
black hat®
ASIA 2026
APRIL 21-24, 2026
MARINA BAY SANDS / SINGAPORE

Silicon Valley's Quiet Leak: Revealing User Activity on macOS for Apple Silicon

Xin Zhang*, **Zhi Zhang***, Chang Liu, Qingni Shen, Trevor E. Carlson

Accepted

Shandong Rizhao

Scan the QR code to add me as friend

Xin Zhang

PhD Student, Peking University

Incoming Faculty Member, Shandong University

zhangxin00@stu.pku.edu.cn

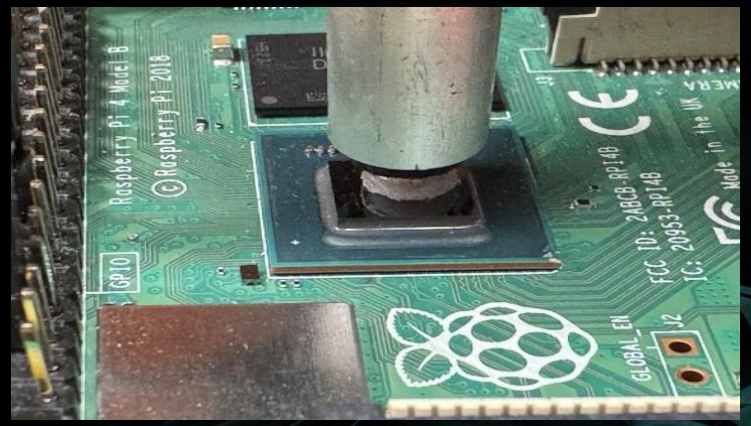
The Rest of the Team

- **Zhi Zhang (The University of Western Austrilia)**,
- Chang Liu (Tsinghua University),
- Qingni Shen (Peking University),
- Trevor E. Carlson (National University of Singapore)

Side Channels

1996

The first side channel attack that can extract RSA keys



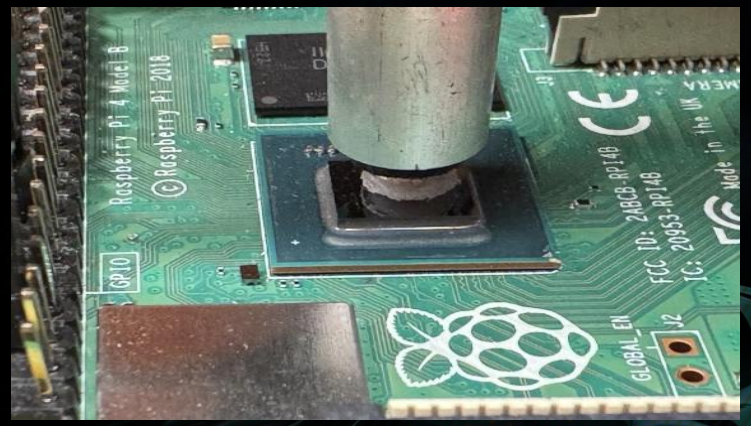
Side Channels

1996

The first side channel attack that can extract RSA keys

2018

Spectre/Meltdown show the potential of remote side channel attacks



Side Channels

1996

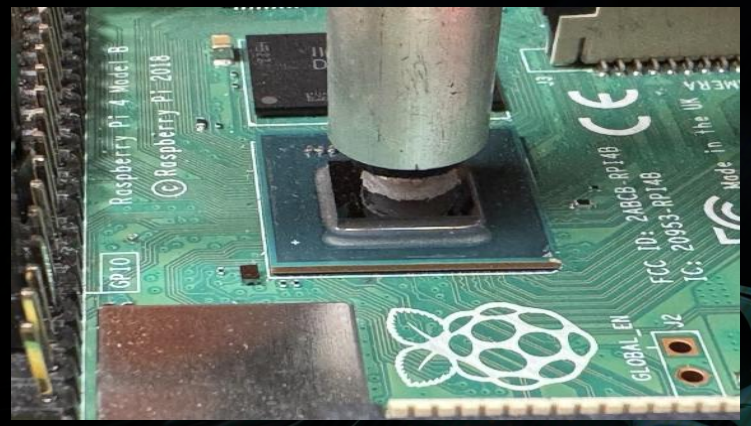
The first side channel attack that can extract RSA keys

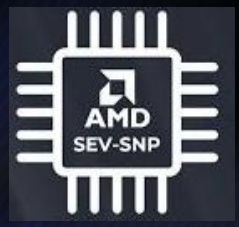
2018

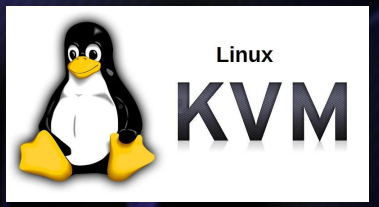
Spectre/Meltdown show the potential of remote side channel attacks

