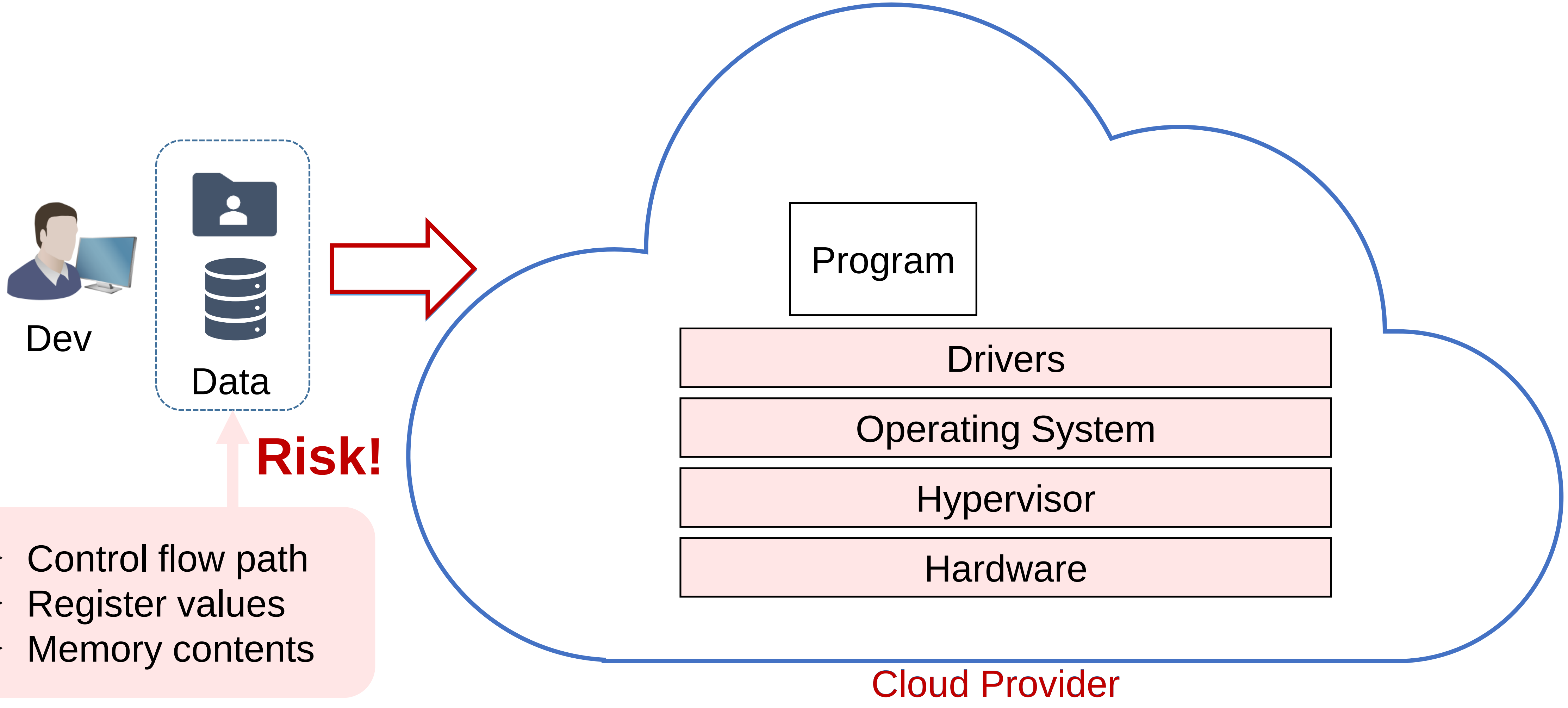
SGX lingers !

A New Side-channel Attack Vector Based on Interrupt Latency against Enclave Execution

Wenjian He	*Hong Kong Univ. of Science and Technology*
Wei Zhang	*Hong Kong Univ. of Science and Technology*
Sanjeev Das	*Univ. of North Carolina at Chapel Hill, USA*
Yang Liu	*Nanyang Technological Univ., Singapore*

