 2022
RIYADH FRONT EXHIBITION CENTRE
SAUDI ARABIA
Ruiyi Zhang, Daniel Weber, Michael Schwarz
CO-ORGANISED BY:
informatech
IN PARTNERSHIP WITH:
Riyadh

# Who am I?

**Ruiyi Zhang**

PhD Student @ CISPA Helmholtz Center for Information Security

@Rayiizzz

ruiyi.zhang@cispa.de

# Who am I?

**Daniel Weber**

PhD Student @ CISPA Helmholtz Center for Information Security

@weber_daniel

daniel.weber@cispa.de

# Who am I?

**Michael Schwarz**

Faculty @ CISPA Helmholtz Center for Information Security

@misc0110

michael.schwarz@cispa.de

# About this presentation

This talk is about how the power-optimization instructions *UMONITOR/UMWAIT* leak information

# About this presentation

This talk is about how the power-optimization instructions *UMONITOR/UMWAIT* leak information

- Not about software bugs

# About this presentation

This talk is about how the power-optimization instructions *UMONITOR/UMWAIT* leak information

- Not about software bugs
- A side channel to leak information

# About this presentation

This talk is about how the power-optimization instructions *UMONITOR/UMWAIT* leak information

- Not about software bugs
- A side channel to leak information
- Not due to the hardware design, but due to the instructions themselves

# Livedemo: Spectral

# What can we do

- Combined with transient-execution attacks

# What can we do

- Combined with transient-execution attacks
- Cross-vm Cross-CPU covert channel

# What can we do

- Combined with transient-execution attacks
- Cross-vm Cross-CPU covert channel
- Leakage of cryptographic keys

# What can we do

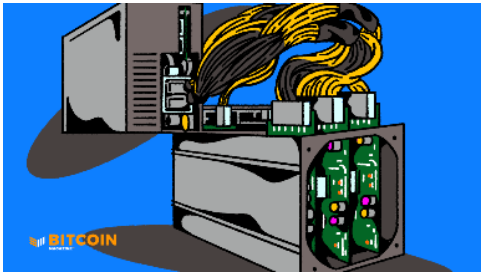
- Combined with transient-execution attacks
- Cross-vm Cross-CPU covert channel
- Leakage of cryptographic keys
- Mainly focus on the Intel's newest CPUs (since Alder Lake and Tremont-based)

# What can we do

- Combined with transient-execution attacks
- Cross-vm Cross-CPU covert channel
- Leakage of cryptographic keys
- Mainly focus on the Intel's newest CPUs (since Alder Lake and Tremont-based)
- But a generic interrupt-timing attack is also available on AMD and ARM


ENERGY CRISIS

# Heat Your Home with Bitcoin Mining

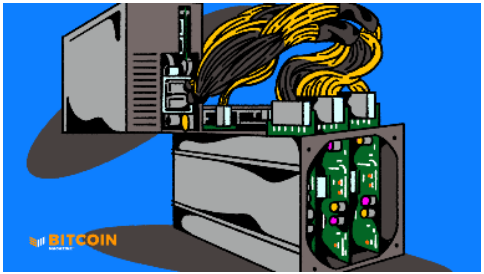



Build an additional heating system from mining cryptocurrency

- Repurposing the waste heat

Warren, R. (2022, March 25). *HOW TO HEAT YOUR HOME WITH BITCOIN MINING*. Bitcoin Magazine. https://bitcoinmagazine.com/business/how-to-heat-your-home-with-bitcoin-mining

# Heat Your Home with Bitcoin Mining



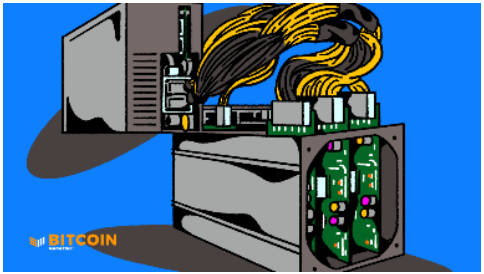

Build an additional heating system from mining cryptocurrency

- Repurposing the waste heat

Overclocking your gaming desktop for heating?

Warren, R. (2022, March 25). *HOW TO HEAT YOUR HOME WITH BITCOIN MINING*. Bitcoin Magazine. https://bitcoinmagazine.com/business/how-to-heat-your-home-with-bitcoin-mining

# Heat Your Home with Bitcoin Mining




Build an additional heating system from mining cryptocurrency

- Repurposing the waste heat

Overclocking your gaming desktop for heating?

 (for longer lifespan of CPU)

Warren, R. (2022, March 25). *HOW TO HEAT YOUR HOME WITH BITCOIN MINING*. Bitcoin Magazine. https://bitcoinmagazine.com/business/how-to-heat-your-home-with-bitcoin-mining

# How does a Processor Save Energy?

CAN'T WASTE POWER


IF YOU DON'T TURN ON IT

# How does a Processor Save Energy?

Idle

Busy

# How does a Processor Save Energy?

Idle

Busy

- Idle the processor

# How does a Processor Save Energy?

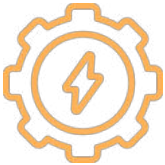
Idle

- Idle the processor

Busy

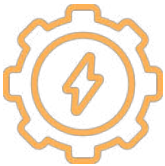
- Lower the voltage
- Decrease the frequency

# Processor Power Management



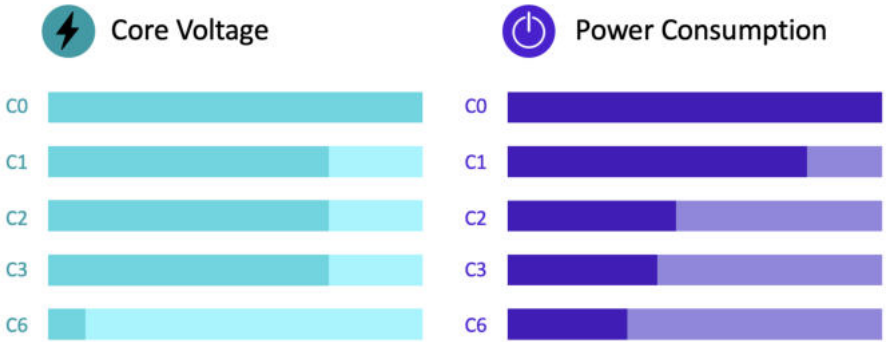
- Modern processors support multiple technologies to optimize the power consumption:
  - CPUIdle driver
  - Scaling drivers (intel_pstate, intel_cpufreq, ...)