Now

Various leakage sources are identified and various information is leaked







Apple Ecosystem

Apple Ecosystem

MacBook/MacMini M1-M5

XNU Kernel

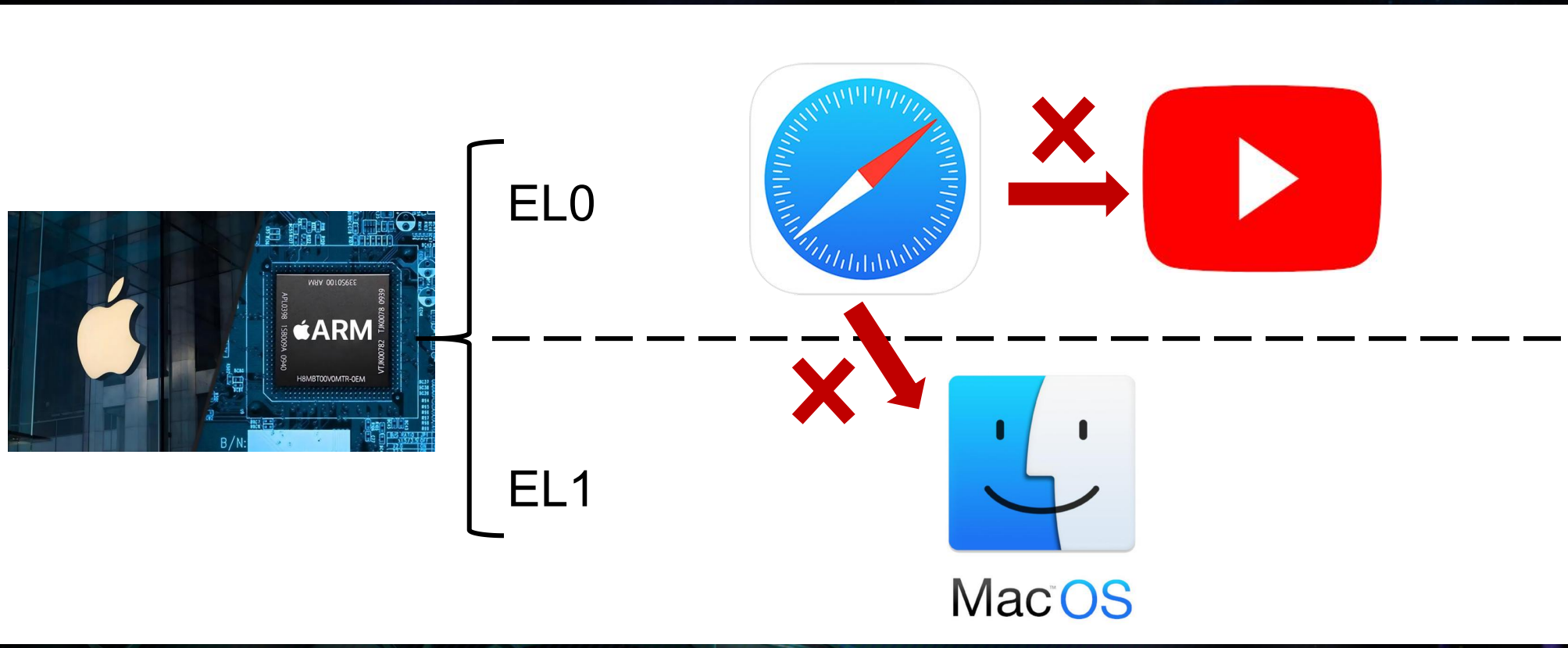
Privileges

EL0

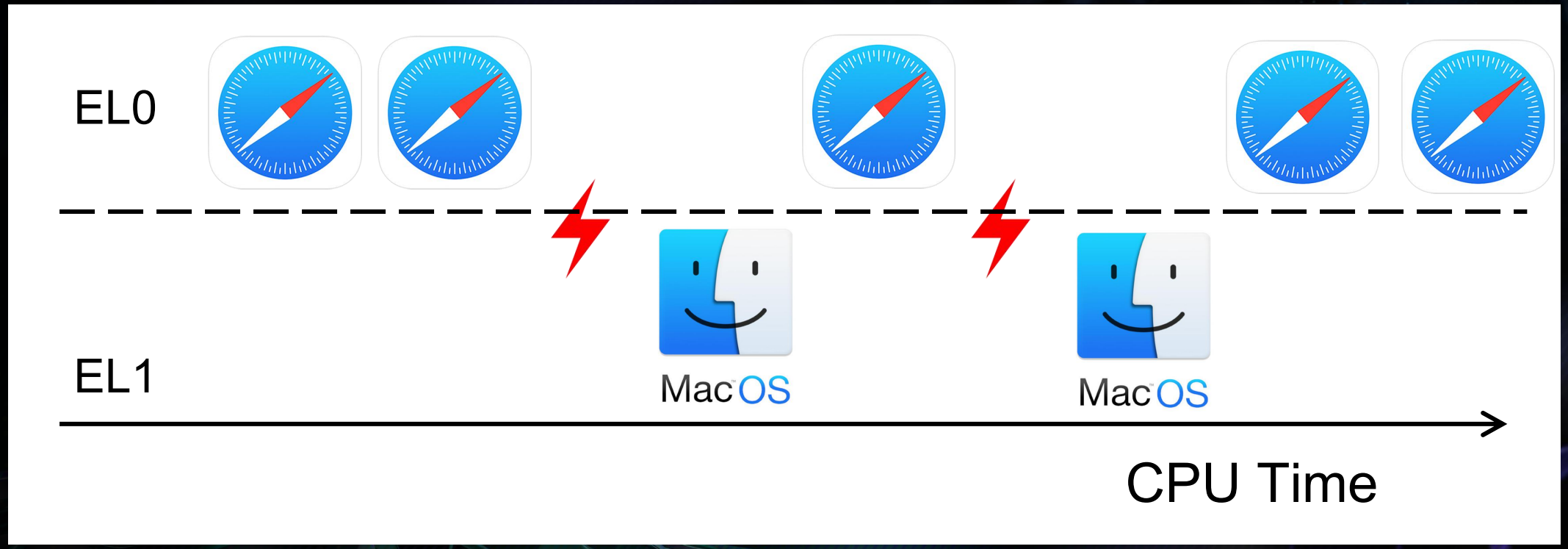
EL1

Mac OS

Privileges

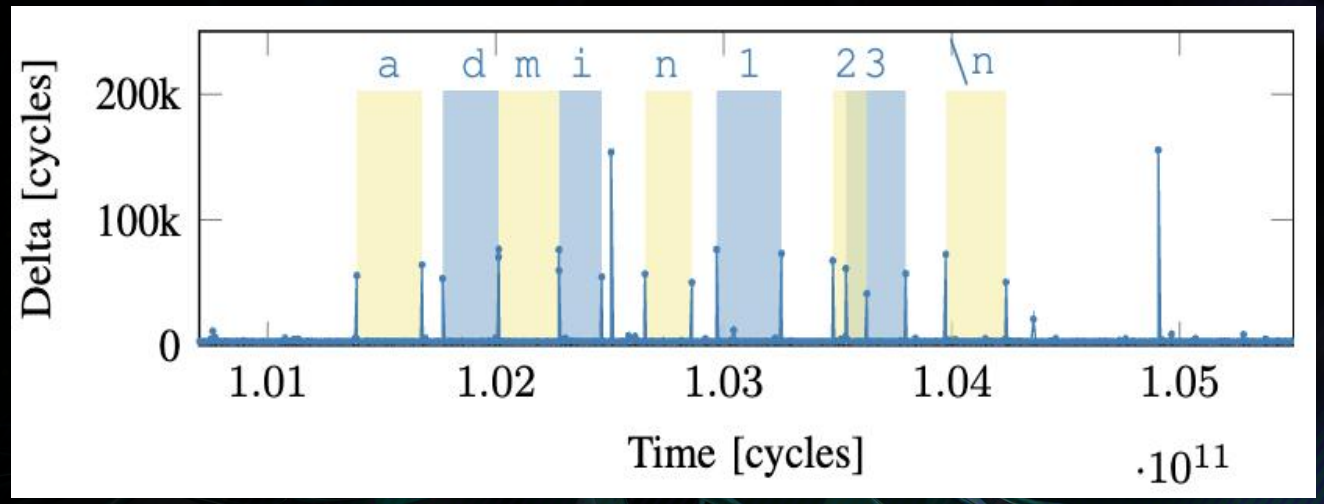


Interrupts



Interrupt Timing is Sensitive

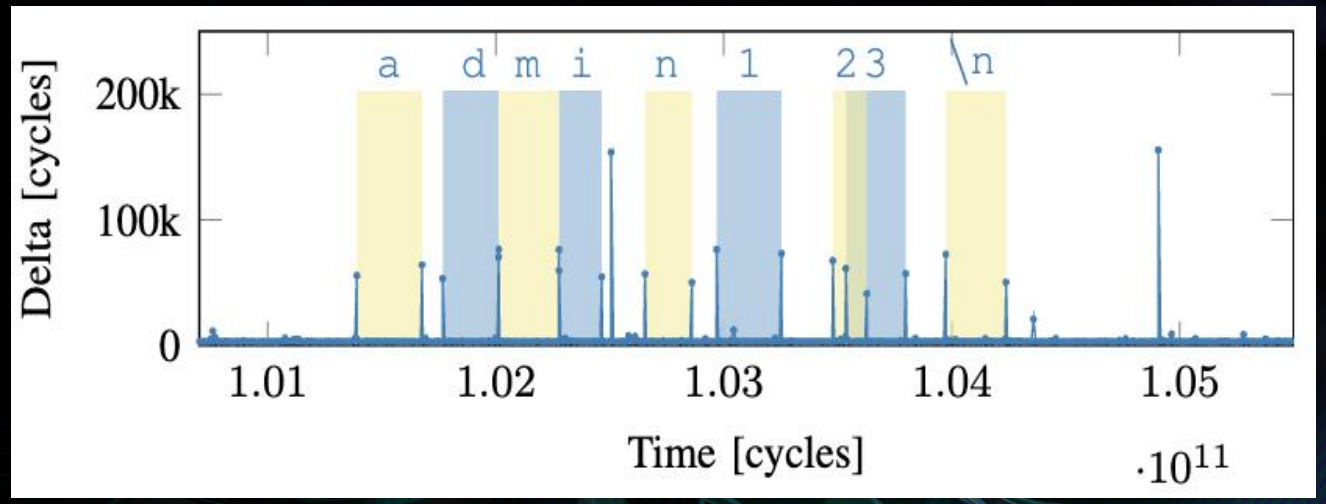
Each keystroke causes two interrupts: press and release.



Interrupt Timing is Sensitive

Each keystroke causes two interrupts: press and release.

Different inputs create different interrupt timing patterns.



Interrupt Detection

Software-based Detection

- /proc/interrupts [S&P 2016]

Hardware-based Detection

Software-based Detection

```
name          Core0      Core1      Core2      ...      Core10      Core11
  1:             92          0          0      ...          64           0  IR-IO-APIC    1-edge      i8042
  8:              0          0          0      ...           0           0  IR-IO-APIC    8-edge      rtc0
  9:           1784        387          0      ...           0           0  IR-IO-APIC    9-fasteoi   acpi
 17:           7467      10878          0      ...          50        3342  IR-IO-APIC   17-fasteoi   idma64.1, i2c_designware.1
 51:         108226          0      78802      ...           0           0  IR-IO-APIC   51-fasteoi   MSFT0001:02
162:             14          0          0      ...           0           0  IR-PCI-MSI 56623104-edge      rtsx_pci
163:          24191          0       9181      ...      104478       87088  IR-PCI-MSI 57147392-edge      iwlwifi:default_queue
177:             10          7          0      ...           0           0  IR-PCI-MSI 57147405-edge      iwlwifi:exception
178:              0          0       2048      ...           0           0  IR-PCI-MSI 514048-edge      snd_hda_intel:card0
NMI:             38       2951        320      ...         104         137  Non-maskable interrupts
LOC:        1494092    4660484    1361822      ...     1079992      412680  Local timer interrupts
SPU:              0          0          0      ...           0           0  Spurious interrupts
PMI:             38       2951        320      ...         104         137  Performance monitoring interrupts
IWI:          51545      15687      17677      ...       16880       13212  IRQ work interrupts
RTR:              0          0          0      ...           0           0  APIC ICR read retries
RES:           9782       6951       7160      ...        6412        7260  Rescheduling interrupts
CAL:         204497     107199      95661      ...       87191       82569  Function call interrupts
TLB:          29054      26963      25718      ...       22560       18894  TLB shootdowns
TRM:        6320738    6320742    6320741      ...     6320739     6320741  Thermal event interrupts
```

Software-based Detection

```
name        Core0     Core1     Core2     ...    Core10     Core11
  1:           92         0         0     ...        64          0   IR-IO-APIC    1-edge      i8042
  8:            0         0         0     ...         0          0   IR-IO-APIC    8-edge      rtc0
  9:         1784       387         0     ...         0          0   IR-IO-APIC    9-fasteoi   acpi
 17:         7467     10878         0     ...        50       3342   IR-IO-APIC   17-fasteoi   idma64.1, i2c_designware.1
 51:       108226         0     78802     ...         0          0   IR-IO-APIC   51-fasteoi   MSFT0001:02
162:           14         0         0     ...         0          0   IR-PCI-MSI 56623104-edge      rtsx_pci
163:        24191         0      9181     ...    104478      87088   IR-PCI-MSI 57147392-edge      iwlwifi:default_queue
177:           10         7         0     ...         0          0   IR-PCI-MSI 57147405-edge      iwlwifi:exception
178:            0         0      2048     ...         0          0   IR-PCI-MSI 514048-edge      snd_hda_intel:card0
NMI:           38      2951       320     ...       104        137   Non-maskable interrupts
LOC:      1494092   4660484   1361822     ...   1079992     412680   Local timer interrupts
SPU:            0         0         0     ...         0          0   Spurious interrupts
PMI:           38      2951       320     ...       104        137   Performance monitoring interrupts
IWI:        51545     15687     17677     ...     16880      13212   IRQ work interrupts
RTR:            0         0         0     ...         0          0   APIC ICR read retries
RES:         9782      6951      7160     ...      6412       7260   Rescheduling interrupts
CAL:       204497    107199     95661     ...     87191      82569   Function call interrupts
TLB:        29054     26963     25718     ...     22560      18894   TLB shootdowns
TRM:      6320738   6320742   6320741     ...   6320739    6320741   Thermal event interrupts
```

Interrupt Detection

Software-based Detection

- /proc/interrupts [S&P 2016]

Challenges: macOS does not provide such interfaces!

Hardware-based Detection

Interrupt Detection

Software-based Detection

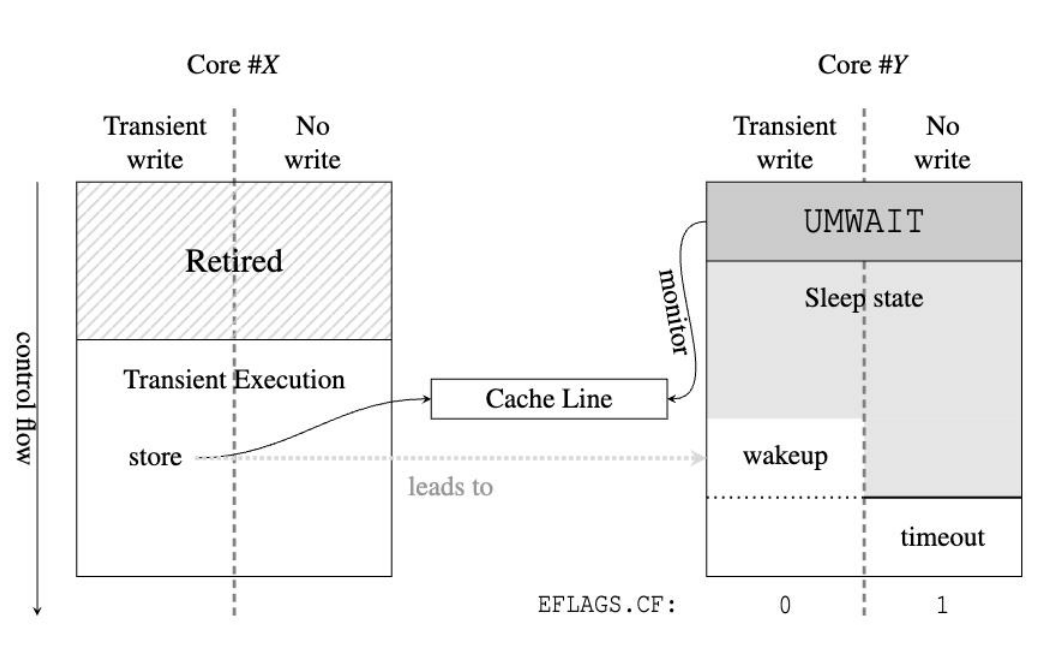
- /proc/interrupts [S&P 2016]

Challenges: macOS does not provide such interfaces!