Cloud Security

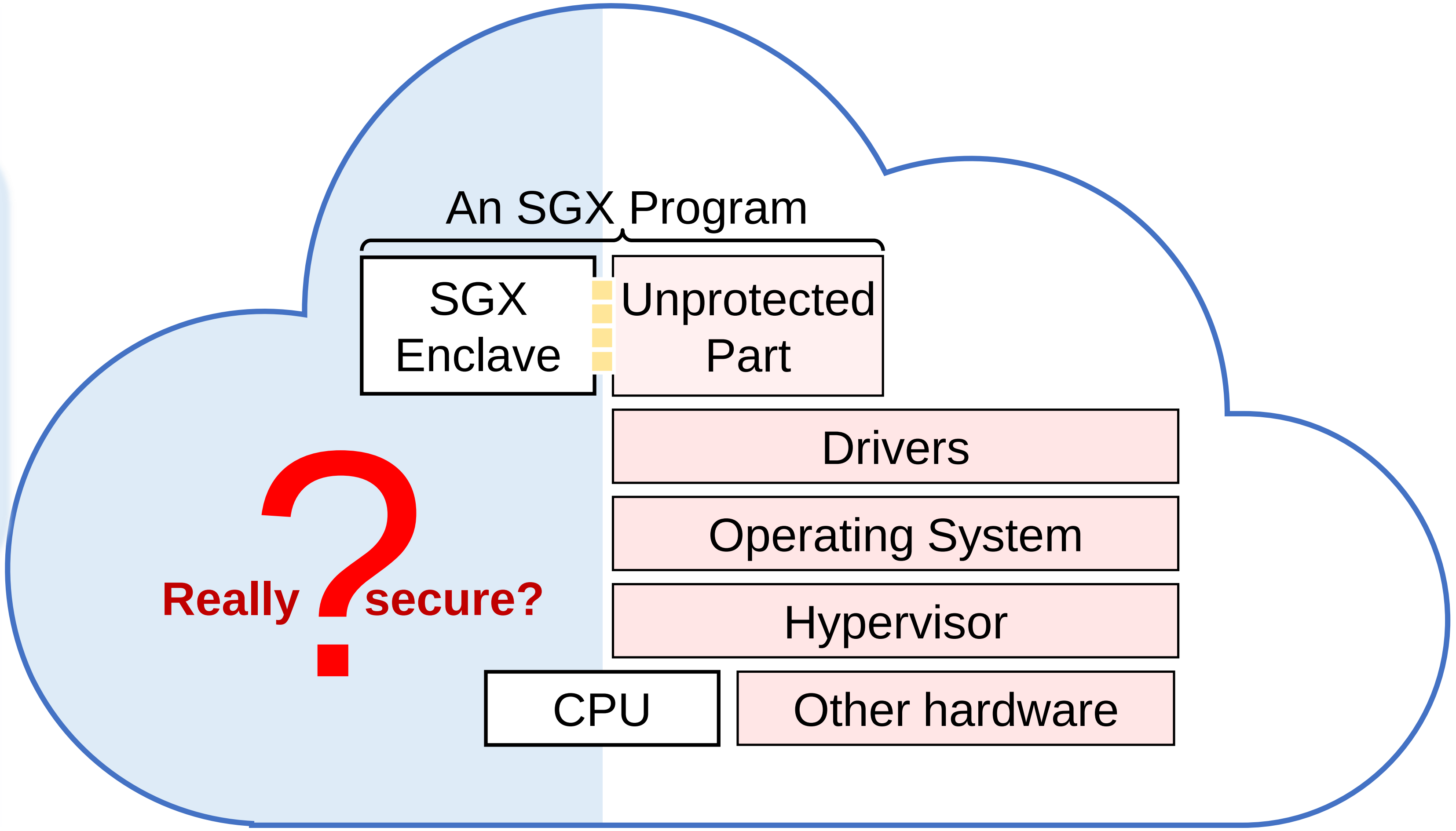


- ➢ Control flow path
- ➢ Register values
- ➢ Memory contents

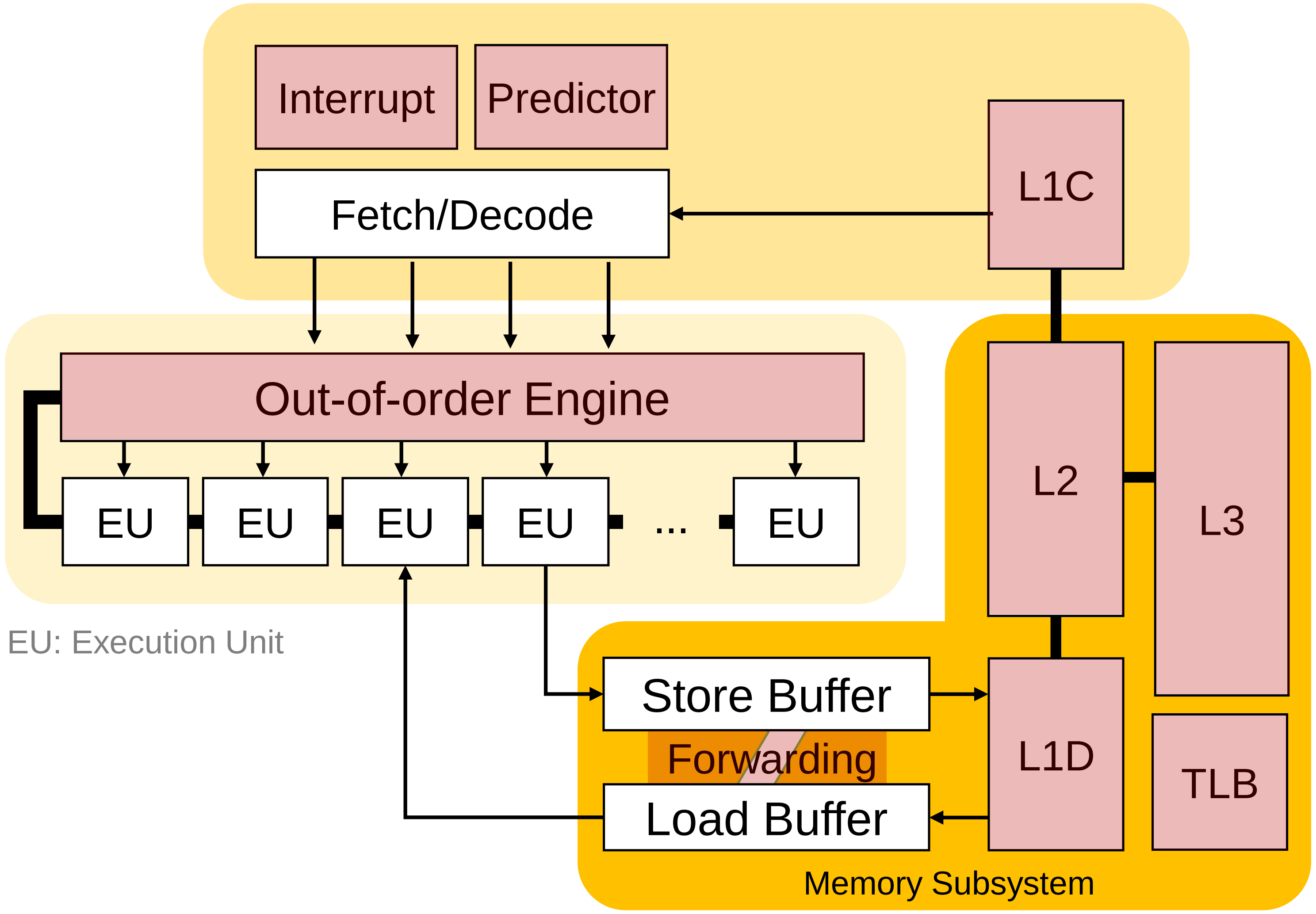
Cloud Security with SGX

SGX:
Software Guard Extensions

- ❖ Hardware-enforced security
- ❖ Isolated execution
- ❖ Execution state invisible
  - ➢ control path, registers
- ❖ Data encrypted



Modern Microarchitecture



EU: Execution Unit

Side-channels against SGX

Cache	“CacheZoom: How SGX Amplifies the Power of Cache Attacks,” CHES, 2017 “Cache Attacks on Intel SGX,” EuroSec, 2017
TLB	“Controlled-Channel Attacks: Deterministic Side Channels for Untrusted Operating Systems,” IEEE S&P, 2015 “Telling Your Secrets Without Page Faults: Stealthy Page Table-based Attacks on Enclaved Execution,'' USENIX Security, 2017 “Leaky Cauldron on the Dark Land: Understanding Memory Side-Channel Hazards in SGX,'' ACM CCS, 2017
Predictor	“BranchScope: A New Side-Channel Attack on Directional Branch Predictor,'' ASPLOS, 2018 “Inferring Fine-grained Control Flow inside SGX Enclaves with Branch Shadowing,'' USENIX Security, 2017
OoO Flaw	“SgxPectre Attacks: Leaking Enclave Secrets via Speculative Execution,'' arXiv:1802.09085, 2018
Forwarding	“MemJam: A False Dependency Attack Against Constant-Time Crypto Implementations in SGX,'' CT-RSA, 2018
Interrupt logic	SGXlinger (This work)