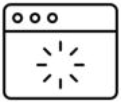
# Processor Power Management



- Modern processors support multiple technologies to optimize the power consumption:
  - CPUIdle driver
  - Scaling drivers (intel_pstate, intel_cpufreq, ...)
- Which corresponds to ...
  - C-states (Idle States)
  - P-states (Performance States)

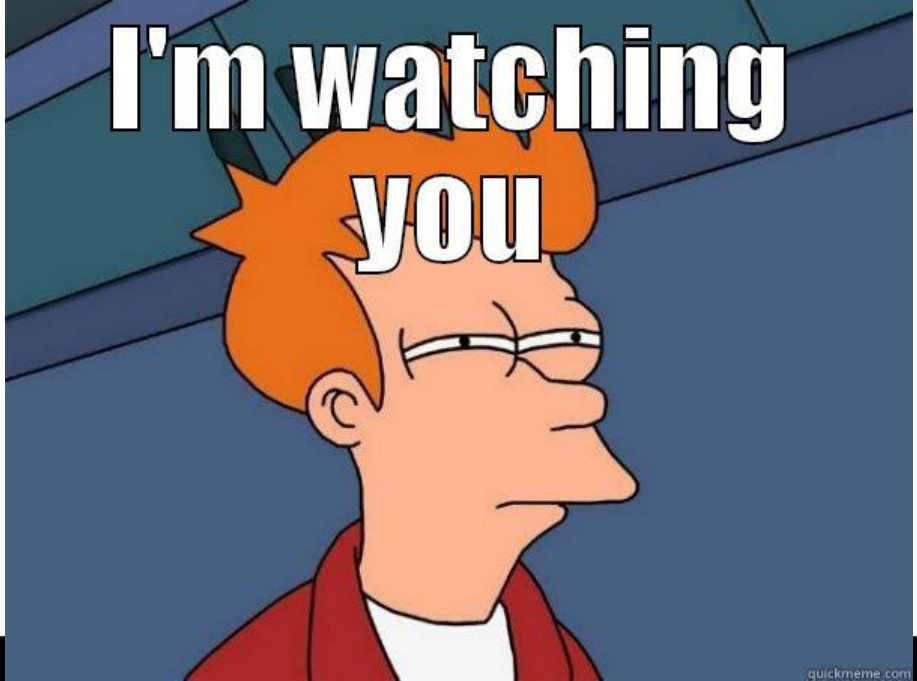
# Idle States





## Privileged Instructions

- (↓C1) HLT - Halt
  - Stops instruction execution, puts the processor in a HALT state
  - Wakes up by interrupts
  - Can be configured to restart after interrupts
- (↓Cx) MONITOR/MWAIT


I'm watching you
quickmeme.com

- MONITOR: Set up an address range for the monitor hardware

# MONITOR / MWAIT

- MONITOR: Set up an address range for the monitor hardware
- MWAIT: Wait until a write-back store is performed within the range specified by the instruction MONITOR

# MONITOR / MWAIT

- MONITOR: Set up an address range for the monitor hardware
- MWAIT: Wait until a write-back store is performed within the range specified by the instruction MONITOR
  - EAX contains the hint of the preferred optimized state

# MONITOR / MWAIT

- MONITOR: Set up an address range for the monitor hardware
- MWAIT: Wait until a write-back store is performed within the range specified by the instruction MONITOR
  - EAX contains the hint of the preferred optimized state
  - ECX specifies optional extensions

# MONITOR / MWAIT

- MONITOR: Set up an address range for the monitor hardware
- MWAIT: Wait until a write-back store is performed within the range specified by the instruction MONITOR
  - EAX contains the hint of the preferred optimized state
  - ECX specifies optional extensions
  - Processor may exit the optimized state due to a store to the monitored memory range or an interrupt

# Timed Wait?

- An undocumented property of the MWAIT instruction on Intel (at least since Skylake microarchitecture)

# Timed Wait?

- An undocumented property of the MWAIT instruction on Intel (at least since Skylake microarchitecture)
- The timed MWAIT wakes up when the timestamp counter reaches or exceeds a specified value

# Timed Wait?

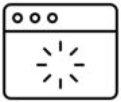
- An undocumented property of the MWAIT instruction on Intel (at least since Skylake microarchitecture)
- The timed MWAIT wakes up when the timestamp counter reaches or exceeds a specified value
  - Can be enabled by setting bit 31 in MSR (0xe2)

# Timed Wait?

- An undocumented property of the MWAIT instruction on Intel (at least since Skylake microarchitecture)
- The timed MWAIT wakes up when the timestamp counter reaches or exceeds a specified value
  - Can be enabled by setting bit 31 in MSR (0xe2)
  - The bit 1 of ECX of MWAIT indicates this feature is used

# Timed Wait?

- An undocumented property of the MWAIT instruction on Intel (at least since Skylake microarchitecture)
- The timed MWAIT wakes up when the timestamp counter reaches or exceeds a specified value
  - Can be enabled by setting bit 31 in MSR (0xe2)
  - The bit 1 of ECX of MWAIT indicates this feature is used
  - The maximum waiting time is an implicit 64-bit timestamp-counter value stored in the EDX:EBX register pair



## Unprivileged Instructions

- (↓C0.2) TPAUSE - Timed PAUSE (Intel)
- (↓C0.2) UMONITOR/UMWAIT (Intel)
- (↓C1) MONITORX/MWAITX (AMD)

# Wakeup Triggers of UMWAIT

# Wakeup Triggers of UMWAIT

- A store to the monitored memory range

# Wakeup Triggers of UMWAIT

- A store to the monitored memory range
- Interrupts

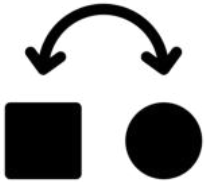
# Wakeup Triggers of UMWAIT

- A store to the monitored memory range
- Interrupts
- User-defined timeout (the implicit EDX:EAX 64-bit input value)

# Wakeup Triggers of UMWAIT

- A store to the monitored memory range
- Interrupts
- User-defined timeout (the implicit EDX:EAX 64-bit input value)
- OS-defined timeout (writing TSC-quanta value to MSR 0xe1)

# Architectural Difference



If the processor exits the optimized state due to the OS-defined timeout, the UMWAIT instruction sets RFLAGS.CF

# MONITOR / MWAIT

- MONITOR: Set up an address range for the monitor hardware
- MWAIT: Wait until a write-back store is performed within the range specified by the instruction MONITOR
  - EAX contains the hint of the preferred optimized state
  - ECX specifies optional extensions
  - Processor may exit the optimized state due to a store to the monitored memory range or an interrupt