Hardware-based Detection

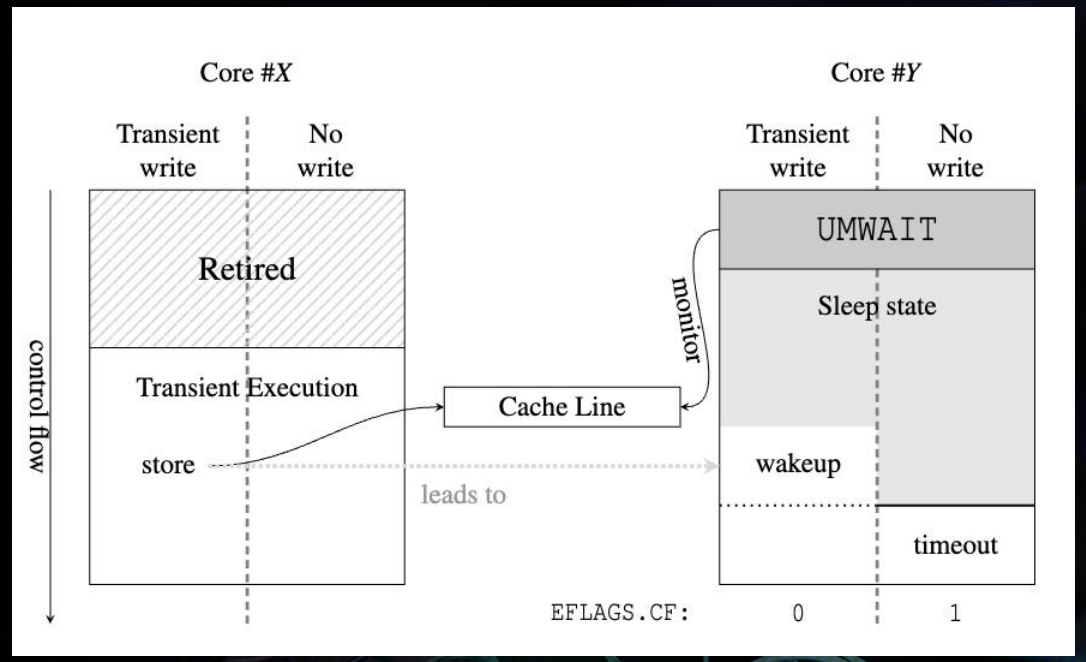
- Segment Protection [HPCA 2024]
- Mwait [SEC 2023]
- Timestamp [NDSS 2017]

Hardware-based Detection

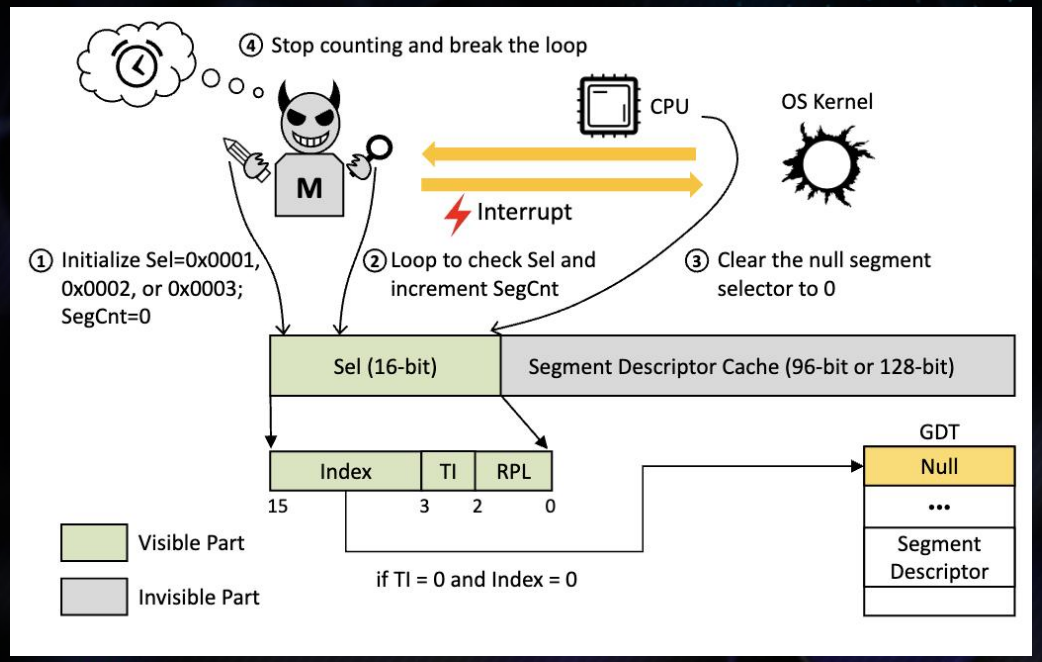


UMONITOR/UMWAIT [SEC 2023]

Hardware-based Detection

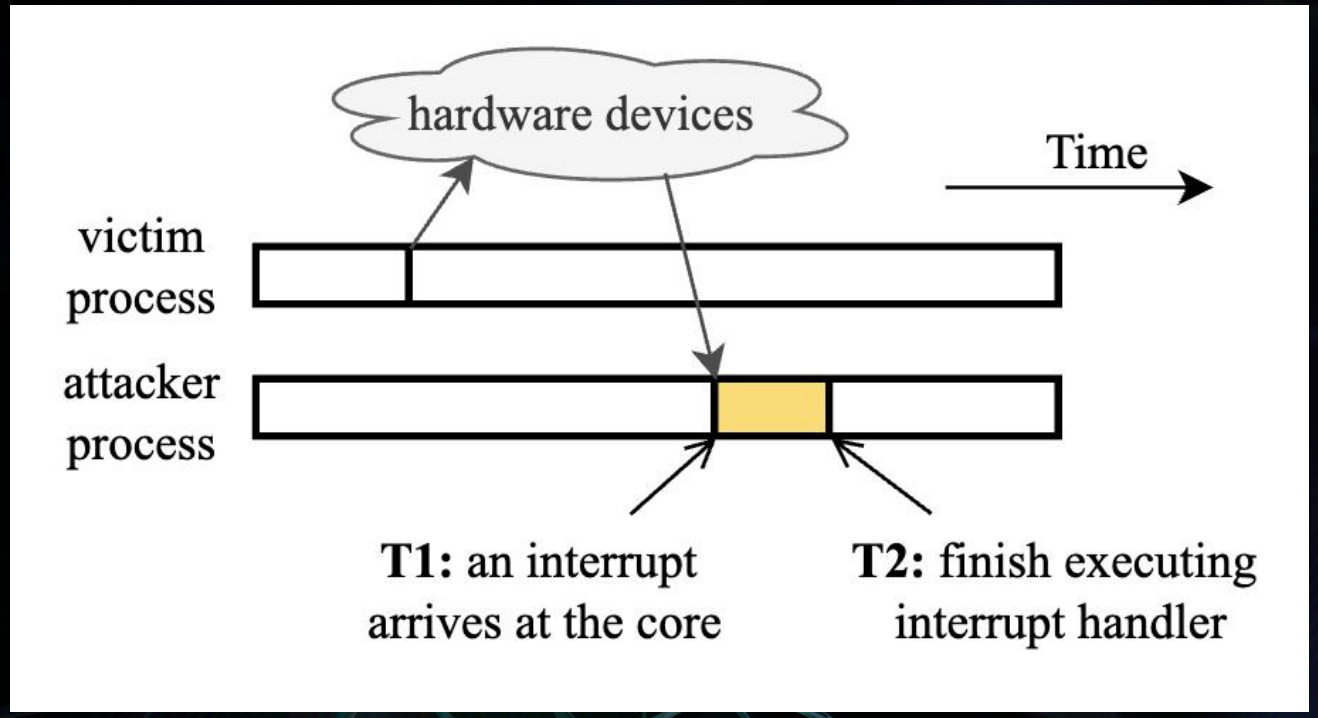


UMONITOR/UMWAIT [SEC 2023]



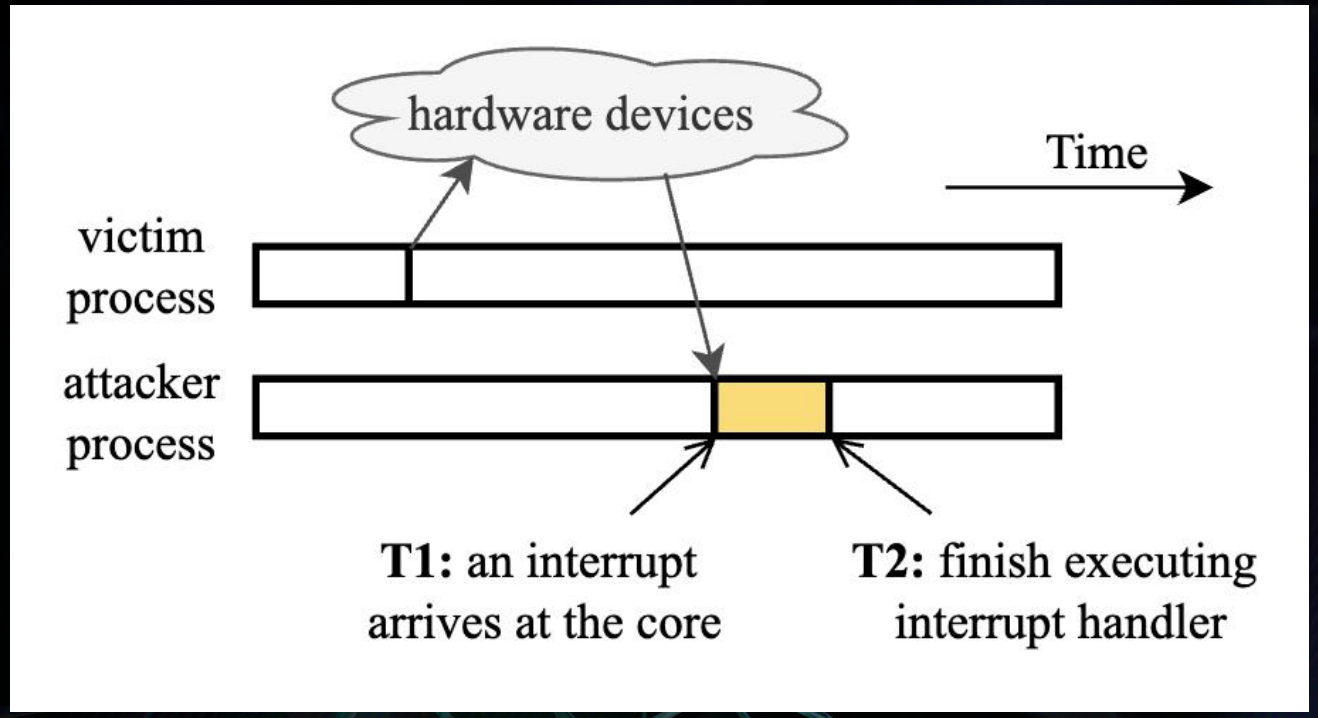
Segment Protection [HPCA 2024]

Hardware-based Detection



Timestamp [NDSS 2017]

Hardware-based Detection




```
..... y < n; .... {
tsc[i] = rdtsc();
if tsc[i] − tsc[i − 1] > threshold
    events[i] = tsc[i];
}
```

Timestamp [NDSS 2017]

Interrupt Detection

Software-based Detection

- /proc/interrupts [S&P 2016]

Challenges: macOS does not provide such interfaces!

Hardware-based Detection

- Segment Protection [HPCA 2024]
- Mwait [SEC 2023]
- Timestamp [NDSS 2017]

Challenges: x86-specific or relies on timers!

Timer-less Interrupt Detection (TIDE)

We created TIDE to enable interrupt side-channel attacks on Apple systems

Timer-less Interrupt Detection (TIDE)

We created TIDE to enable interrupt side-channel attacks on Apple systems

- Identify a deterministic ***heartbeat*** during interrupt handler routine

Timer-less Interrupt Detection (TIDE)

We created TIDE to enable interrupt side-channel attacks on Apple systems

- Identify a deterministic ***heartbeat*** during interrupt handler routine
- TIDE arises from a mitigation-oriented macOS/Arm design between Double Map, XNU's exception-entry/return, and user-visible x18.