Background

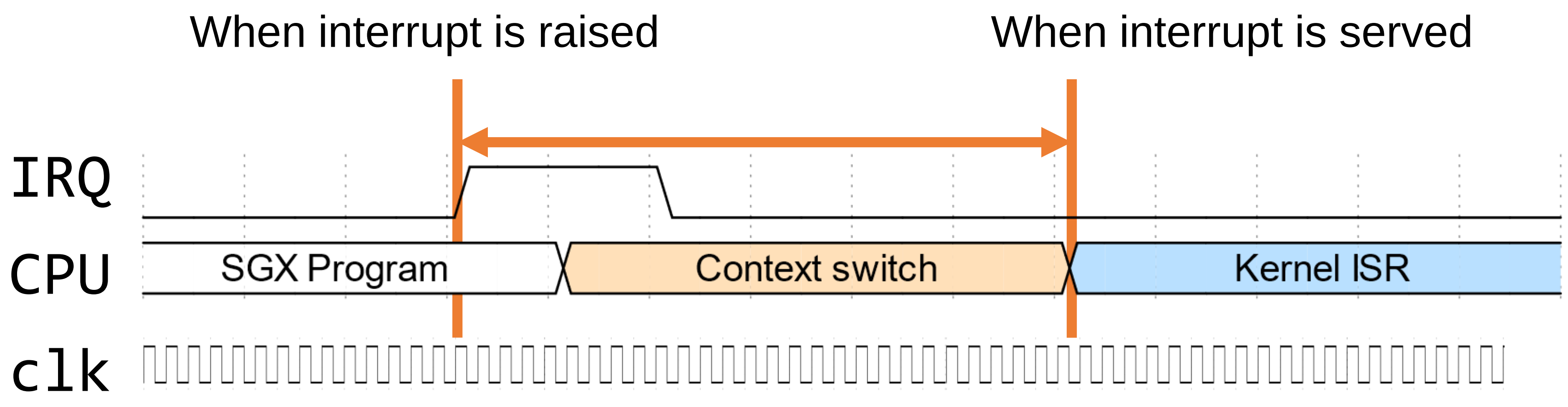
SGXlinger

Experiments

Interrupt

- Common event in the system
- OS responds to an external request

Interrupt Latency



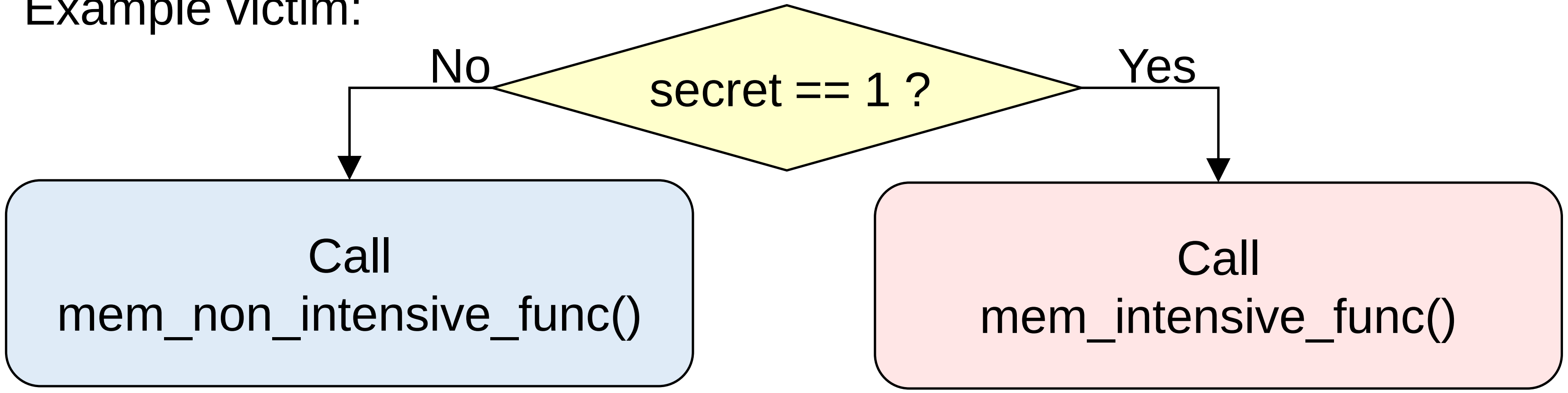
ISR: Interrupt service routine

Interrupt Latency as a Side-channel

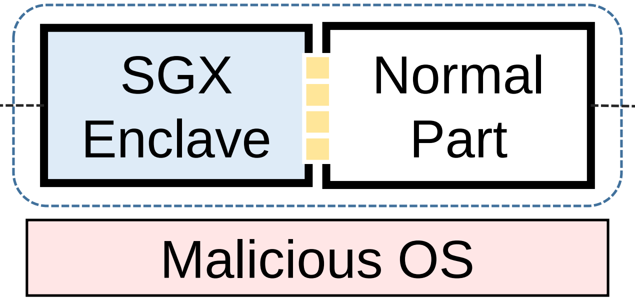
Interrupt latency $\propto$ Memory-intensiveness

CPU lingers in SGX mode ↑ Memory-intensiveness ↑

Example victim:

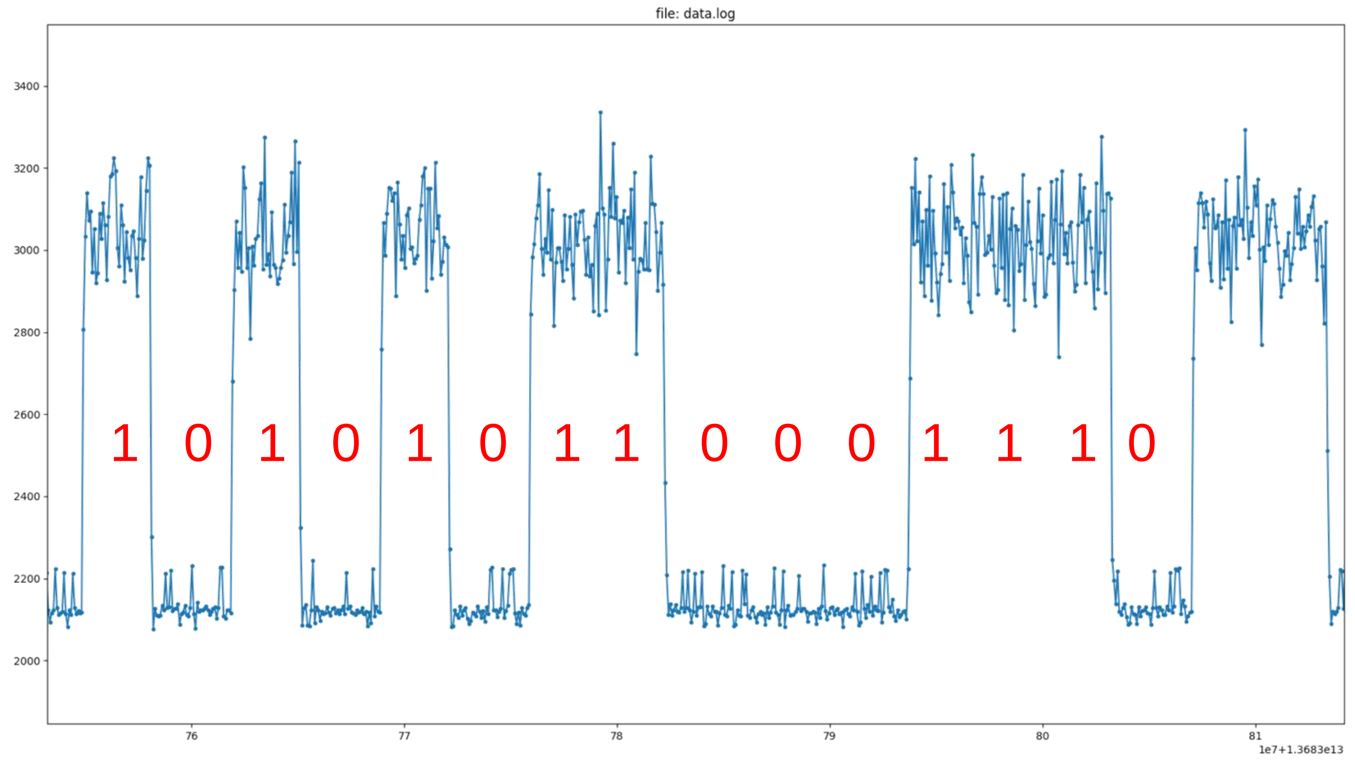


SGXlinger Demo

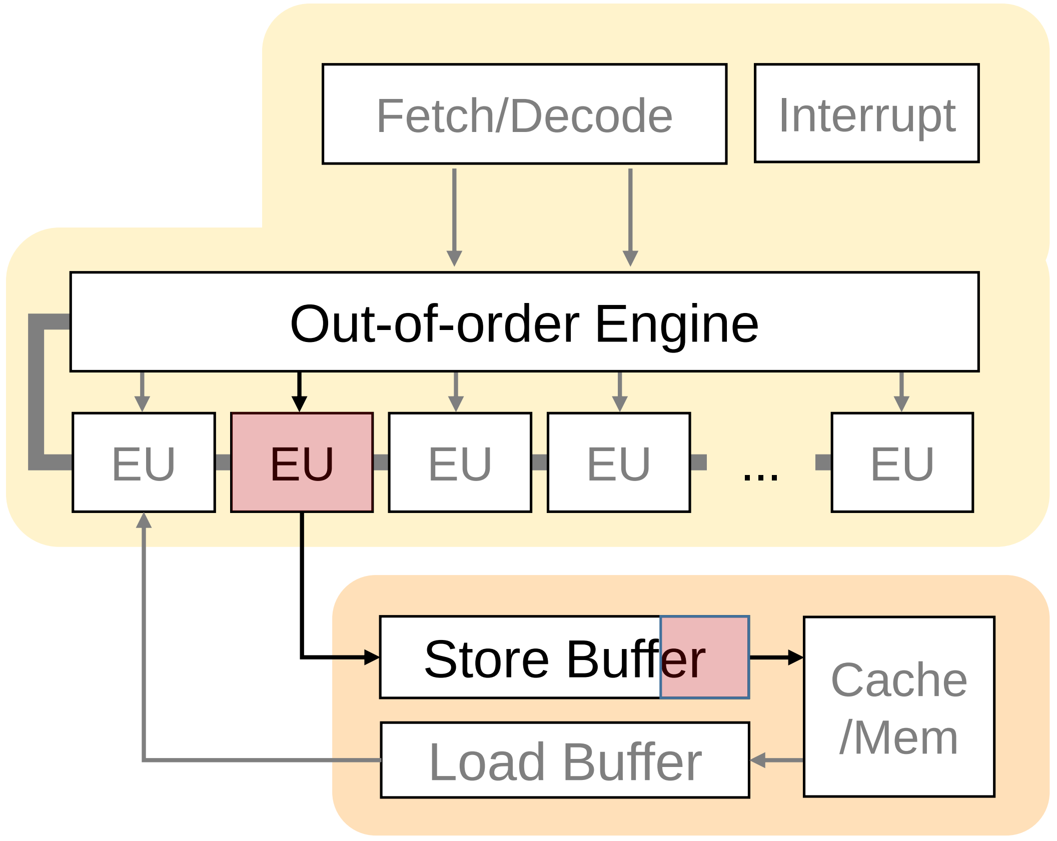


```
int main(){
    ...
    recv(data);
    res = enclave_process(data);
    ...
}
```

```
int enclave_process(char* data){
    secret = decrypt(data);
    for( bool& bit : secret )
        if(bit==1)
            mem_intensive_func();
        else
            mem_non_intensive_func();
    ...
}
```