# How does UMONITOR/UMWAIT Spy on a Memory Range?

# How does UMONITOR/UMWAIT Spy on a Memory Range?

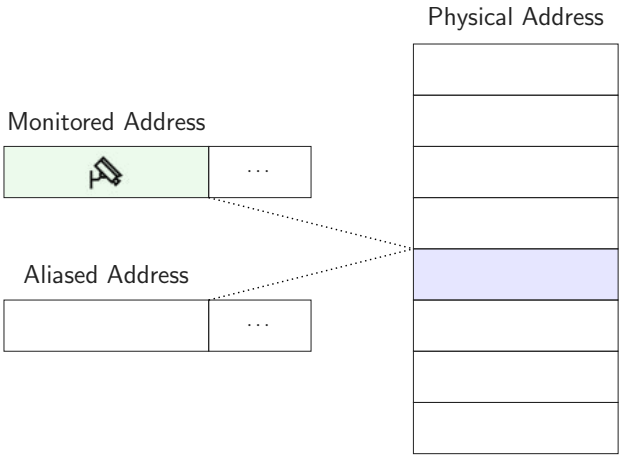

Monitored Address

# How does UMONITOR/UMWAIT Spy on a Memory Range?

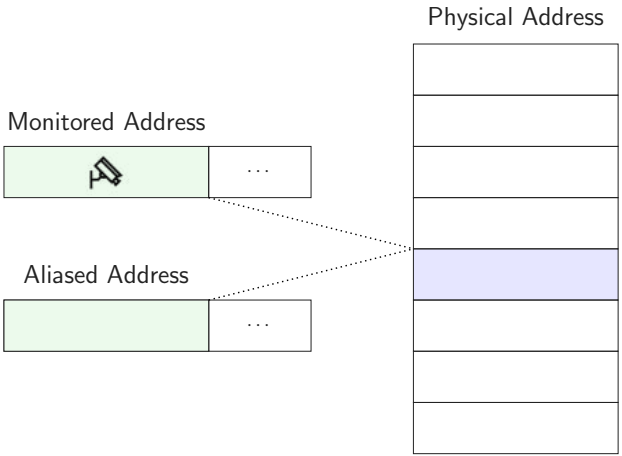

Monitored Address


Aliased Address

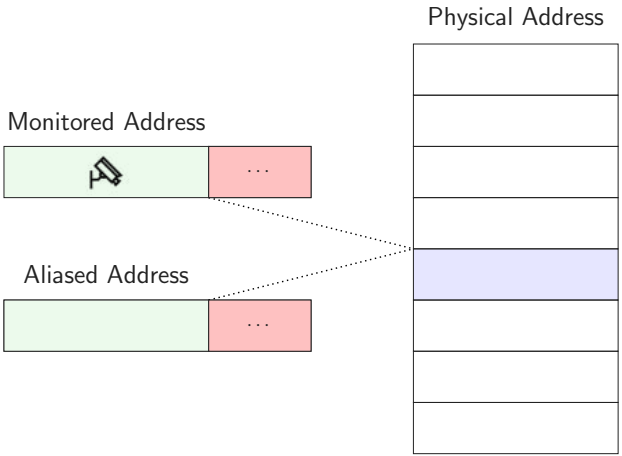
# How does UMONITOR/UMWAIT Spy on a Memory Range?

# How does UMONITOR/UMWAIT Spy on a Memory Range?

# How does UMONITOR/UMWAIT Spy on a Memory Range?

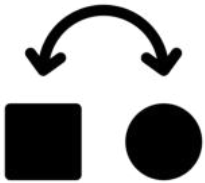



# How does UMONITOR/UMWAIT Spy on a Memory Range?

| 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+

# Architectural Feedback for Microarchitectural State

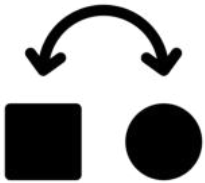


- Modification to the monitored microarchitectural state → RFLAGS.CF=0
- Interrupts → RFLAGS.CF=0
- User-defined timeout → RFLAGS.CF=0

---

- OS-defined timeout → RFLAGS.CF=1

# Architectural Feedback for Microarchitectural State



- Modification to the monitored microarchitectural state → RFLAGS.CF=0
- Interrupts → RFLAGS.CF=0
- User-defined timeout → RFLAGS.CF=0

---

- OS-defined timeout → RFLAGS.CF=1

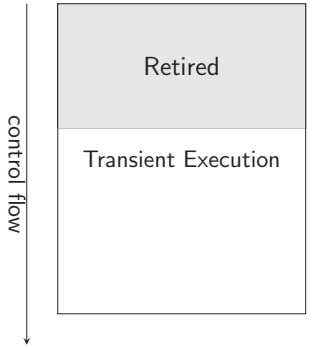
→ A side channel to convert the microarchitectural state into architectural state

# Attack Primitive #1: Transient-Write Monitor

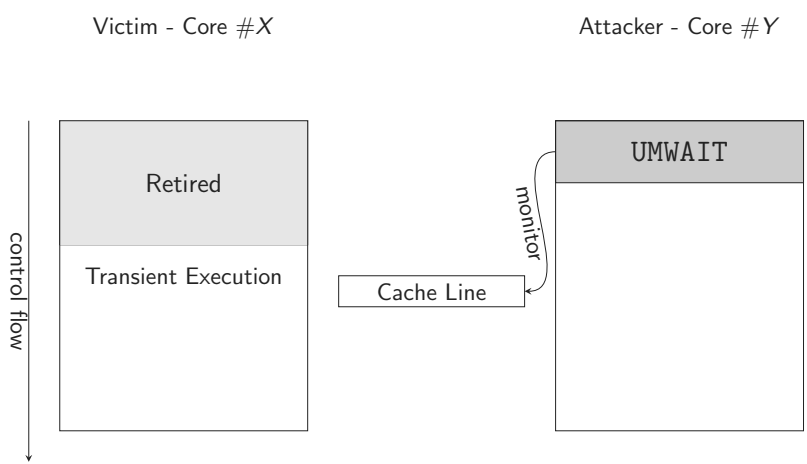
# Attack Primitive #1: Transient-Write Monitor

# Attack Primitive #1: Transient-Write Monitor

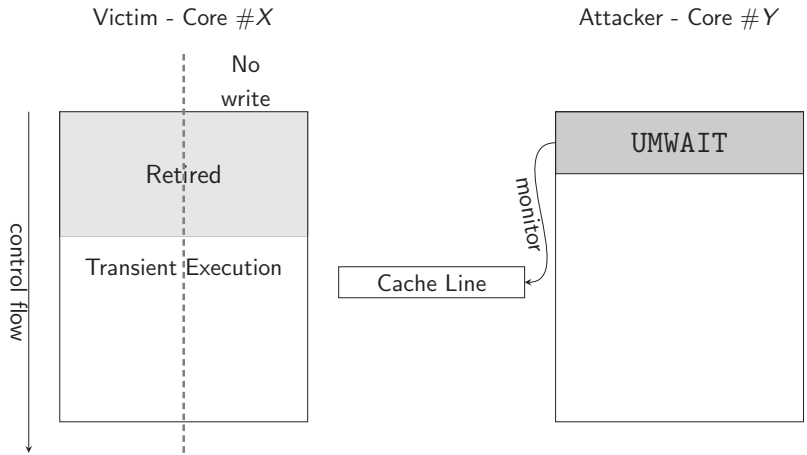
Victim - Core #$X$



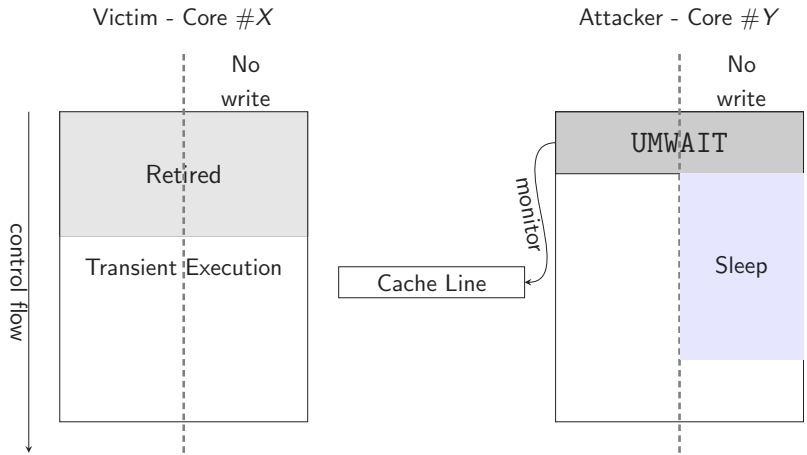
# Attack Primitive #1: Transient-Write Monitor