Procedure Call Standard for the Arm 64-bit Architecture (AAPCS)

```c
int main(int argc, char **argv)
{
    uint64_t result = 0;
    uint64_t a, b, c, d, e, f, g, h;

    // Get initial values from the user
    std::cin >> a >> b >> c >> d
             >> e >> f >> g >> h;

    result = loop(a, b, c, d, e, f, g, h);

    return (int) result;
}
```


```asm
main:
        stp     x29, x30, [sp, -80]!
        mov     x29, sp
        ...
        ldr     x7, [sp, 16]
        ldr     x6, [sp, 24]
        ldr     x5, [sp, 32]
        ldr     x4, [sp, 40]
        ldr     x3, [sp, 48]
        ldr     x2, [sp, 56]
        ldr     x1, [sp, 64]
        ldr     x0, [sp, 72]
        bl      loop(unsigned long, ...)
        ldp     x29, x30, [sp], 80
        ret
```

Platform Registers

In AAPCS, x18 is the ***platform register*** and is reserved for the use of platform ABIs.

X0	1st Argument / Ret
X1	2nd Argument
X2	3rd Argument
X3	4th Argument
X4	5th Argument
X5	6th Argument
X6	7th Argument
X7	8th Argument

X8	Indirect Result
X9	1st Caller-Saved
X10	2nd Caller-Saved
X11	3rd Caller-Saved
X12	4th Caller-Saved
X13	5th Caller-Saved
X14	6th Caller-Saved
X15	7th Caller-Saved

X16	IP0
X17	IP1
X18	Platform Register
X19	1st Callee-Saved
X20	2nd Callee-Saved
X21	3rd Calle-Saved
X22	4th Callee-Saved
X23	5th Callee-Saved

X24	6th Callee-Saved
X25	7th Callee-Saved
X26	8th Callee-Saved
X27	9th Callee-Saved
X28	10th Callee-Saved
X29	Frame Pointer
X30	Link Register

Platform Registers

In AAPCS, x18 is the platform register and is reserved for the use of platform ABIs.

Apple's ABI documentation [1] says: "The platforms reserve register x18. Don’t use this register."

[1] https://developer.apple.com/documentation/xcode/writing-arm64-code-for-apple-platforms

Double Map

To mitigate Meltdown attack, Apple introduced the Double Map feature in macOS 10.13.2 and iOS 11.2 (XNU kernel 4570.31.3)

Double Map

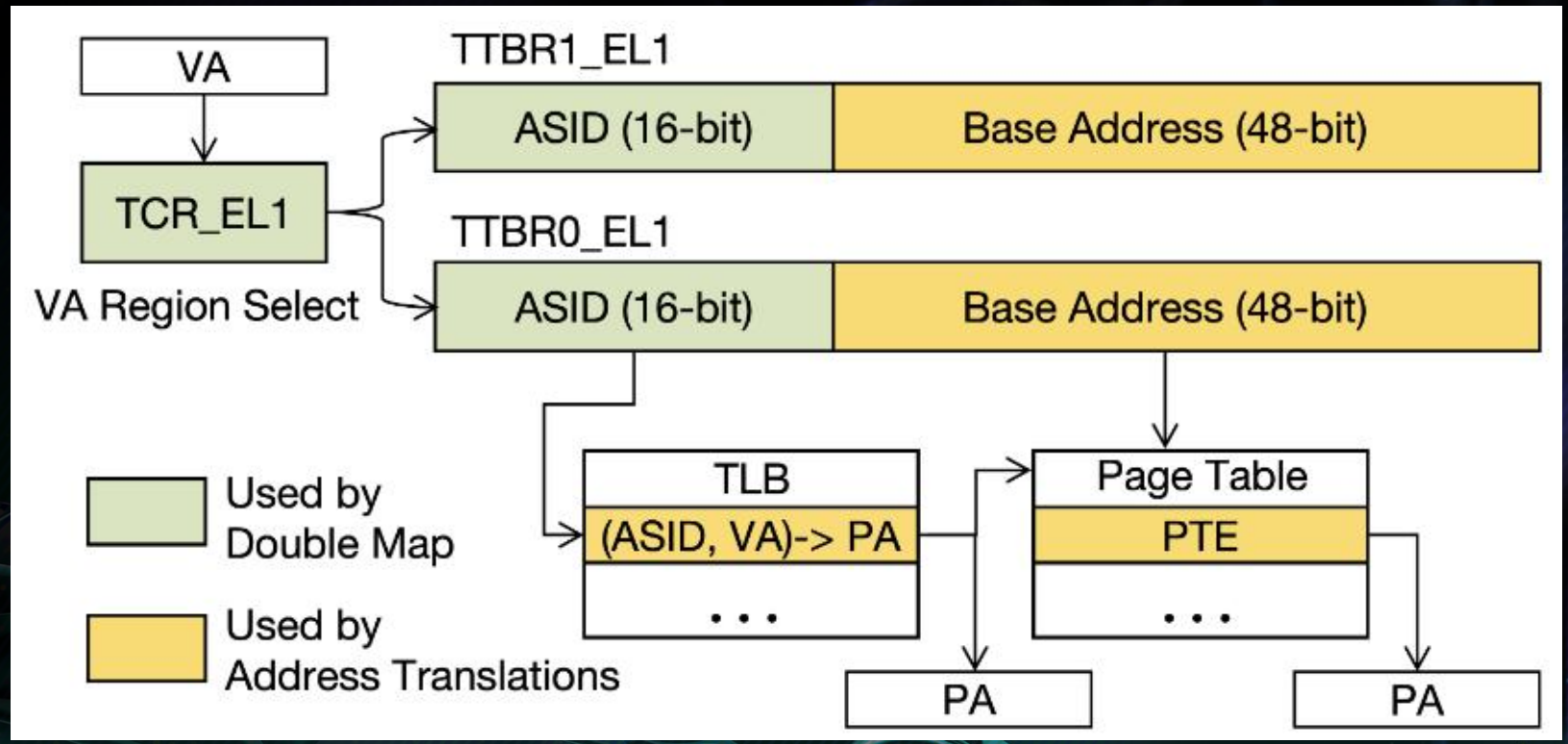
To mitigate Meltdown attack, Apple introduced the Double Map feature in macOS 10.13.2 and iOS 11.2 (XNU kernel 4570.31.3)

Double Map is implemented using two types of registers: TTBR (Translation Table Base Registers) and TCR (Translation Control Register).

TTBR controls the base address of the page tables.
TCR controls how virtual addresses are translated.

Double Map

To mitigate Meltdown attack, Apple introduced the Double Map feature in macOS 10.13.2 and iOS 11.2 (XNU kernel 4570.31.3)



Double Map

- Under Double Map, two macros are defined to be executed before accessing the kernel stack.

```
.macro MAP_KERNEL
    /* Switch to the kernel ASID for the task. */
    mrs     x18, TTBR0_EL1
    orr     x18, x18, #(1 << TTBR_ASID_SHIFT)
    msr     TTBR0_EL1, x18
    /* Update the TCR to map the kernel using the
     kernel ASID. */
    MOV64   x18, TCR_EL1_BOOT
    msr     TCR_EL1, x18
    isb     sy
.endmacro

.macro BRANCH_TO_KVA_VECTOR
    /* Find the kernelcache table for the exception
     vectors by accessing the per-CPU data. */
    mrs     x18, TPIDR_EL1
    ldr     x18, [x18, ACT_CPUDATAP]
    ldr     x18, [x18, CPU_EXC_VECTORS]
    /*Branch to the corresponding handler.*/
    ldr     x18, [x18, #($1 << 3)]
    br      x18
.endmacro
...
Lel0_irq_vector_64:
    MAP_KERNEL
    BRANCH_TO_KVA_VECTOR Lel0_irq_vector_64_long, 9
```

Double Map

- Under Double Map, two macros are defined to be executed before accessing the kernel stack.
- Both use x18 as a scratch register, thus overwriting its original value from user-space.

```
.macro MAP_KERNEL
    /* Switch to the kernel ASID for the task. */
    mrs     x18, TTBR0_EL1
    orr     x18, x18, #(1 << TTBR_ASID_SHIFT)
    msr     TTBR0_EL1, x18
    /* Update the TCR to map the kernel using the
     kernel ASID. */
    MOV64   x18, TCR_EL1_BOOT
    msr     TCR_EL1, x18
    isb     sy
.endmacro

.macro BRANCH_TO_KVA_VECTOR
    /* Find the kernelcache table for the exception
     vectors by accessing the per-CPU data. */
    mrs     x18, TPIDR_EL1
    ldr     x18, [x18, ACT_CPUDATAP]
    ldr     x18, [x18, CPU_EXC_VECTORS]
    /*Branch to the corresponding handler.*/
    ldr     x18, [x18, #($1 << 3)]
    br      x18
.endmacro
...
Lel0_irq_vector_64:
    MAP_KERNEL
    BRANCH_TO_KVA_VECTOR Lel0_irq_vector_64_long, 9
```

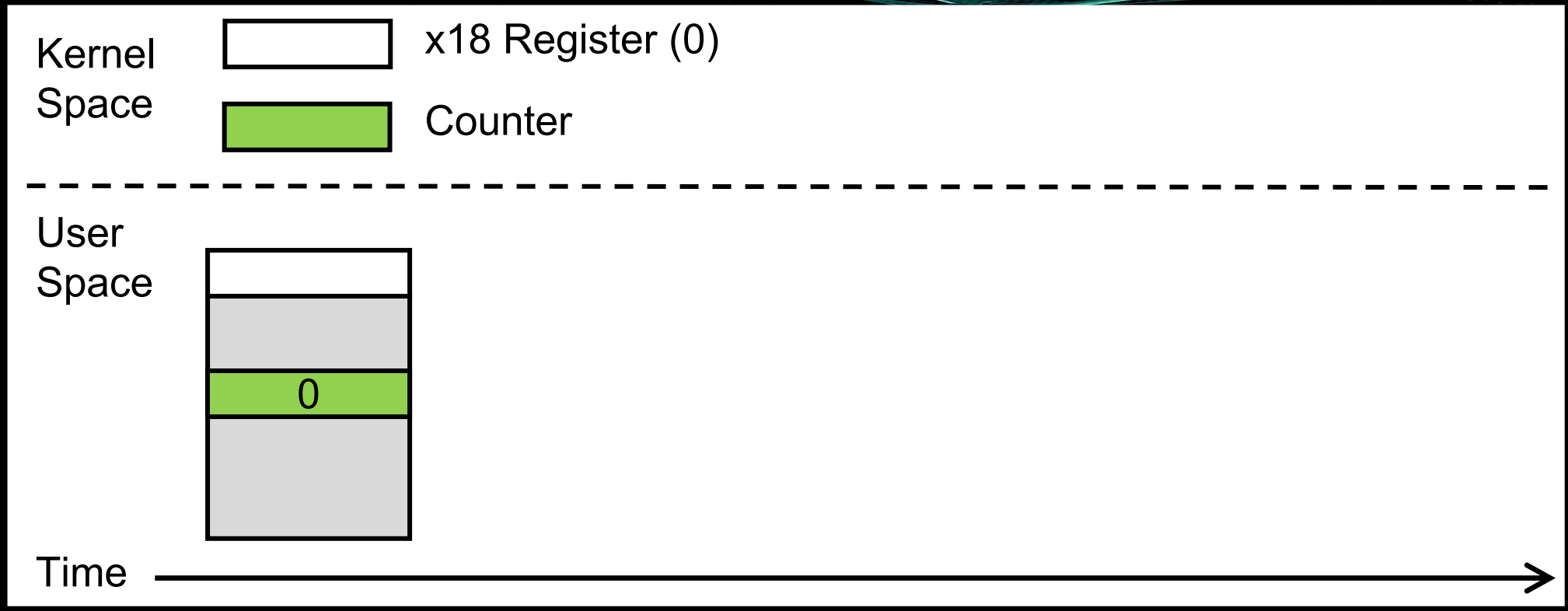
Platform Registers

As a result, the x18 register is always cleared during each context switches !