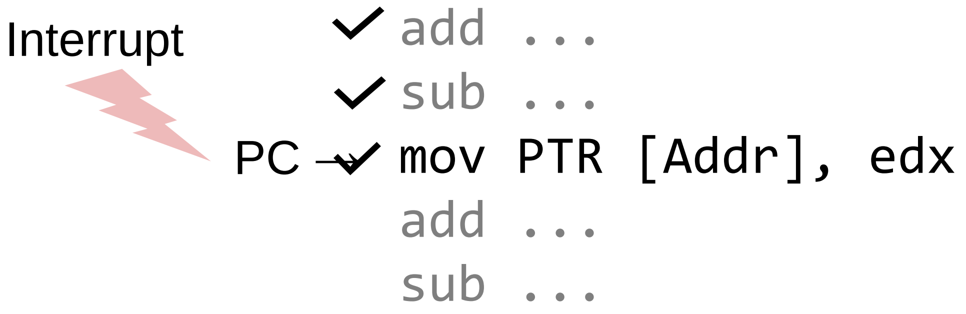


Reasons behind SGXlinger



Steps of **Store** instruction:

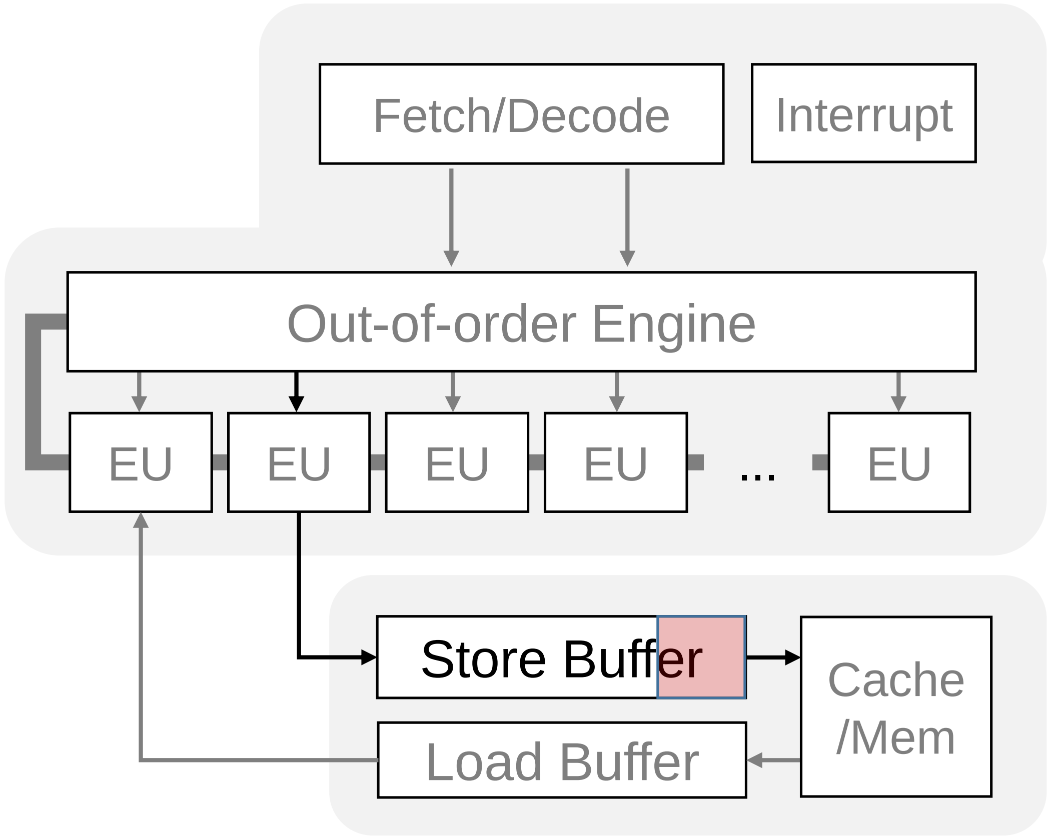
1) Prepare in EU, and register in store buffer;
2) Retire (commit) unless exception;
3) Store buffer starts to process the store.



```
✓ add ...
✓ sub ...
✓ mov PTR [Addr], edx
  add ...
  sub ...
```

[1] Intel, “Intel Architectures Optimization Reference Manual,” April 2018. Reference no. 248966-032.
[2] Intel, “Intel Architectures Software Developer’s Manual,” May 2018. Reference no. 325462-067US.

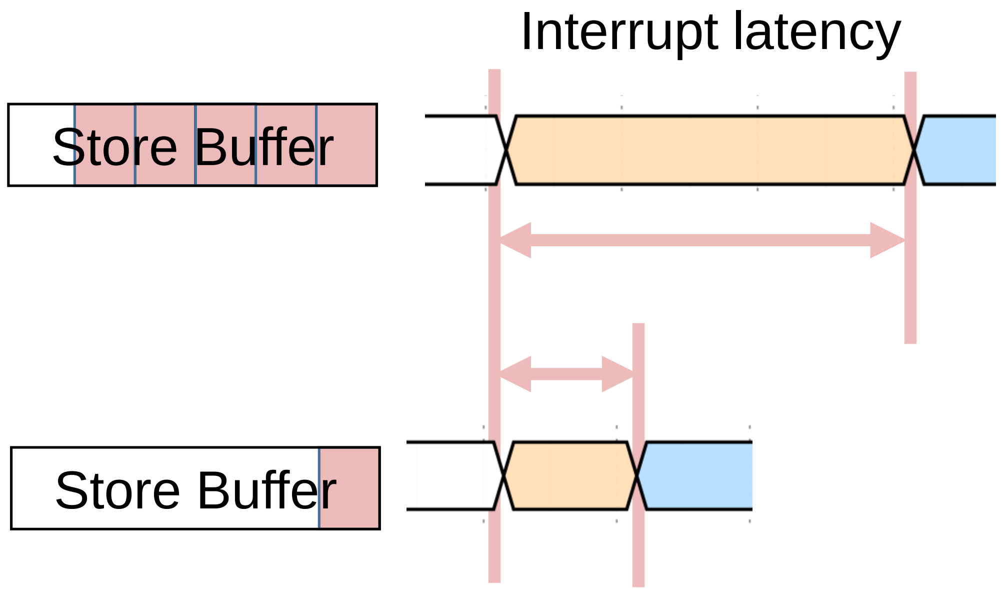
Reasons behind SGXlinger



Content of store buffer is always drained to memory on interrupt.