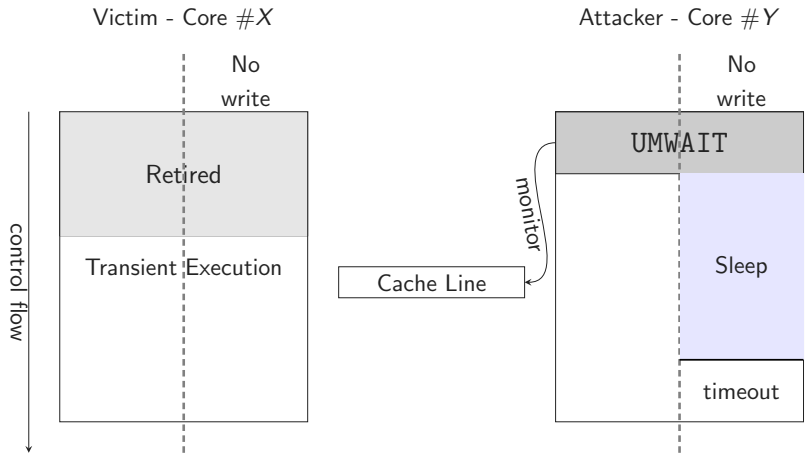
# Attack Primitive #1: Transient-Write Monitor



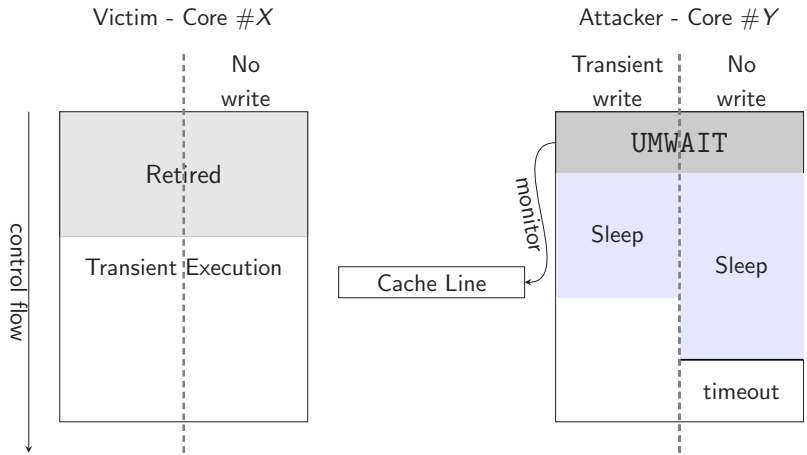
# Attack Primitive #1: Transient-Write Monitor



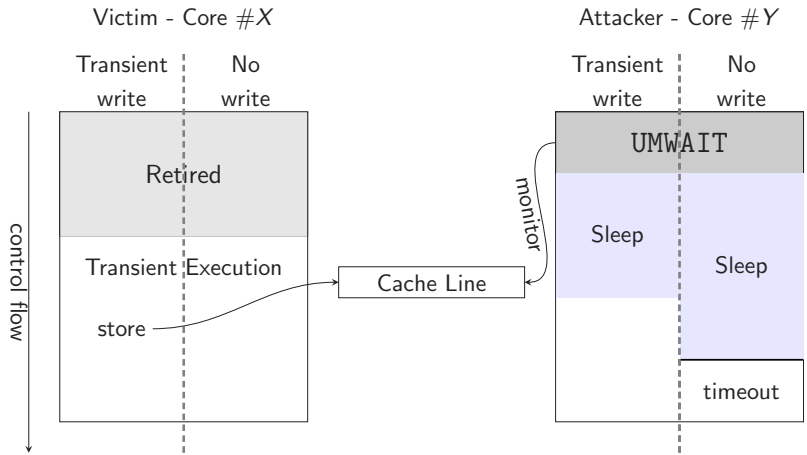
# Attack Primitive #1: Transient-Write Monitor



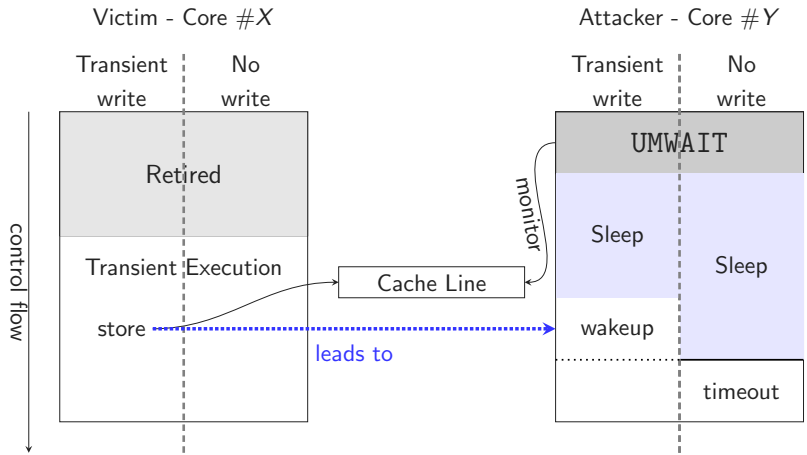
# Attack Primitive #1: Transient-Write Monitor



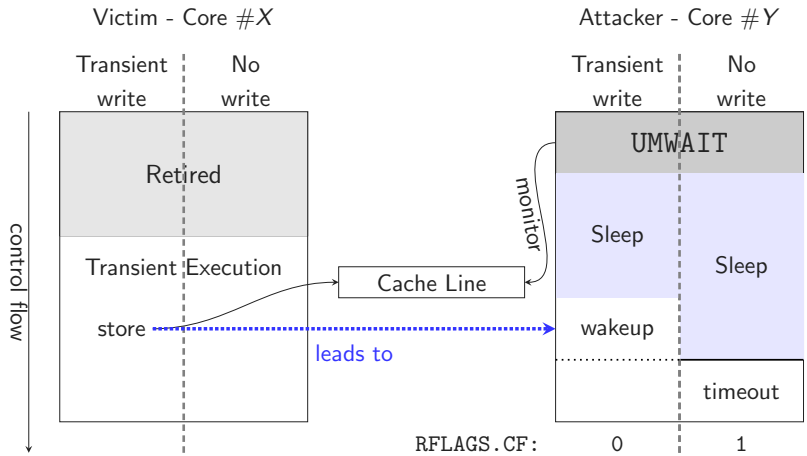
# Attack Primitive #1: Transient-Write Monitor