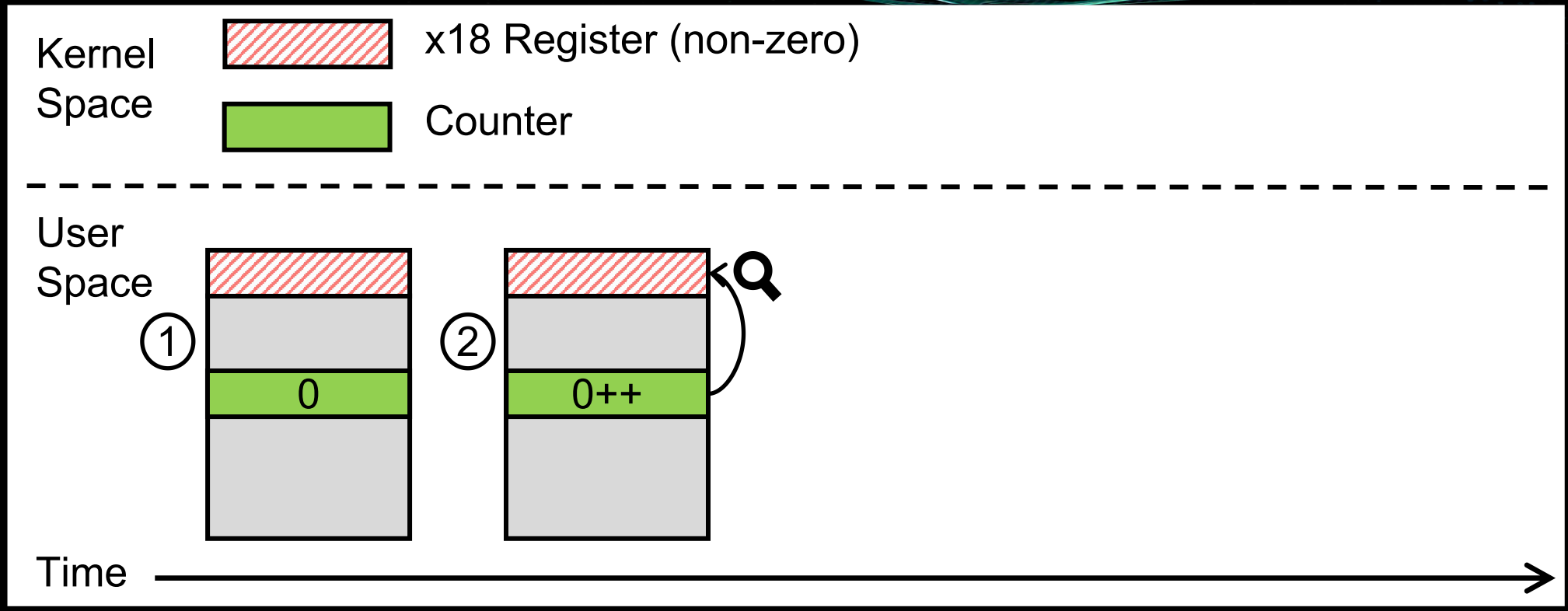
```
void set_x18(int value) {
    __asm__ volatile("mov x18, %x0" : : "r"((uint64_t)value));
}

static int get_x18() {
    uint64_t value;
    __asm__ volatile("mov %x0, x18" : "=r"(value));
    return (int)value;
}
```