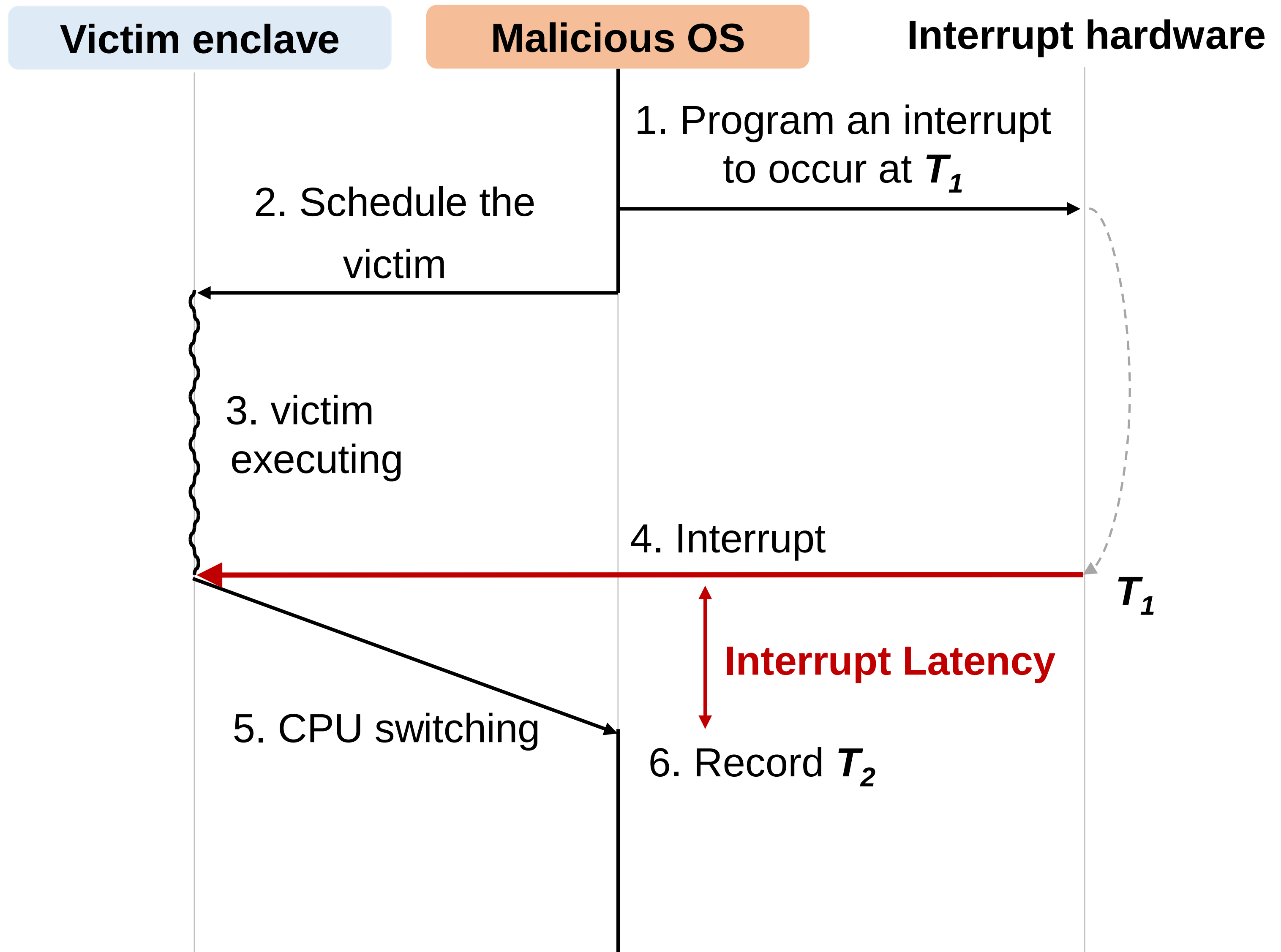
```
✓ mov ...
✓ mov ...
✓ mov ...
✓ mov ...
→ sub ...
```

```
✓ add ...
✓ mul ...
✓ add ...
✓ sub ...
→ mov ...
```

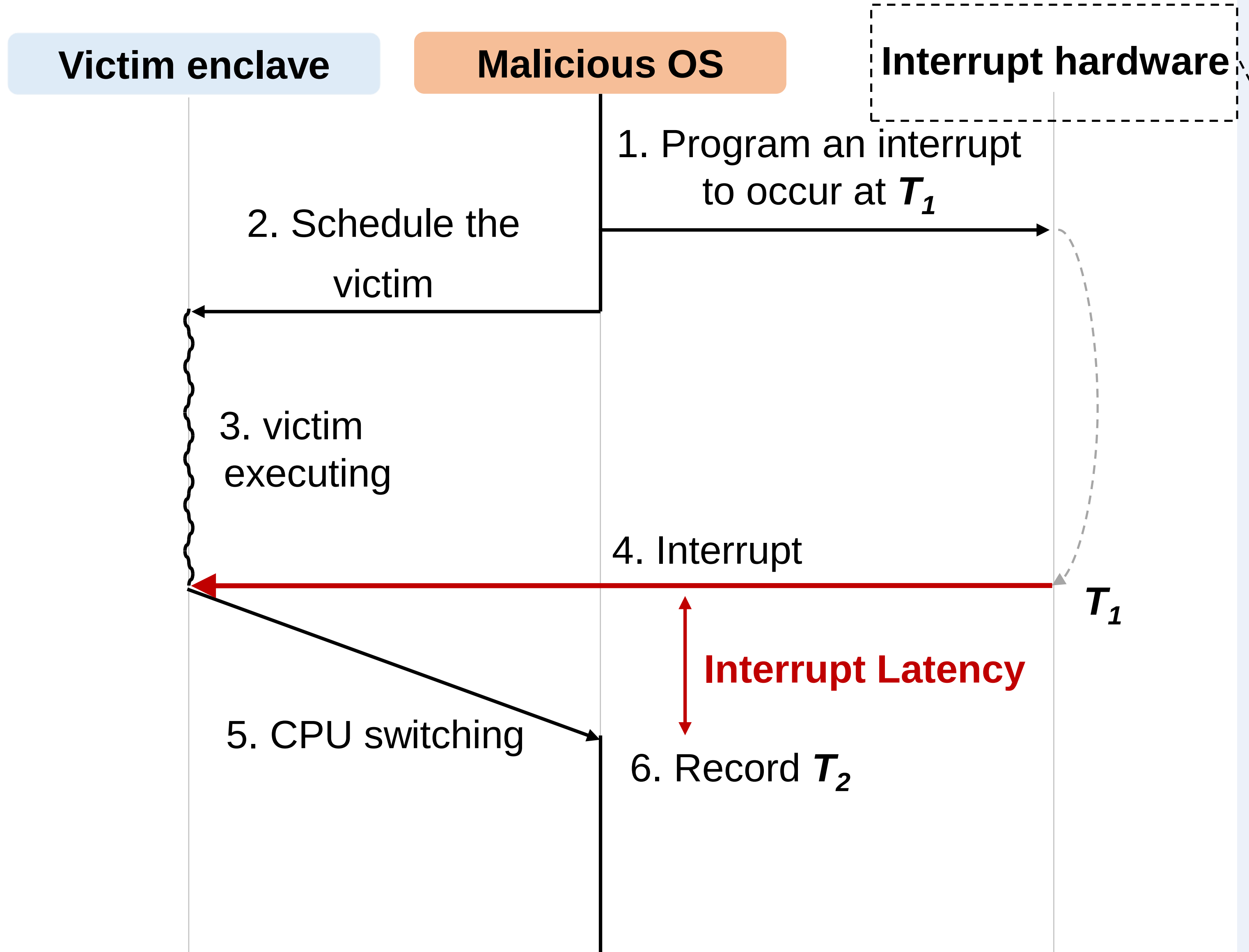


[1] Intel, "Intel Architectures Optimization Reference Manual," April 2018. Reference no. 248966-032.
[2] Intel, "Intel Architectures Software Developer's Manual," May 2018. Reference no. 325462-067US.