# Attack Primitive #1: Transient-Write Monitor



# Attack Primitive #1: Transient-Write Monitor



# Properties

Low noise

# Properties

Low noise

High accuracy

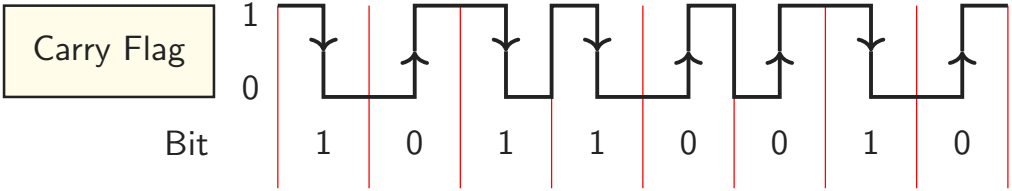
# Properties

Low noise

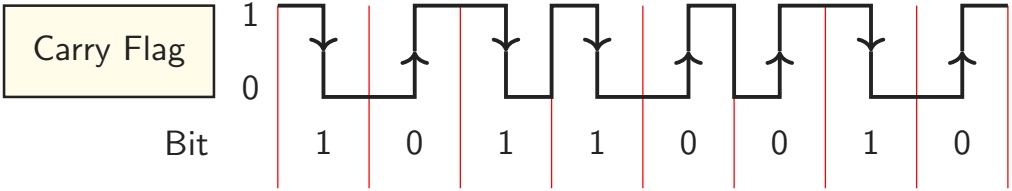
High accuracy

Stealthy

# UMWAIT Covert Channel



# UMWAIT Covert Channel




- Cross-core cross-VM covert channel
- 697 bit/s
- 0% error rate

# Spectre with Architectural Leakage

```
if ( x < array1_size)
  array2[(array1[x] * 64)] = ...;
```

# Spectre with Architectural Leakage

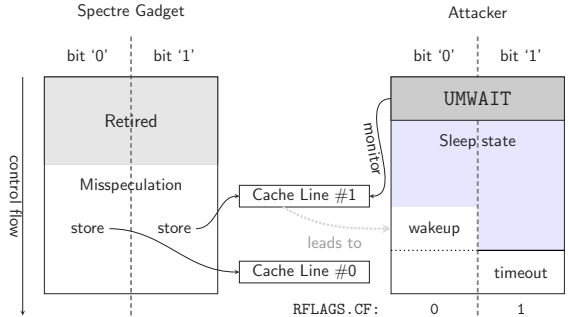
```
if ( x < array1_size)
  array2[(array1[x] * 64)] = ...;
```

Monitor the cache line | array2[0]

# Spectre with Architectural Leakage

```
if ( x < array1_size)
  array2[(array1[x] * 64)] = ...;
```

Monitor the cache line | array2[0]



# Attack Primitive #2: Timer-less Timing Measurement

# Attack Primitive #2: Timer-Less Timing Measurement

- Distinguish fast events from slow events

# Attack Primitive #2: Timer-Less Timing Measurement

- Distinguish fast events from slow events
- High-resolution timer:

# Attack Primitive #2: Timer-Less Timing Measurement

- Distinguish fast events from slow events
- High-resolution timer:
  → Timestamp Counter

# Attack Primitive #2: Timer-Less Timing Measurement

- Distinguish fast events from slow events
- High-resolution timer:
  - → Timestamp Counter
  - → Counting Thread

# Attack Primitive #2: Timer-Less Timing Measurement

- Distinguish fast events from slow events
- High-resolution timer:
  - → Timestamp Counter
  - → Counting Thread
  - → ... ...

# Attack Primitive #2: Timer-Less Timing Measurement

```
memory access
```

# Attack Primitive #2: Timer-Less Timing Measurement

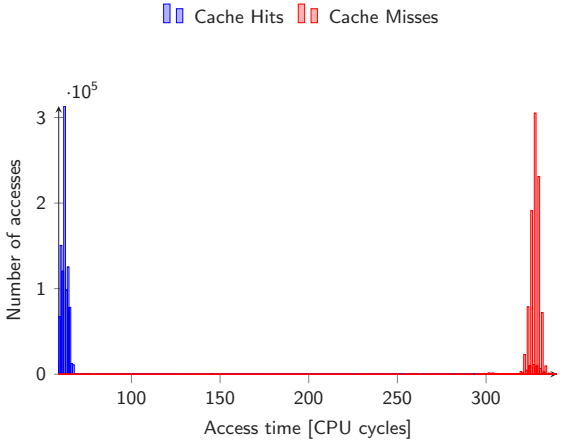
```
start = ⏱
 memory access
end = ⏱
```

# Attack Primitive #2: Timer-Less Timing Measurement

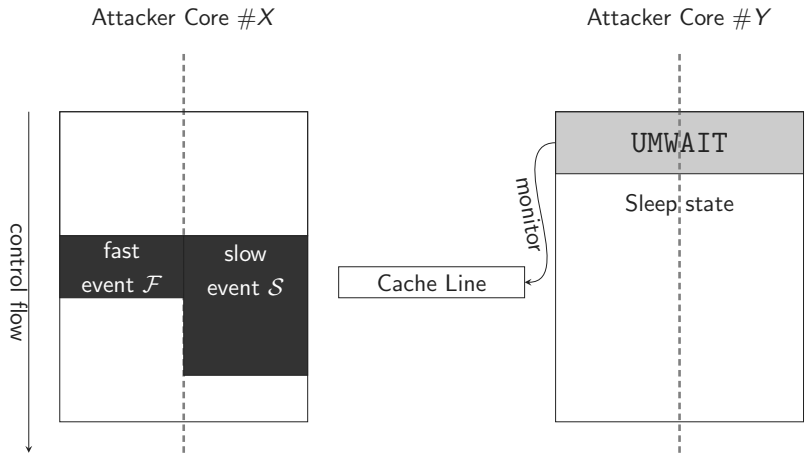
```
start = ⏱
 memory access
end = ⏱
Δ = end - start
```

# Attack Primitive #2: Timer-Less Timing Measurement

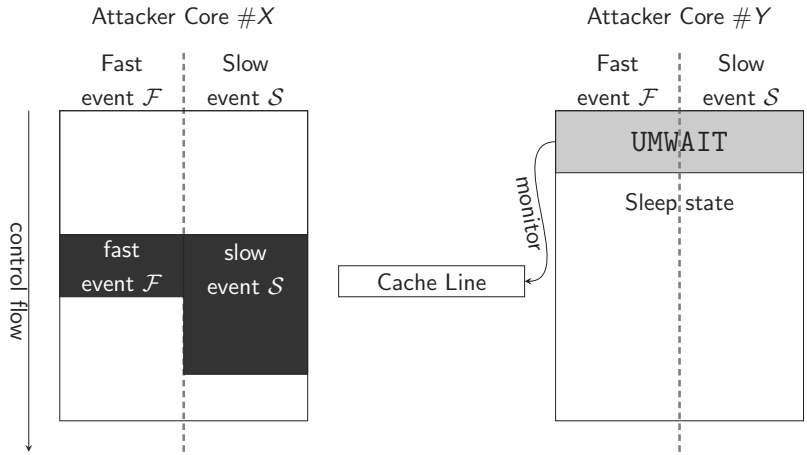
```
start = ⏱
 memory access
end = ⏱
Δ = end - start
```