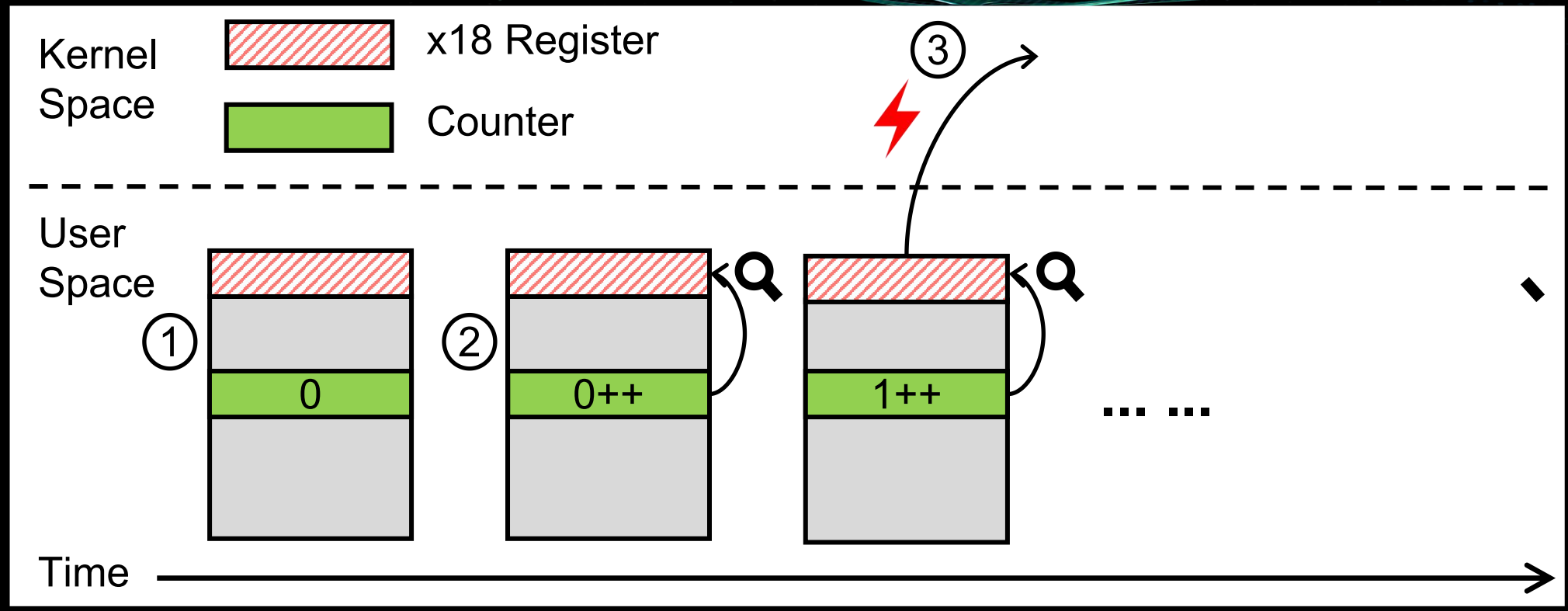
Apply TIDE to Detect Interrupt Timing



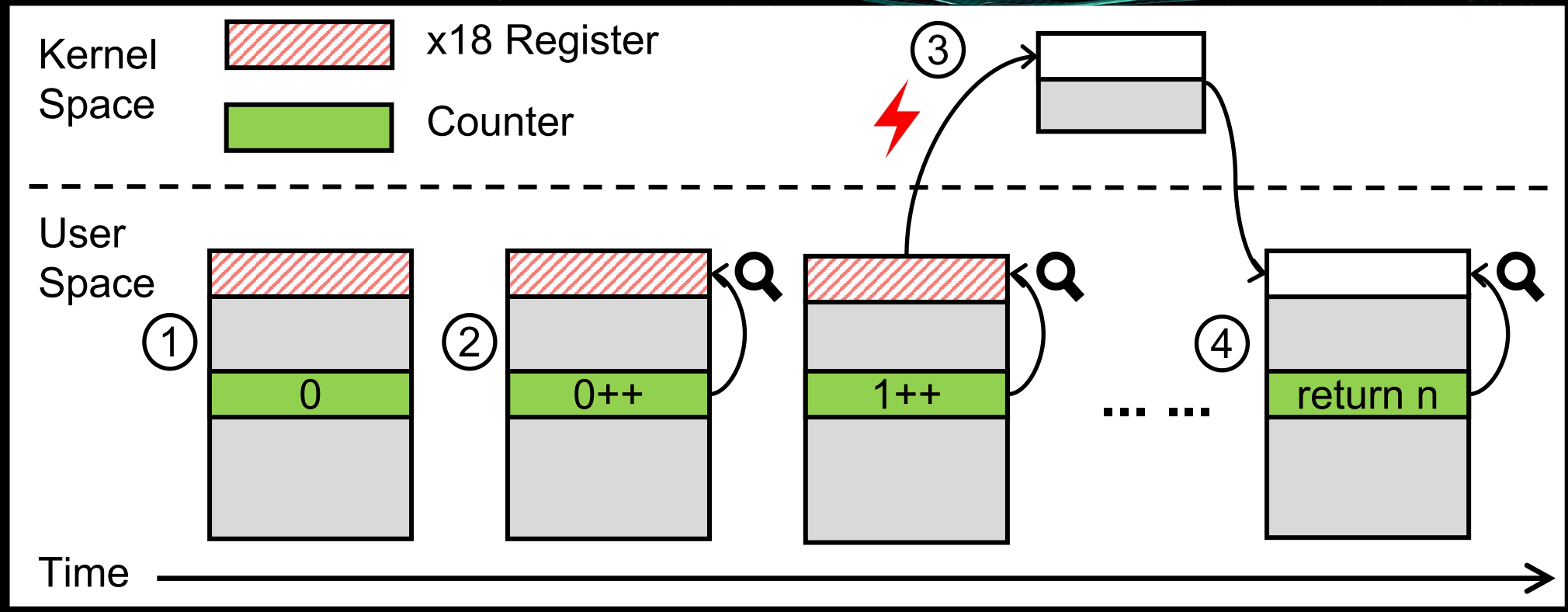
Apply TIDE to Detect Interrupt Timing



Apply TIDE to Detect Interrupt Timing

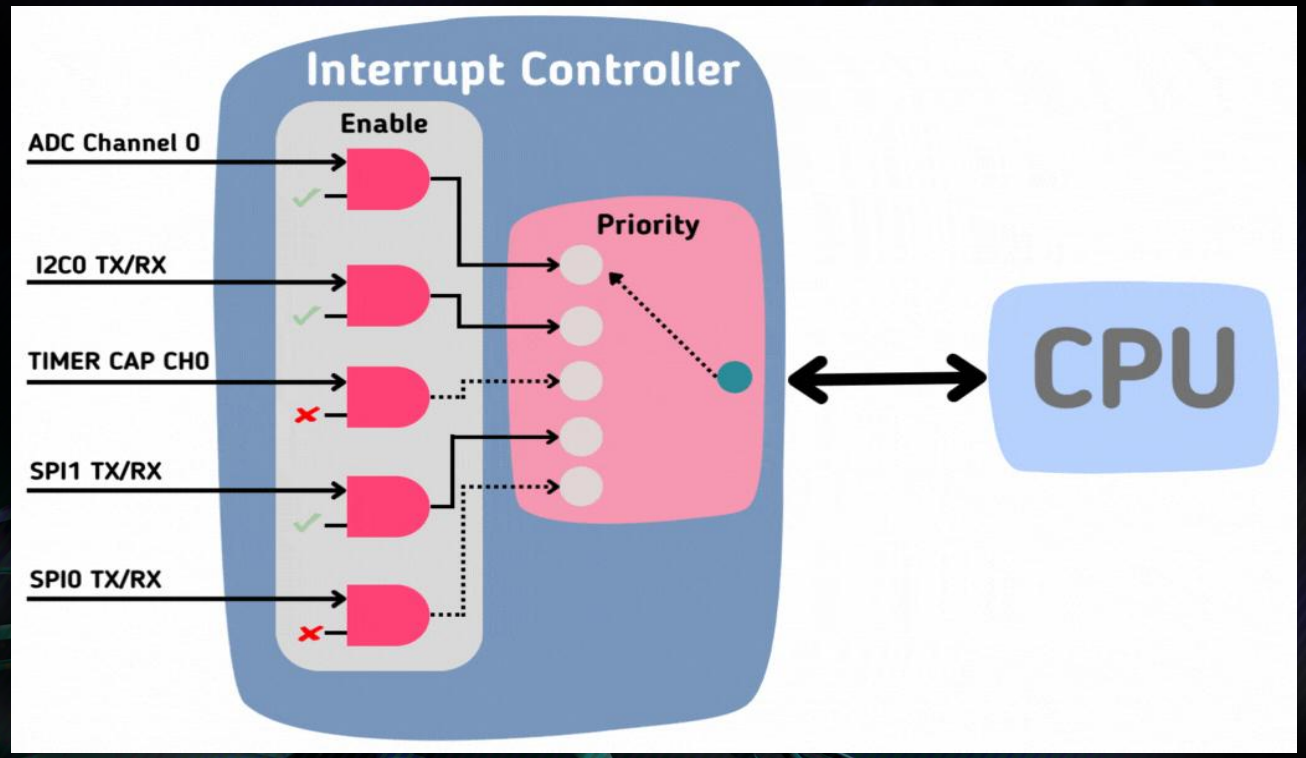


Apply TIDE to Detect Interrupt Timing



Apple Interrupt Controller

TIDE only detects interrupts on its operating core, extending it to a multi-core attack requires understanding the Apple interrupt controller.



Apple Interrupt Controller

Goals:

- Q1: How does AIC deliver shared peripheral interrupts?
- Q2: What factors can affect interrupt delivery decisions?

https://github.com/junjie1475/MacOS_CoreBinder

Apple Interrupt Controller

Goals:

- Q1: How does AIC deliver shared peripheral interrupts?
- Q2: What factors can affect interrupt delivery decisions?

Efficiency: To quantify how often interrupts are received under a certain setting, we defined the efficiency metric as the ratio between the number of SPI requests generated and the number of interrupts successfully received.

Apple Interrupt Controller

Goals:

- Q1: How does AIC deliver shared peripheral interrupts?
- Q2: What factors can affect interrupt delivery decisions?

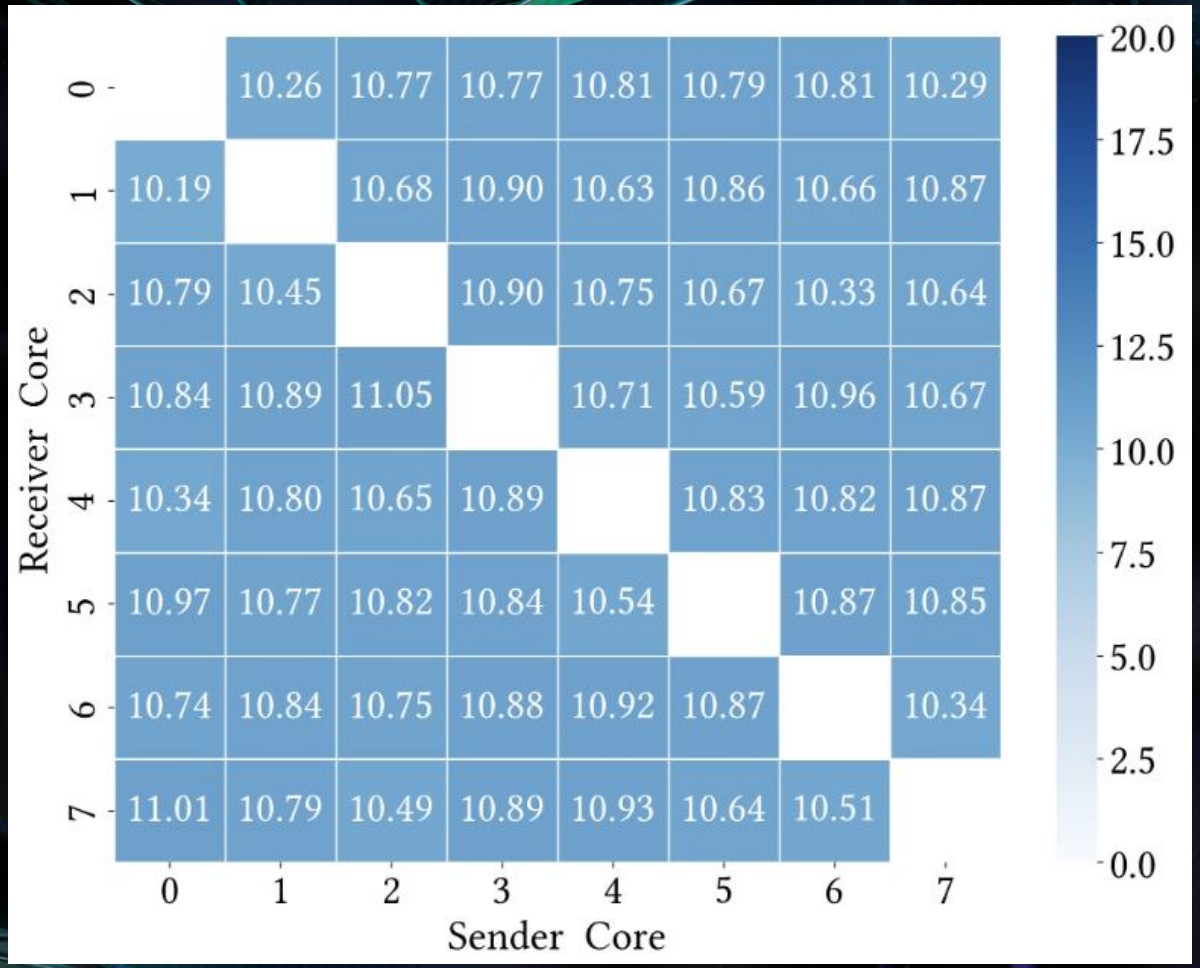
Efficiency: To quantify how often interrupts are received under a certain setting, we defined the efficiency metric as the ratio between the number of SPI requests generated and the number of interrupts successfully received.

We employ CoreBinder, a kernel extension that allows pinning threads to specific CPU cores on Apple silicon.

https://github.com/junjie1475/MacOS_CoreBinder

Apple Interrupt Delivery

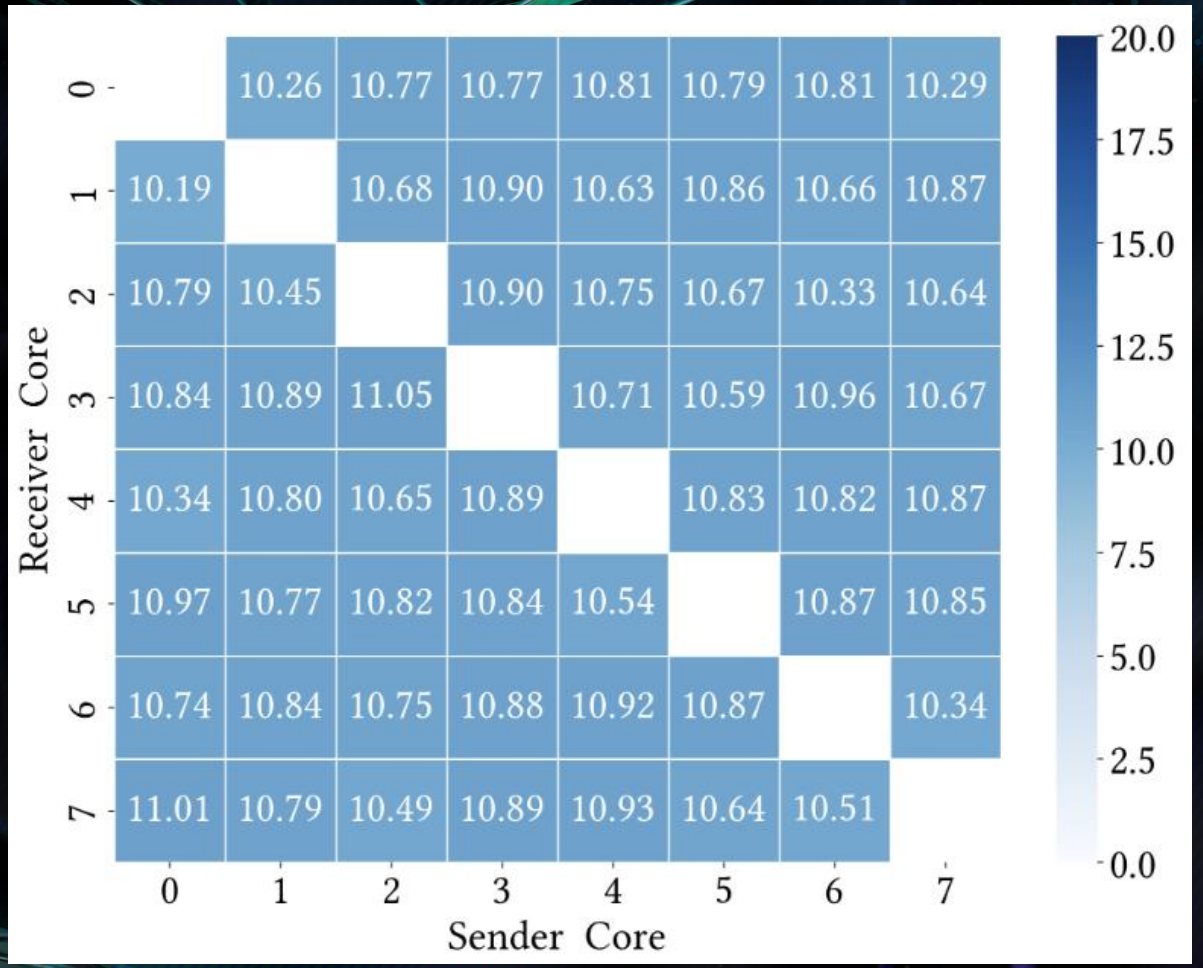
The figure shows the efficiency matrix when we bind sender and receiver to specific cores. Each cell corresponds to one pair of cores.