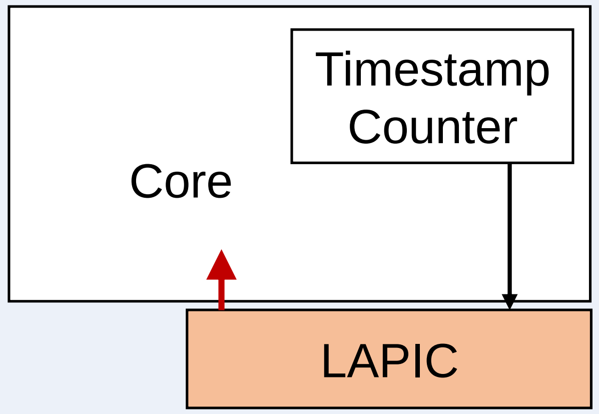
Measurement of Interrupt Latency



Optimize Accuracy



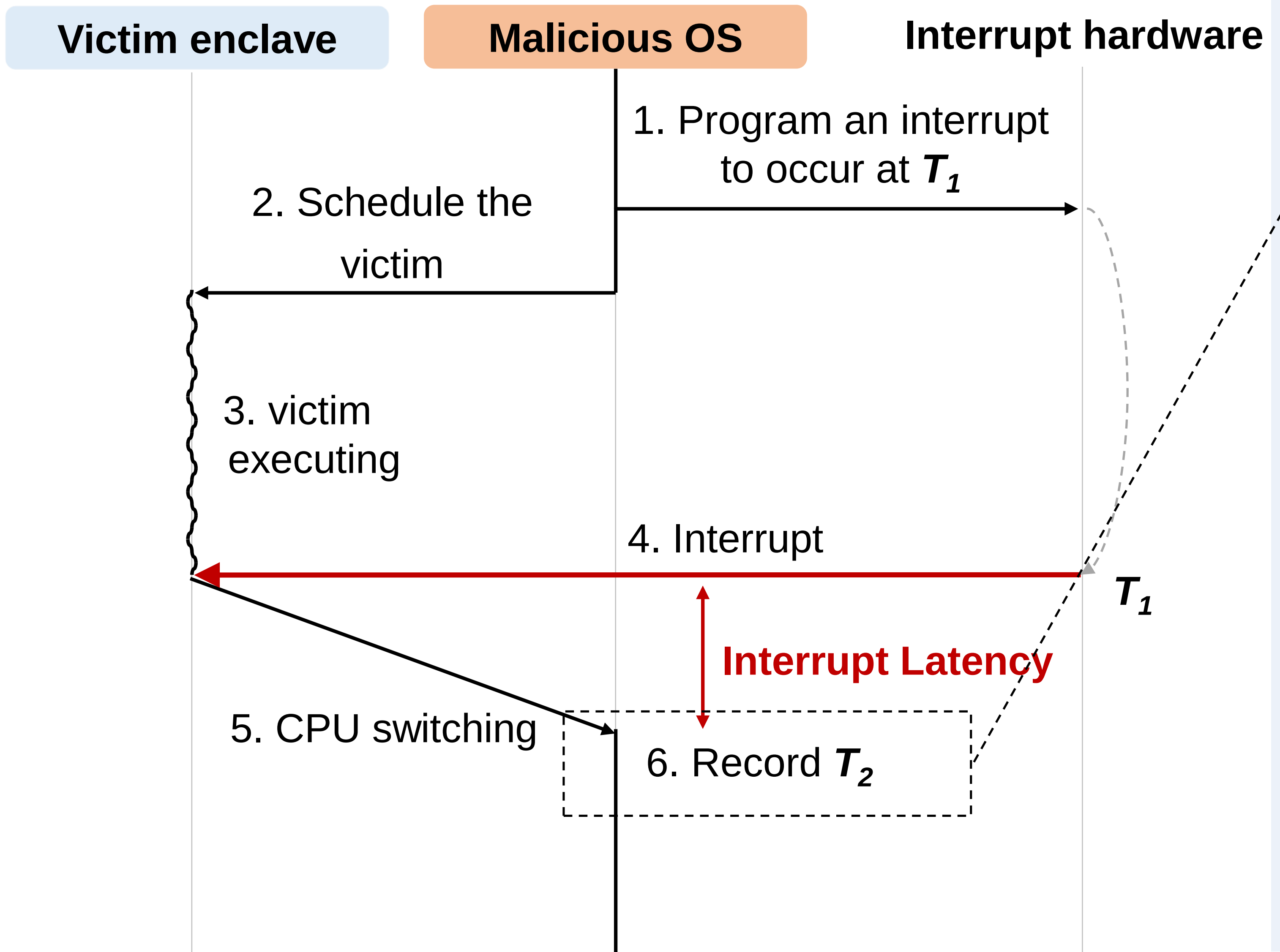
(1) Interrupt source selection



LAPIC: Local Adv. Programmable Interrupt Controller

- Programmable interrupt
- Accurate timestamp counter
- Near the processor core

Optimize Accuracy



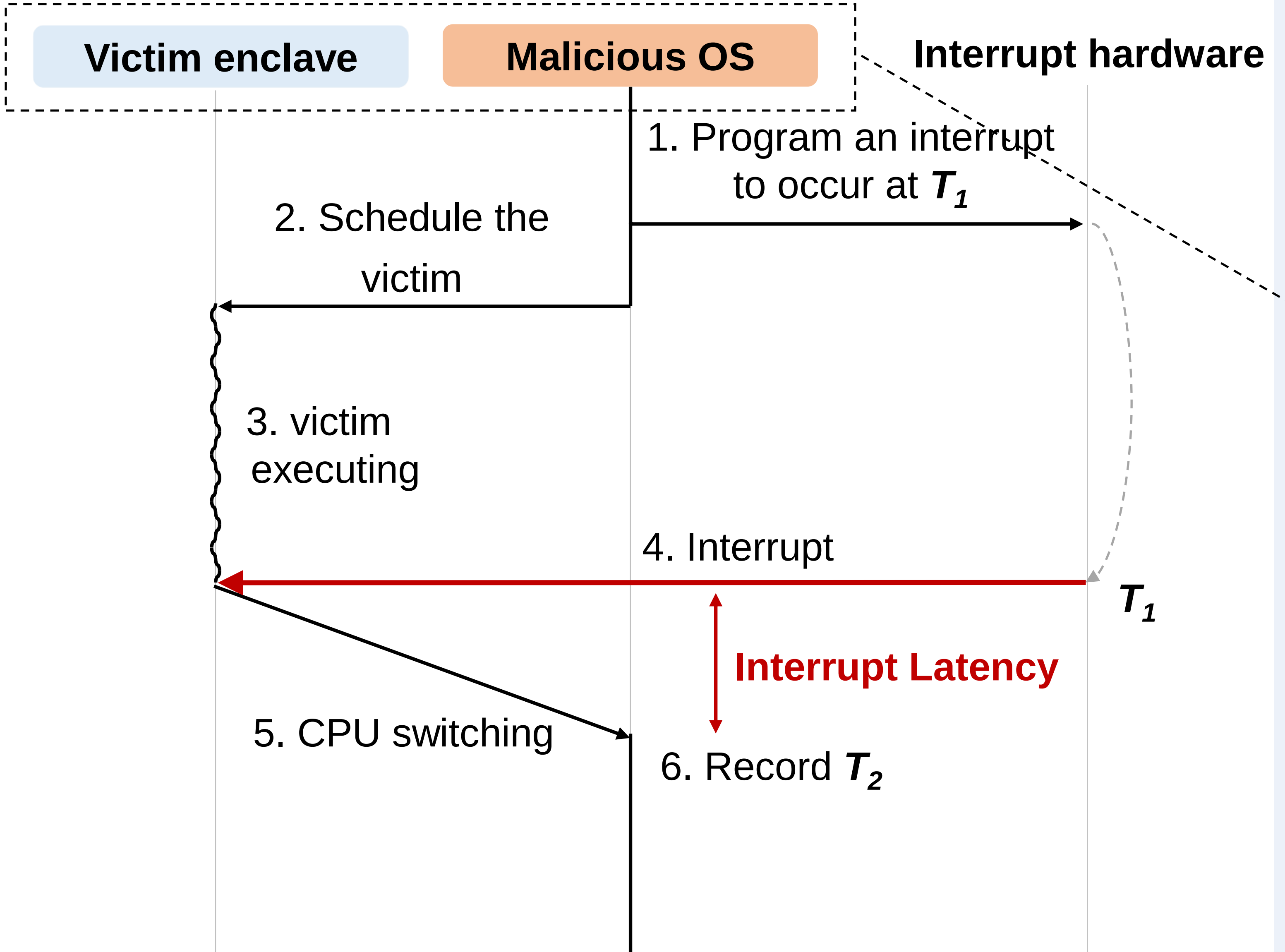
(1) Interrupt source selection

(2) Kernel instrumentation

- ➢ Based on Ubuntu 16.04 LTS

```
File: arch/x86/entry/entry_64.S:
  ENTRY(apic_timer_interrupt)
      mov   PTR [rsp-6*8], rax
      mov   PTR [rsp-4*8], rdx
      rdtsc
      shl   rdx, 32
      or    rax, rdx
      ...
```