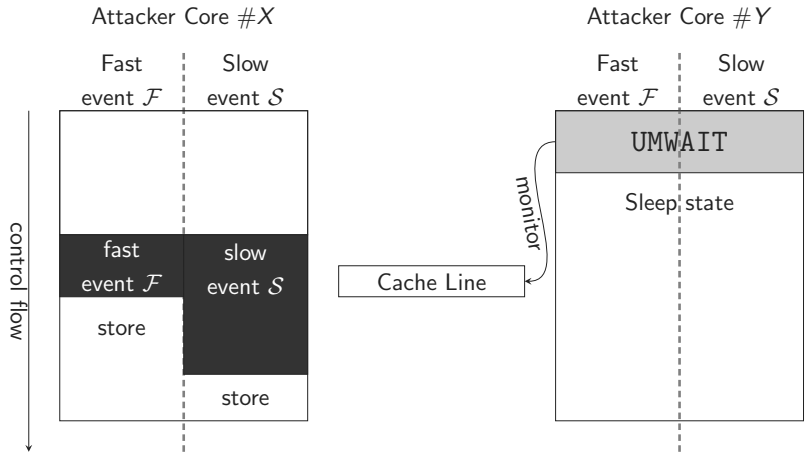
# Attack Primitive #2: Timer-Less Timing Measurement



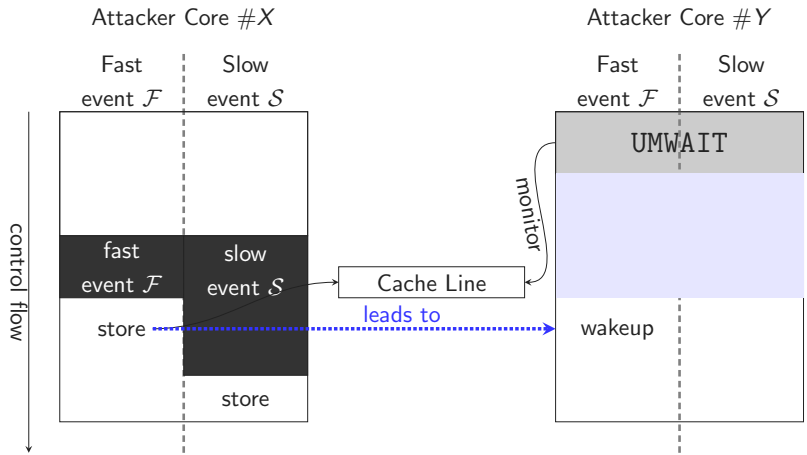
# Attack Primitive #2: Timer-Less Timing Measurement



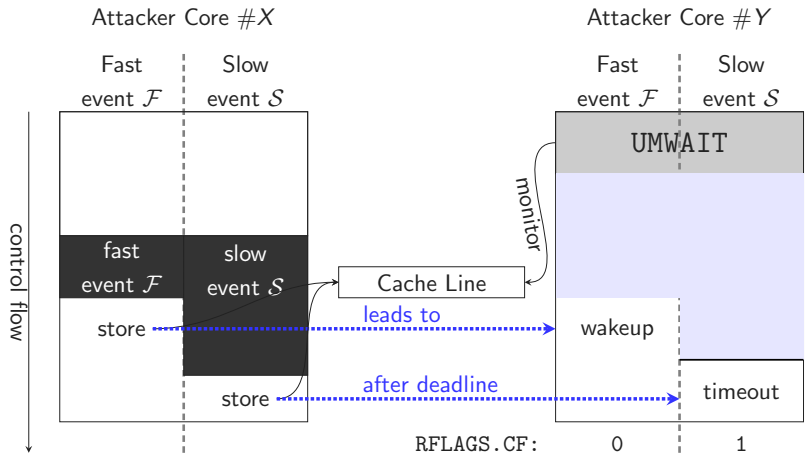
# Attack Primitive #2: Timer-Less Timing Measurement



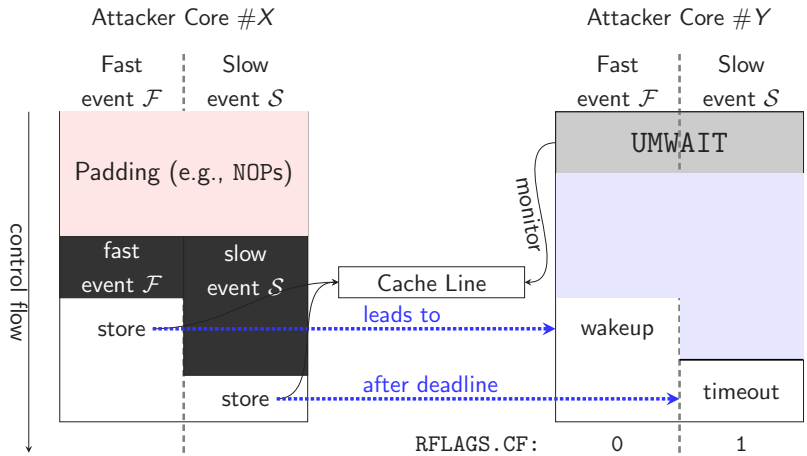
# Attack Primitive #2: Timer-Less Timing Measurement



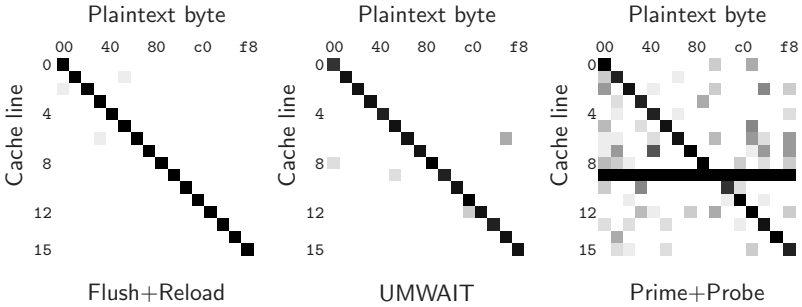
# Attack Primitive #2: Timer-Less Timing Measurement