Apple Interrupt Delivery

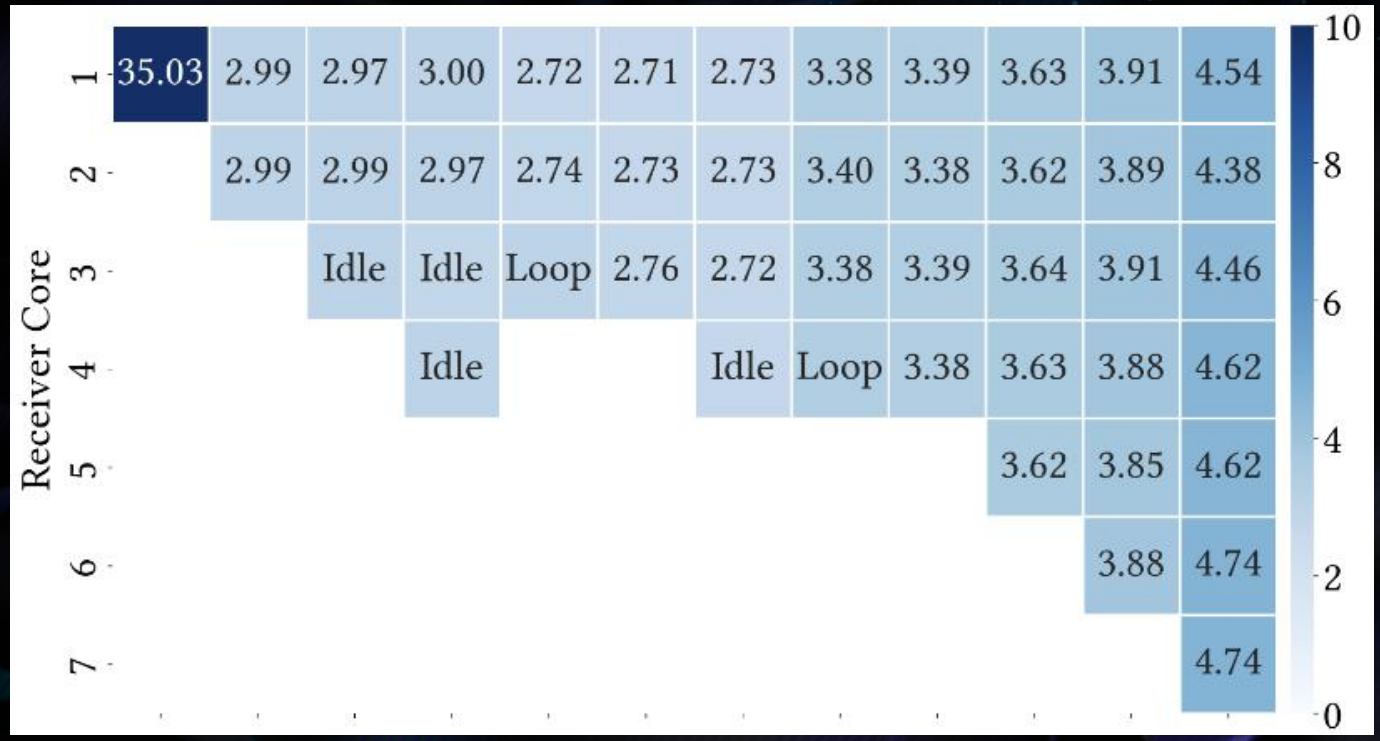
The figure shows the efficiency matrix when we bind sender and receiver to specific cores. Each cell corresponds to one pair of cores.

Unlike Linux-based systems, the combination of Apple silicon and macOS does not deliver shared peripheral interrupts to a fixed core.



Apple Interrupt Delivery

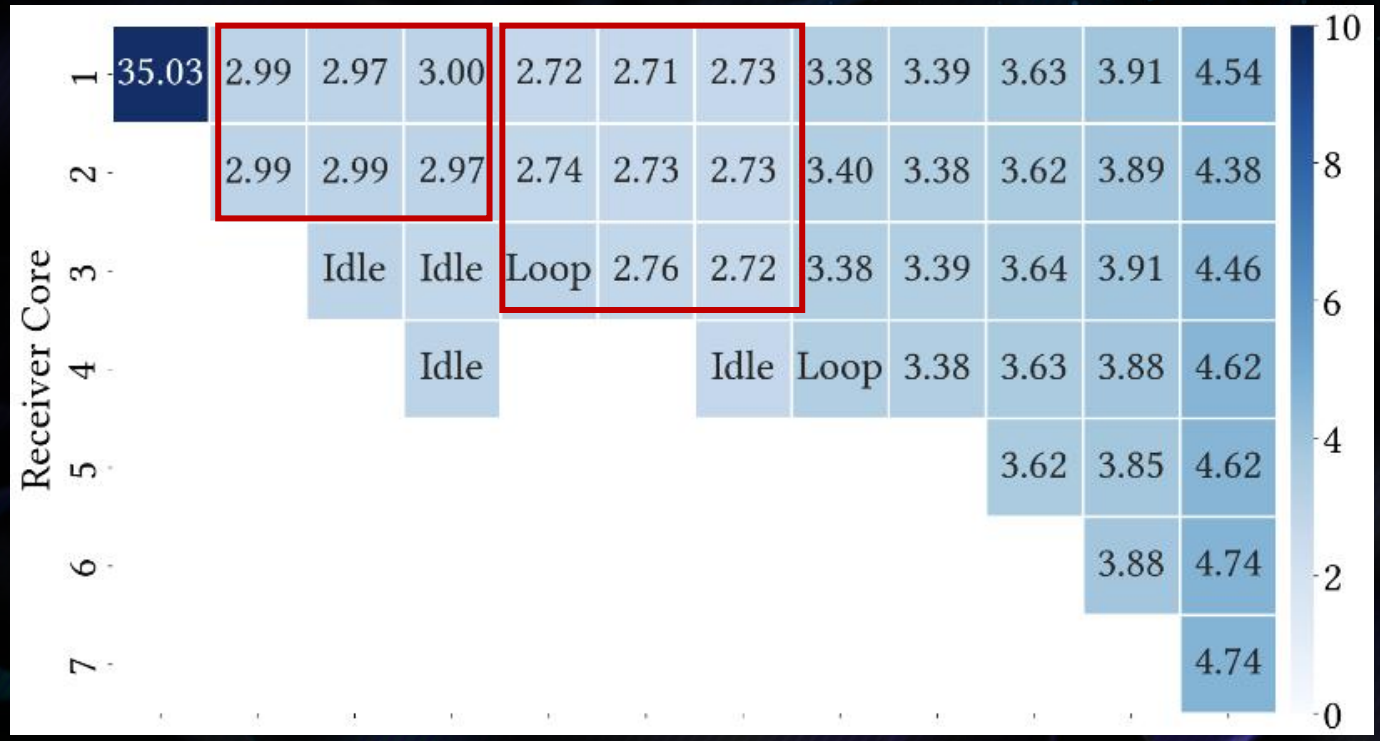
The heatmap shows the interrupt efficiency under different configurations.



Apple Interrupt Delivery

The heatmap shows the interrupt efficiency under different configurations.

Apple silicon with macOS uniformly delivers SPIs to all active cores and ignores the idle cores.



Case Studies

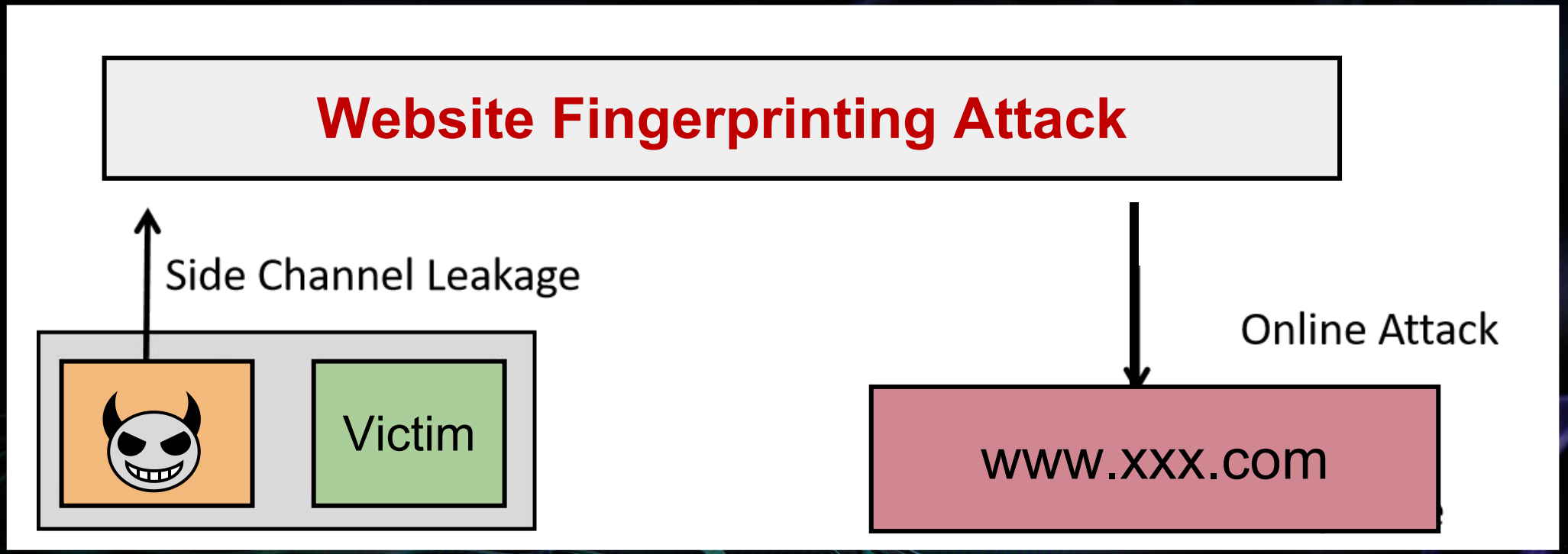
- Website Fingerprinting
- Video Fingerprinting

Website Fingerprinting

When accessing a website, distinctive network packets will be sent and received, which generates distinctive network interrupts.

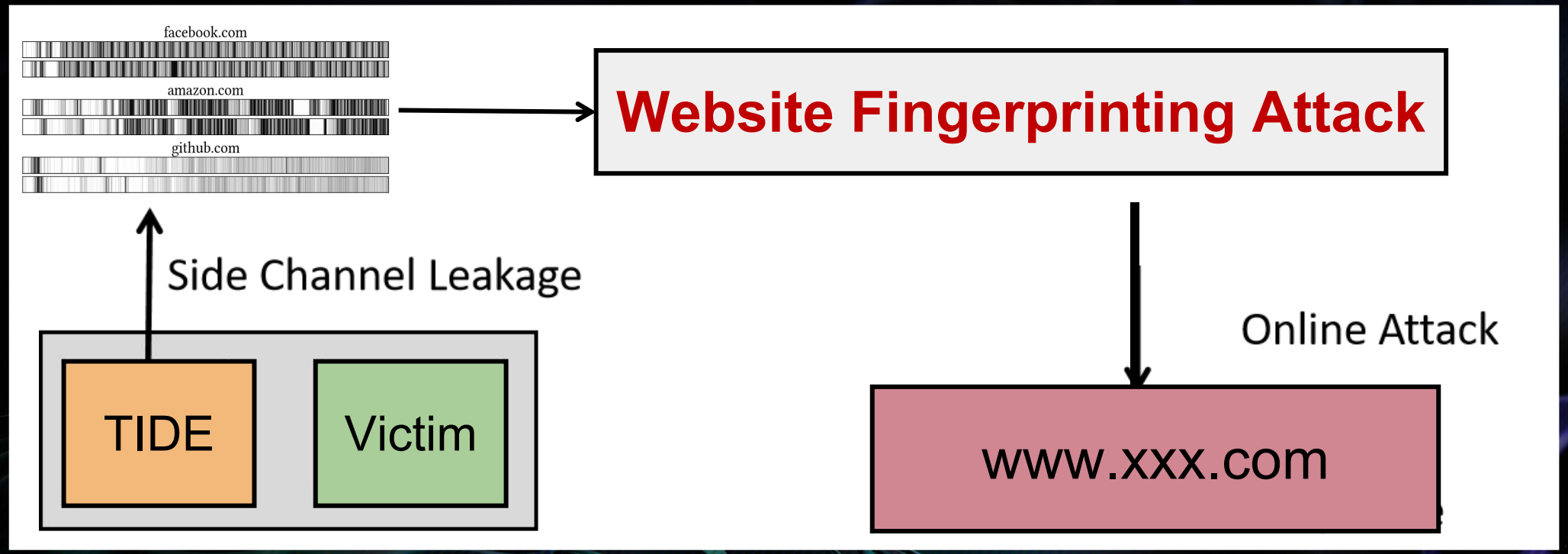
Website fingerprinting

Our fingerprinting attack aims to distinguish which website the victim is accessing.