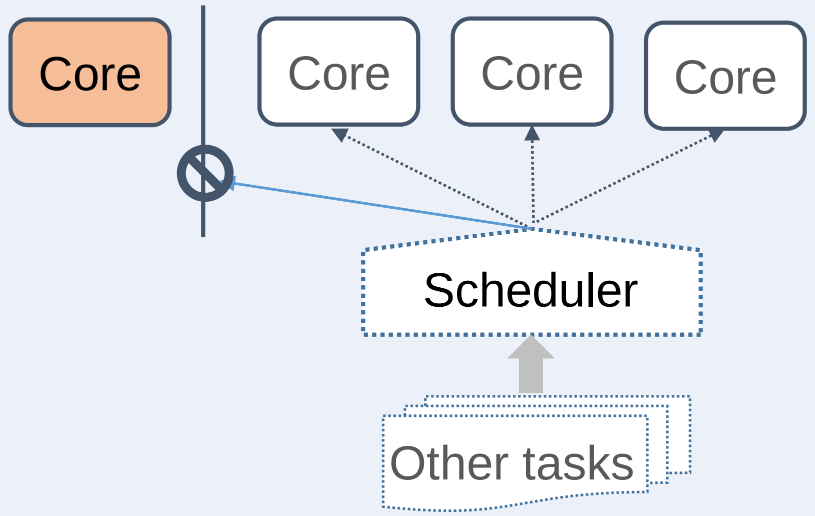
Optimize Accuracy



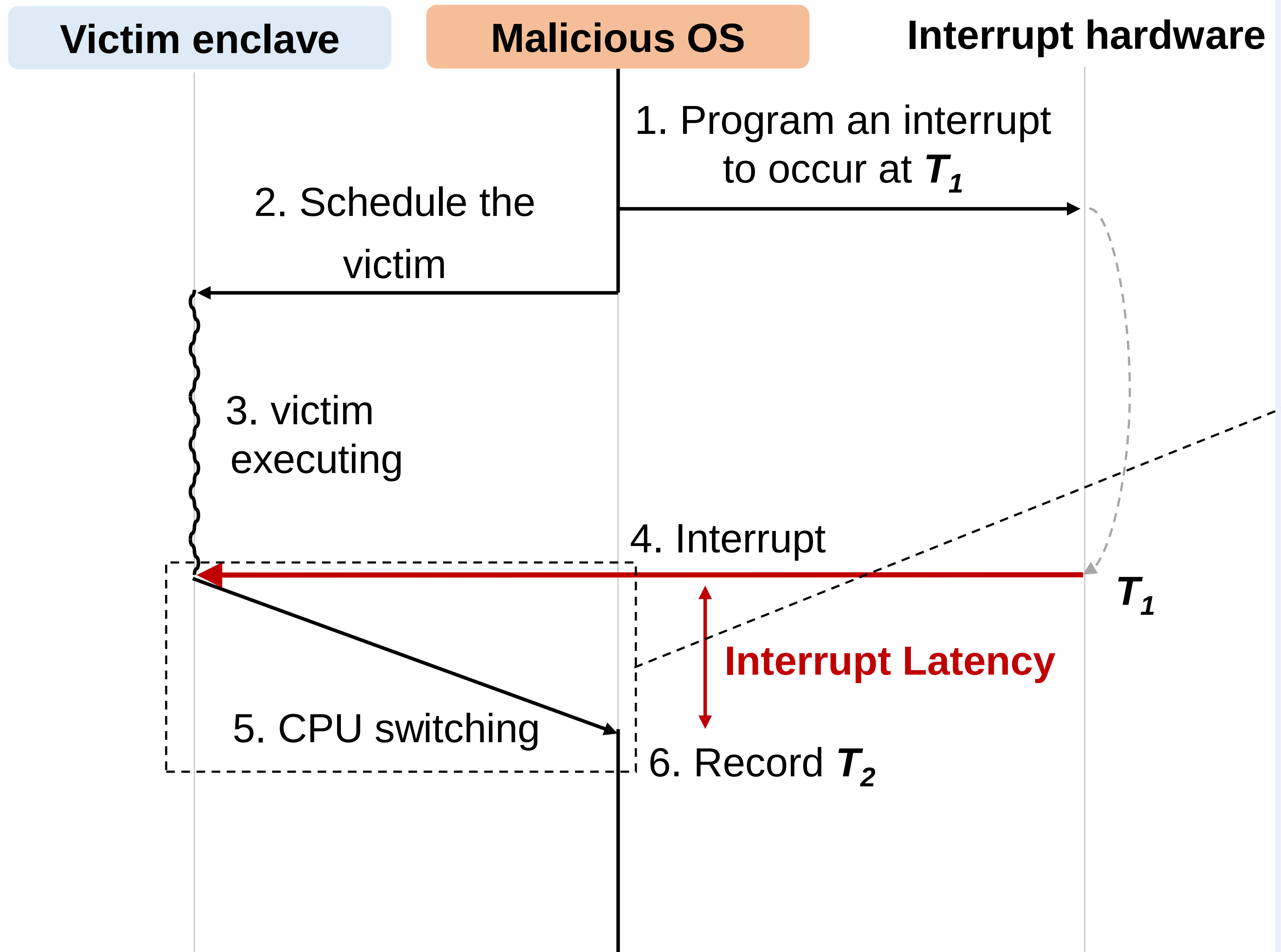
(1) Interrupt source selection

(2) Kernel instrumentation

(3) Isolating a core



Optimize Accuracy



(1) Interrupt source selection

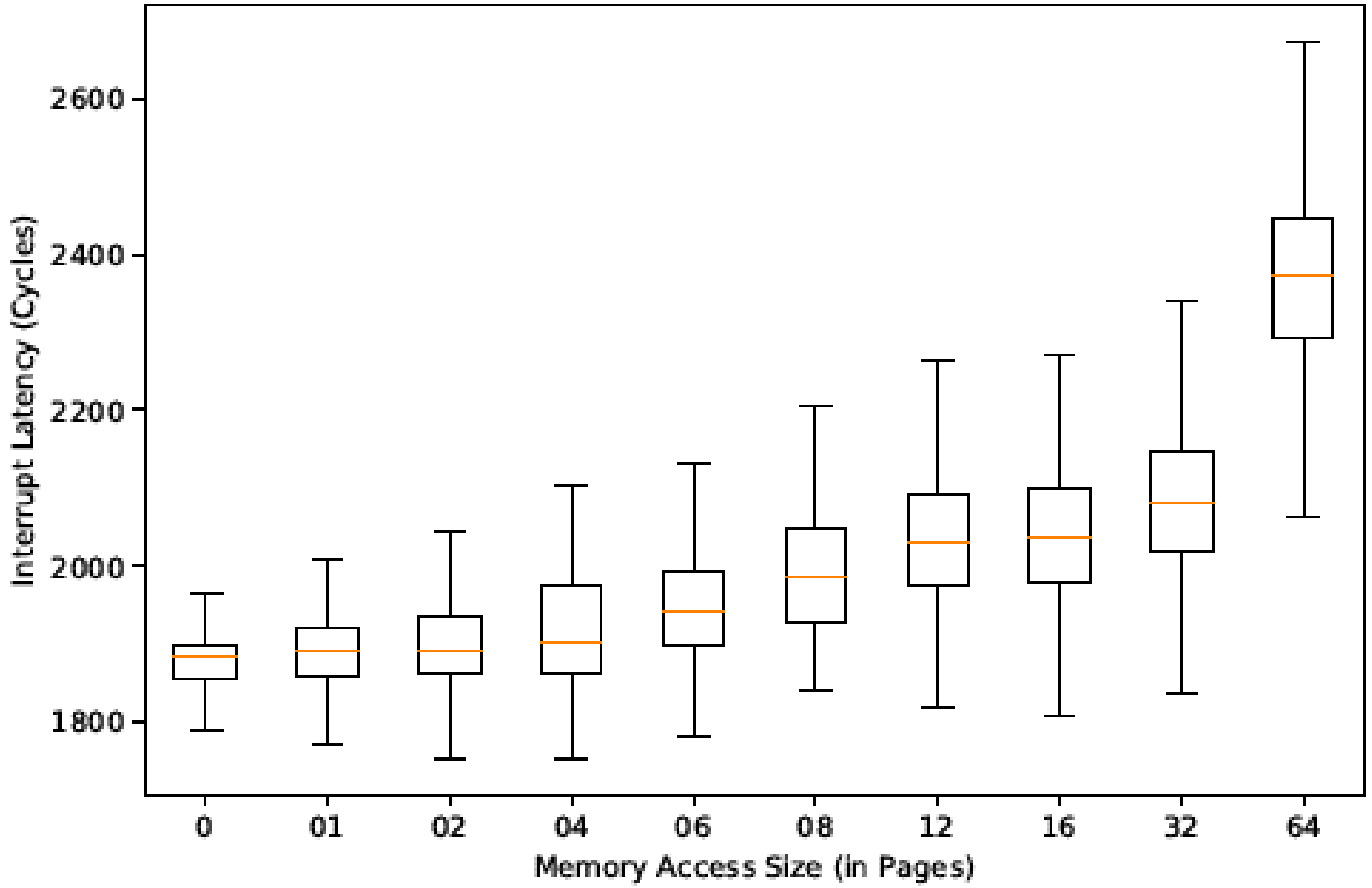
(2) Kernel instrumentation

(3) Isolating a core

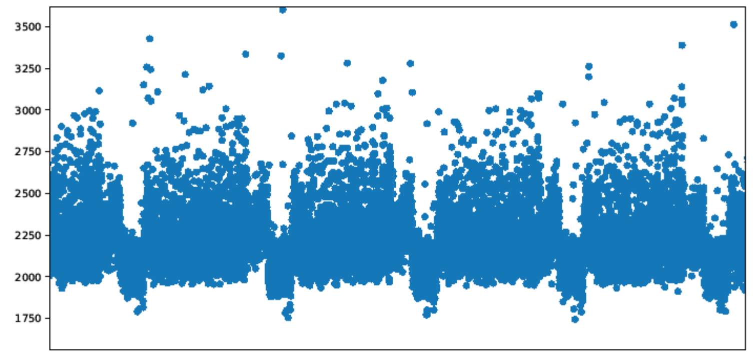
(4) Stabilizing clock speed

- Disable TurboBoost
- Fix processor frequency through Intel P-state driver

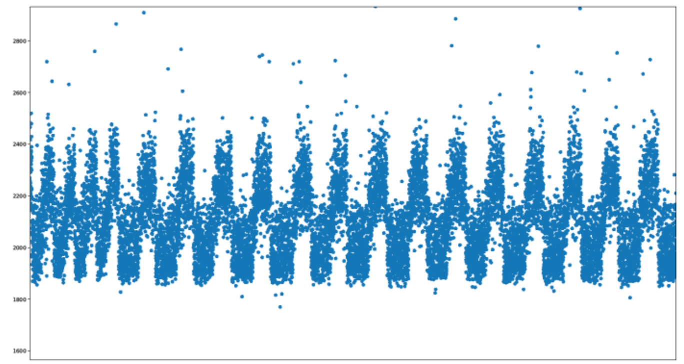
Interrupt Latency vs. Mem Footprint



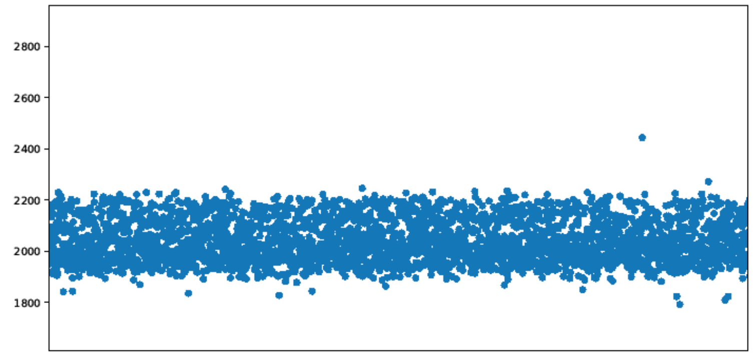
Differentiate Programs



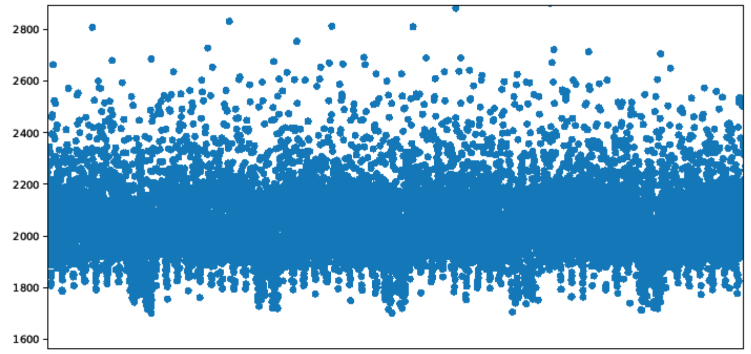
bzip - compression



bzip - decompression



soplex (comptation-intensive)



h264

Interrupt Latency Side-channel

To be independently reported by J.V. Bulck *et. al.* in CCS'18:

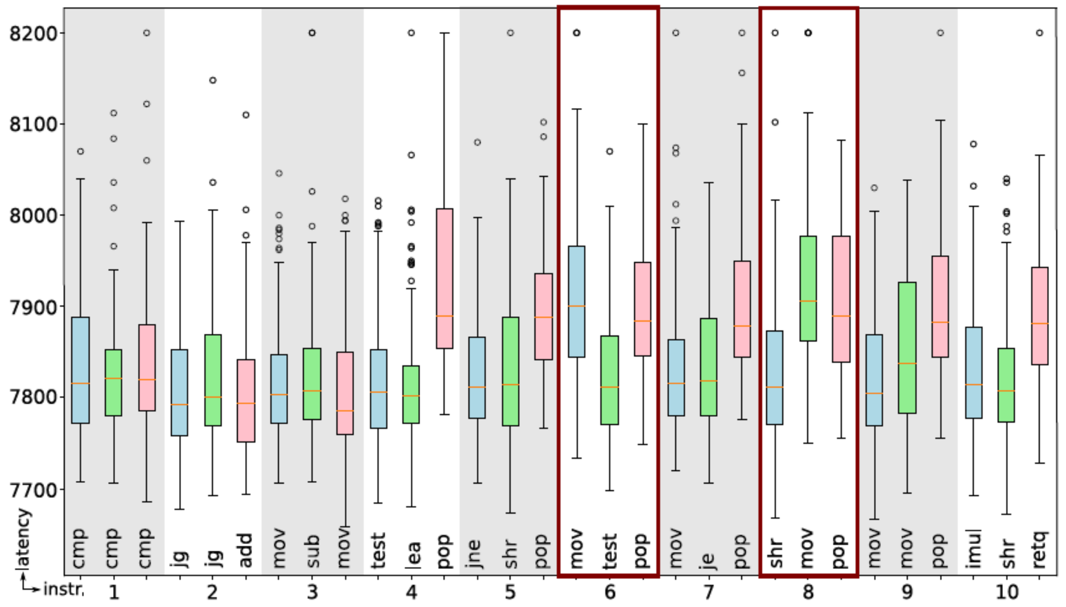


Figure 8: IRQ latency distributions for 100 runs of bsearch left (blue) vs. right (green) vs. equal (red) execution paths.

CCS'18 [1]

	CCS'18 [1]	**This work**
Mechanism	Pipeline delay	Buffer drain
CPU speed	Slow	Normal
Interrupt occurrences	High (Per Instruction)	Medium
Noise	High	High
Granularity	Instruction level	Coarse

$\downarrow$ More leakage

$\downarrow$ Stealthier

[1] Jo Van Bulck, Frank Piessen and Raoul Strackx, "Nemesis: Studying Microarchitectural Timing Leaks in Rudimentary CPU Interrupt Logic," ACM CCS, Oct. 15-19, 2018

Conclusion

- **Interrupt Latency**
  - A new attack vector
- **The SGXlinger Attack**
  - Break SGX security
  - Open-sourced
- **Limitations**:
  - Coarse-grained
  - Noisy Channel

http://git.io/sgxlinger

SGXlinger Tools