# Attack Primitive #2: Timer-Less Timing Measurement



# AES Key Leakage



# Attack Primitive #3: Monitoring Interrupts

# Measuring Time

```
start = ⏱
end = ⏱
```

# Measuring Time

```
start = ⏱
end = ⏱
Δ = end - start
```

# Measuring Time

```
start = ⏱
end = ⏱
Δ = end - start
```

1. run: $\Delta = 12$

# Measuring Time

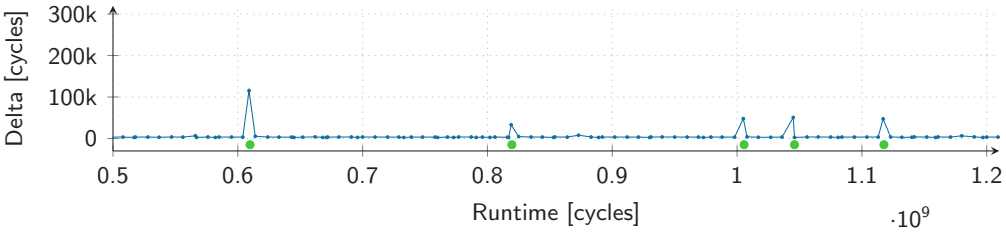
```
start = ⏱
end = ⏱
Δ = end - start
```

1. run: Δ = 12
2. run: Δ = 12

# Measuring Time

```
start = ⏱
end = ⏱
Δ = end - start
```

1. run: Δ = 12
2. run: Δ = 12



# Interrupt-timing Attacks

```
int now = rdtsc();
while (true) {
  int last = now;
  now = rdtsc();
  if ((now - last) > threshold) {
    reportEvent(now, now - last);
  }
}
```

- Schwarz, M., Lipp, M., Gruss, D., Weiser, S., Maurice, C., Spreitzer, R., & Mangard, S "Keydrown: Eliminating software-based keystroke timing side-channel attacks.; NDSS 2018

# Interrupt-timing Attacks

```
int now = rdtsc();
while (true) {
  int last = now;
  now = rdtsc();
  if ((now - last) > threshold) {
    reportEvent(now, now - last);
  }
}
```

- Continuously acquire high-resolution timestamp

Schwarz, M., Lipp, M., Gruss, D., Weiser, S., Maurice, C., Spreitzer, R., & Mangard, S "Keydrown: Eliminating software-based keystroke timing side-channel attacks.; NDSS 2018

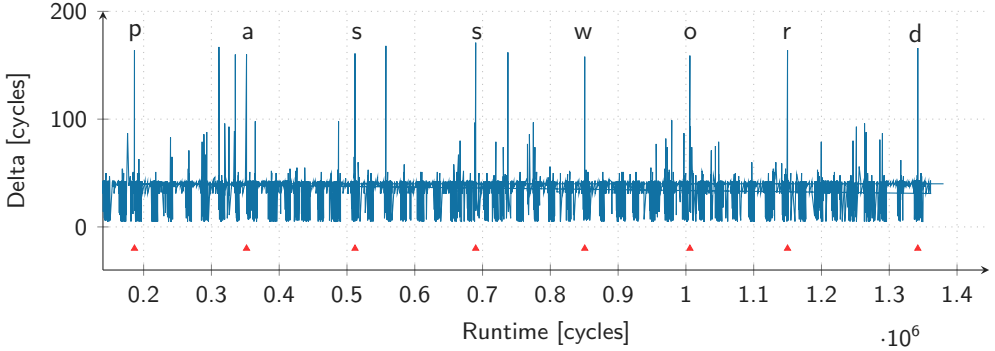
# Interrupt-timing Attacks

```
int now = rdtsc();
while (true) {
  int last = now;
  now = rdtsc();
  if ((now - last) > threshold) {
    reportEvent(now, now - last);
  }
}
```

- Continuously acquire high-resolution timestamp
- Interrupt → large difference between timestamps

Schwarz, M., Lipp, M., Gruss, D., Weiser, S., Maurice, C., Spreitzer, R., & Mangard, S "Keydrown: Eliminating software-based keystroke timing side-channel attacks.; NDSS 2018

# Interrupt-timing Attacks




Schwarz, M., Lipp, M., Gruss, D., Weiser, S., Maurice, C., Spreitzer, R., & Mangard, S "Keydrown: Eliminating software-based keystroke timing side-channel attacks.; NDSS 2018

# Attack Primitive #3: Monitoring Interrupts

- Intel: UMONITOR/UMWAIT & TPAUSE
- AMD: MONITORX/MWAITX
- ARM: WFI

# Attack Primitive #3: Monitoring Interrupts

E.g., the timeout of MWAITX is a 32-bit value → 2s for a 2 GHz CPU

App

OS

Δ

# Attack Primitive #3: Monitoring Interrupts

E.g., the timeout of MWAITX is a 32-bit value → 2s for a 2 GHz CPU

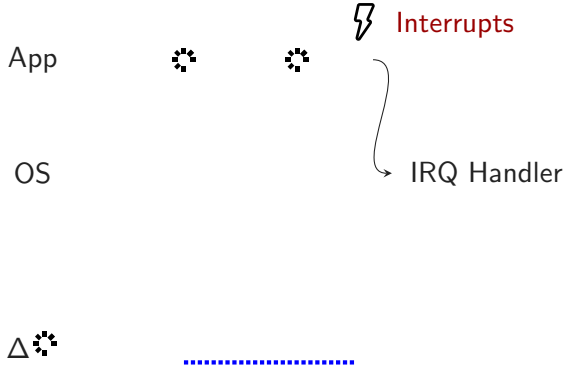
App

OS

Δ

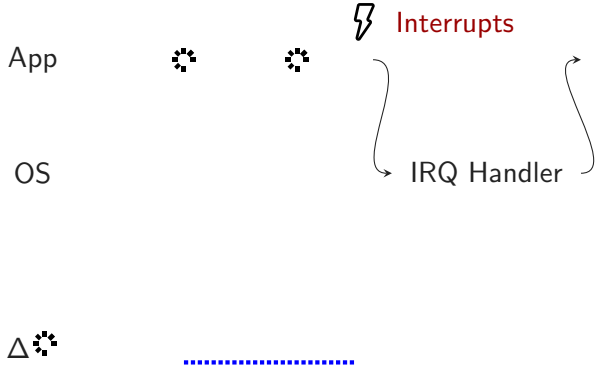
# Attack Primitive #3: Monitoring Interrupts

E.g., the timeout of MWAITX is a 32-bit value → 2s for a 2 GHz CPU



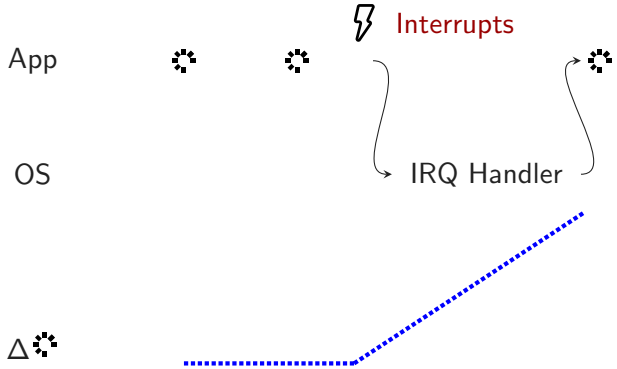
# Attack Primitive #3: Monitoring Interrupts