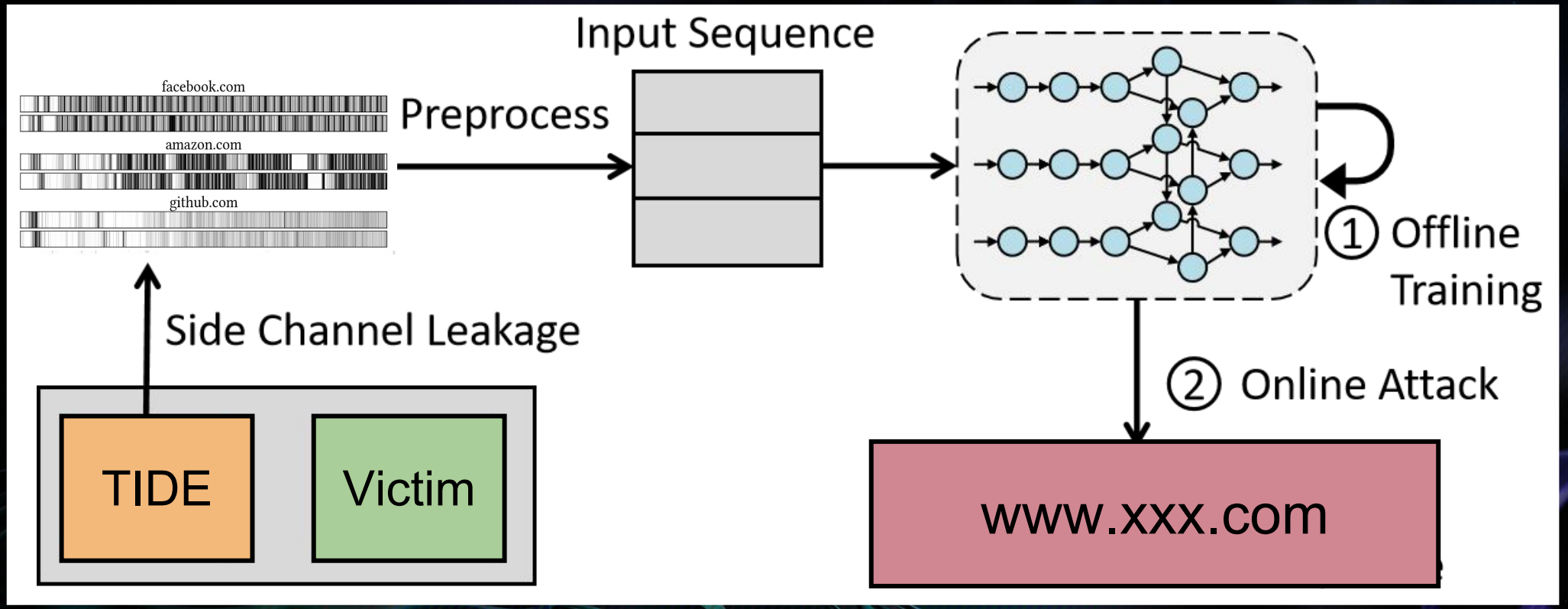
Website fingerprinting

Our fingerprinting attack aims to distinguish which website the victim is accessing.



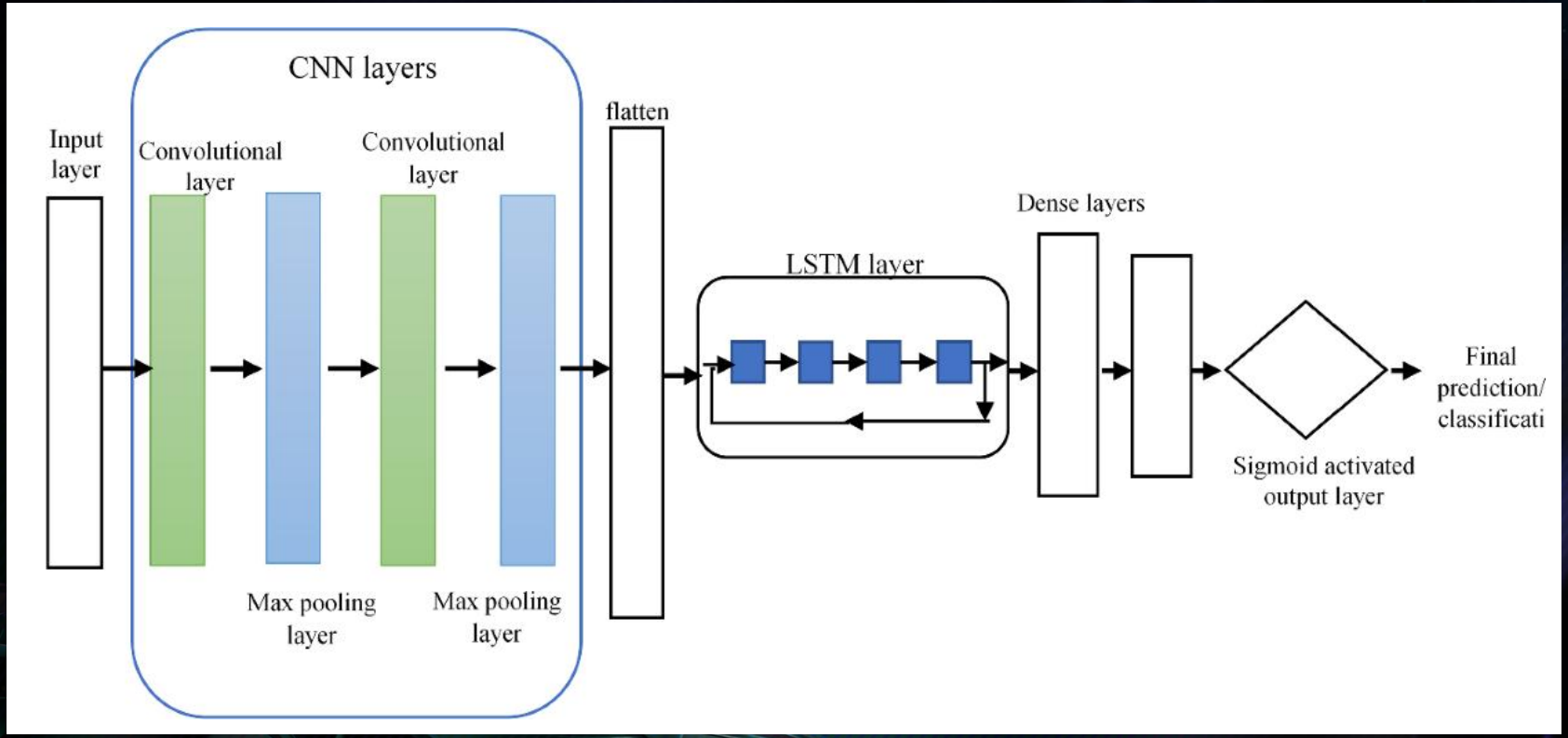
Website fingerprinting

Our fingerprinting attack aims to distinguish which website the victim is accessing.



Website fingerprinting

We uses a CNN-LSTM model to achieve the classification task.



https://github.com/jackcook/bigger-fish/tree/main

Website Fingerprinting

For closed-world website fingerprinting, we consider Alexa top-100 websites.

For the open-world setting, we consider an additional case that the victim accesses a non top-100 websites.

Website Fingerprinting

For closed-world website fingerprinting, we consider Alexa top-100 websites.

For the open-world setting, we consider an additional case that the victim accesses a non top-100 websites.

Attack Scenario	Default		Fixed Frequency	
	Top-1 Acc	Top-5 Acc	Top-1 Acc	Top-5 Acc
Closed-world Website FP	93.8%	98.7%	93.1%	98.5%
Open-world Website FP	91.5%	98.5%	91.1%	98.3%

Video Fingerprinting

- For video fingerprinting, we consider Youtube top 20 videos
- The attack routine and ML classifier are consistent with website fingerprinting

Attack Scenario	Default		Fixed Frequency	
	Top-1 Acc	Top-5 Acc	Top-1 Acc	Top-5 Acc
Closed-world Website FP	93.8%	98.7%	93.1%	98.5%
Open-world Website FP	91.5%	98.5%	91.1%	98.3%
Video FP	78.1%	97.9%	87.2%	98.3%

How widespread is the issue?

XNU kernel + Apple's Arm Silicon:

- macBook M1 - M5, macBook Neo
- macMini M1 - M5
- iPhone (we validated iPhone 16 Pro)

Vulnerability Disclosure

- On 22-January-2025, we disclosed our findings to Apple’s hardware security team.
- On 2-February-2025, we disclosed our findings to Apple’s macOS security team.

Vulnerability Disclosure

- On 22-January-2025, we disclosed our findings to Apple’s hardware security team.
- On 2-February-2025, we disclosed our findings to Apple’s macOS security team.
- On 2-February-2025, the hardware security team responded on and stated: *We have forwarded your report to the appropriate team to investigate as a potential future hardening effort.*

Vulnerability Disclosure

- On 22-January-2025, we disclosed our findings to Apple's hardware security team.
- On 2-February-2025, we disclosed our findings to Apple's macOS security team.
- On 2-February-2025, the hardware security team responded on and stated: *We have forwarded your report to the appropriate team to investigate as a potential future hardening effort.*
- On 15-March-2025, the macOS security team requested a copy of our manuscript

Vulnerability Disclosure

- On 22-January-2025, we disclosed our findings to Apple’s hardware security team.
- On 2-February-2025, we disclosed our findings to Apple’s macOS security team.
- On 2-February-2025, the hardware security team responded on and stated: *We have forwarded your report to the appropriate team to investigate as a potential future hardening effort.*
- On 15-March-2025, the macOS security team requested a copy of our manuscript
- On 9-April-2025, the macOS security team commented: *Our engineers found your paper intriguing, and it helped shed more light on the side-channel techniques you describe. This helps with the team's considerations for possible platform enhancements.*

Mitigations

- Fully mitigating TIDE can be achieved by preserving the user's original x18 value.

93.8% -> 0%

Mitigations

- Fully mitigating TIDE can be achieved by preserving the user's original x18 value.

93.8% -> 0%

- Injecting artificial jitter can reduce the signal strength, which serves as a general mitigation strategy.

93.8% -> 61.8%

What Comes Next

- Timer defenses alone are insufficient to mitigate side-channel attacks. We should pay more attention for timer-less attacks!

What Comes Next

- Timer defenses alone are insufficient to mitigate side-channel attacks. We should pay more attention for timer-less attacks!

- Closed ecosystems should not be treated as automatically secure. Even in a highly controlled platform, hidden leakage channels may still exist.

What Comes Next

- Timer defenses alone are insufficient to mitigate side-channel attacks. We should pay more attention for timer-less attacks!

- Closed ecosystems should not be treated as automatically secure. Even in a highly controlled platform, hidden leakage channels may still exist.

- Companies could benefit from more open evaluation and more careful checking of the systems they build and we use.

Silicon Valley's Quiet Leak: Revealing User Activity on macOS for Apple Silicon

Thanks for your attention!