E.g., the timeout of MWAITX is a 32-bit value → 2s for a 2 GHz CPU



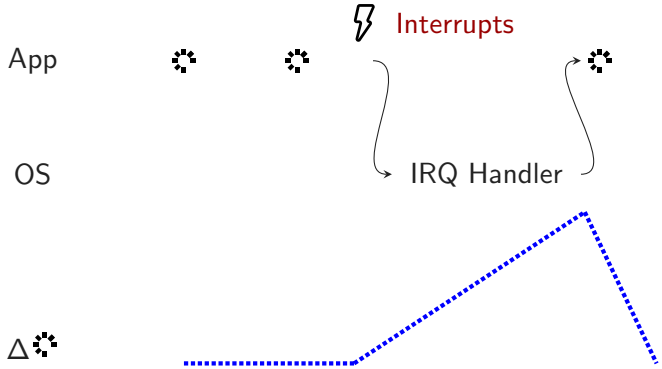
# Attack Primitive #3: Monitoring Interrupts

E.g., the timeout of MWAITX is a 32-bit value → 2s for a 2 GHz CPU



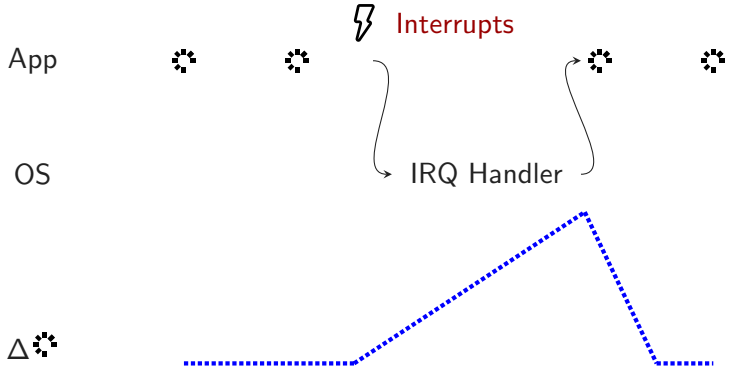
# Attack Primitive #3: Monitoring Interrupts

E.g., the timeout of MWAITX is a 32-bit value → 2s for a 2 GHz CPU

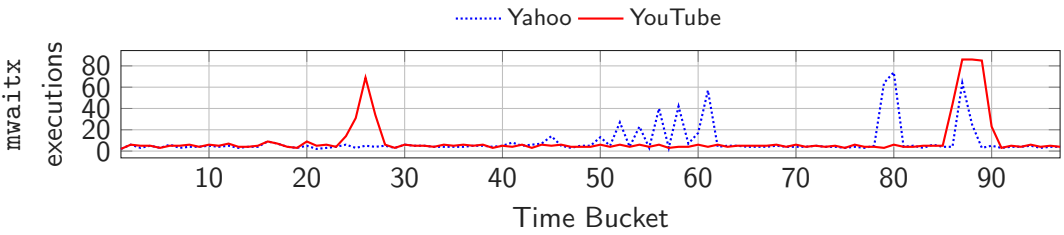


# Attack Primitive #3: Monitoring Interrupts

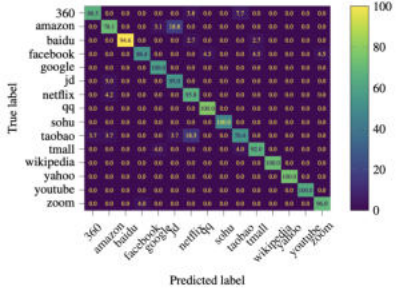
E.g., the timeout of MWAITX is a 32-bit value → 2s for a 2 GHz CPU



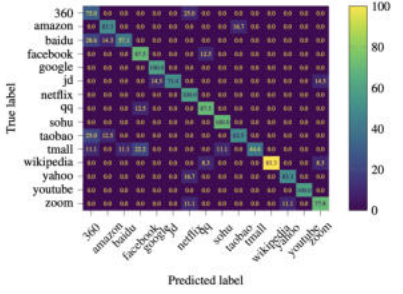
# Monitoring Network Interrupts



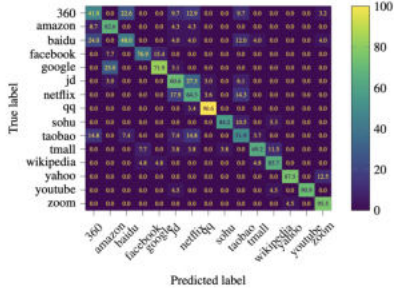
# Website Fingerprinting Attack




(a) AMD



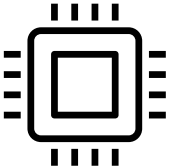

(b) ARM



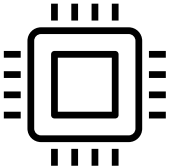

(c) Intel

# Countermeasures

# Countermeasures - Hardware

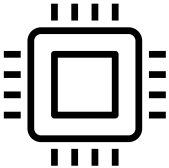


# Countermeasures - Hardware



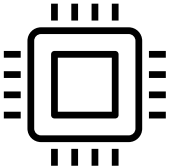
- Remove wake-up reason

# Countermeasures - Hardware



- Remove wake-up reason
- Remove functionality

# Countermeasures - Hardware



- Remove wake-up reason
- Remove functionality
- Instruction deprecation

# Countermeasures - Operating System

# Countermeasures - Operating System

- Disable in the OS

# Countermeasures - Operating System

- Disable in the OS
  - → Setting the CR4.TSD bit makes UMWAIT privileged
  - → Setting a VM-execution-control bit to trap UMWAIT

# Countermeasures - Operating System

- Disable in the OS
  - → Setting the CR4.TSD bit makes UMWAIT privileged
  - → Setting a VM-execution-control bit to trap UMWAIT
- Remove the umwait OS timeout

# Countermeasures - Software

# Countermeasures - Software

- Spectre mitigations

# Countermeasures - Software

- Spectre mitigations
- Constant-time implementations

# Countermeasures - Software

- Spectre mitigations
- Constant-time implementations
- (Introduce artificial noise)

# Take Aways

- The instructions UMONITOR/UMWAIT leak information

# Take Aways

- The instructions UMONITOR/UMWAIT leak information
- We can enhance transient-execution attacks

# Take Aways

- The instructions UMONITOR/UMWAIT leak information
- We can enhance transient-execution attacks
- We can spy on other software

# Take Aways

- The instructions UMONITOR/UMWAIT leak information
- We can enhance transient-execution attacks
- We can spy on other software

→ `https://github.com/cispa/mwait`


black hat
MIDDLE EAST AND AFRICA
(M)Wait for it
Bridging the Gap between Microarchitectural and Architectural Side Channels
15 - 17 NOVEMBER 2022
RIYADH FRONT EXHIBITION CENTRE
SAUDI ARABIA
Ruiyi Zhang, Daniel Weber, Michael Schwarz
CO-ORGANISED BY:
informatech
IN PARTNERSHIP WITH:
Riyadh