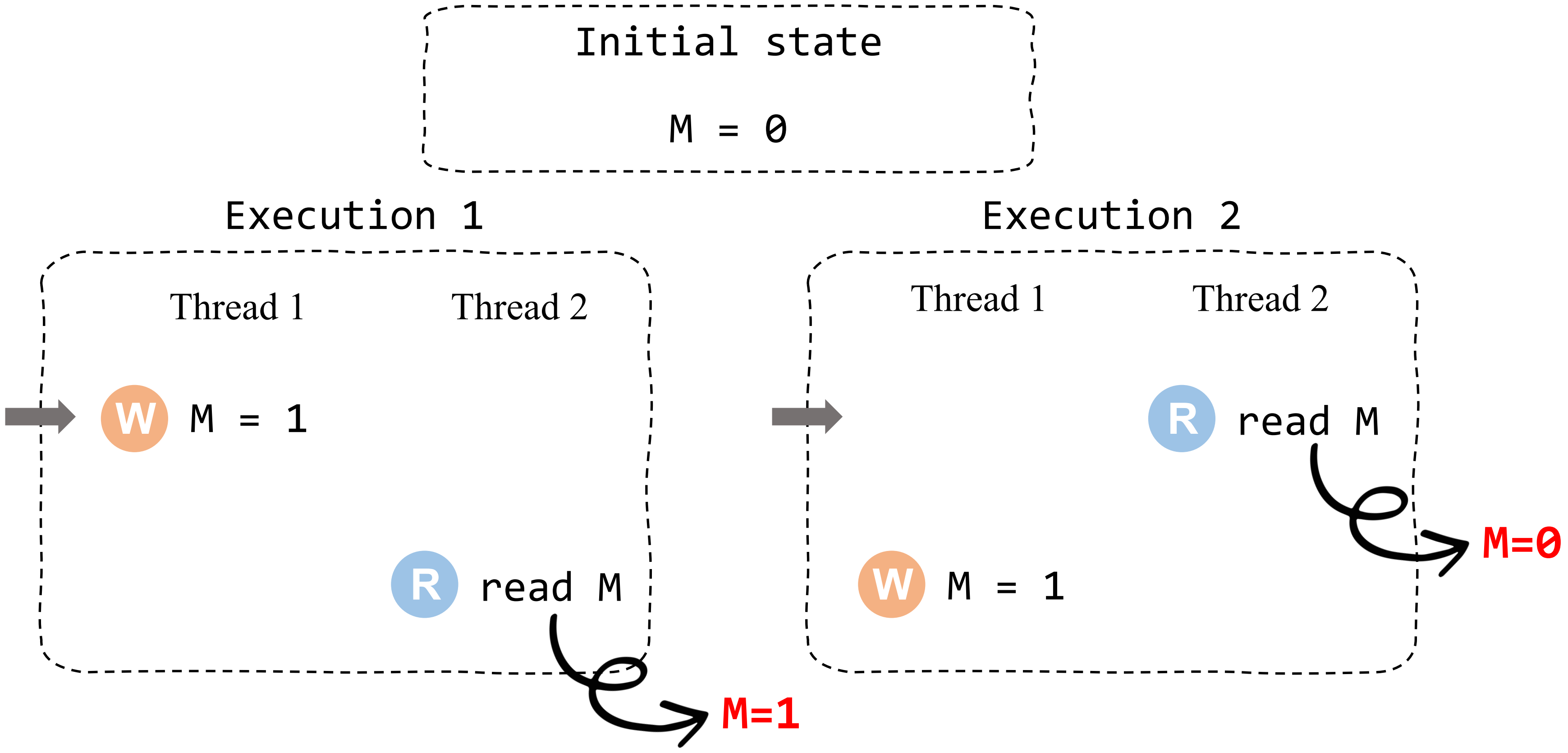
EXPRACE: Exploiting Kernel Races through Raising Interrupts

Yoochan Lee[†], Changwoo Min[‡], Byoungyoung Lee[†]

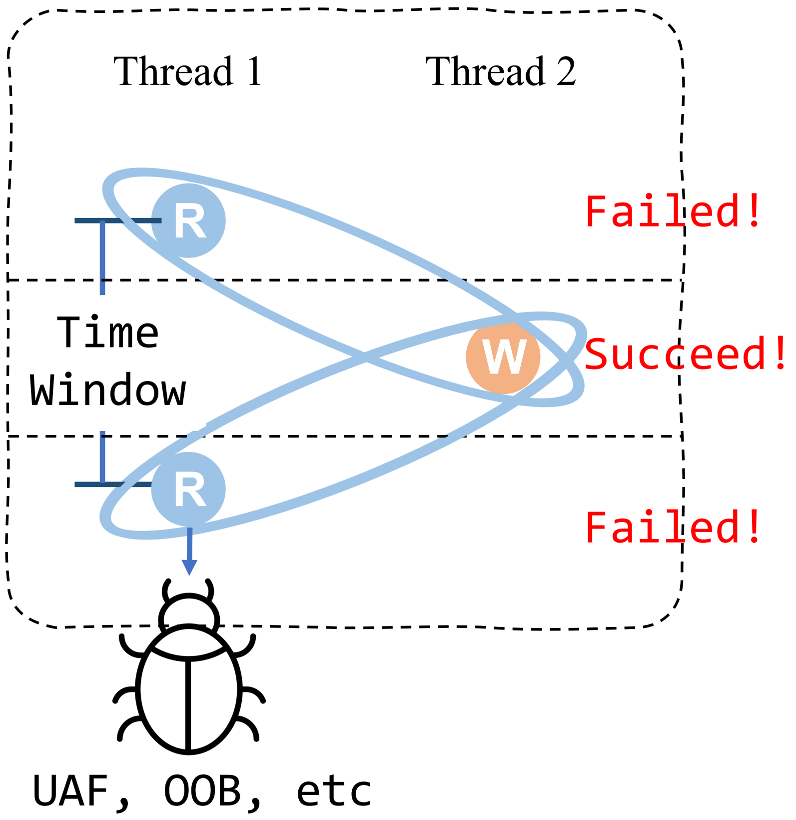
Seoul National University[†], Virginia Tech[‡]

Race Condition



Different execution order
➔ Different execution results!

Race Condition Bug



- Race bug consist of two or more race pairs
  - Must be executed in a **specific order**
- Difficult to succeed
  - Two threads must be executed in order
- Bruteforce somehow works still

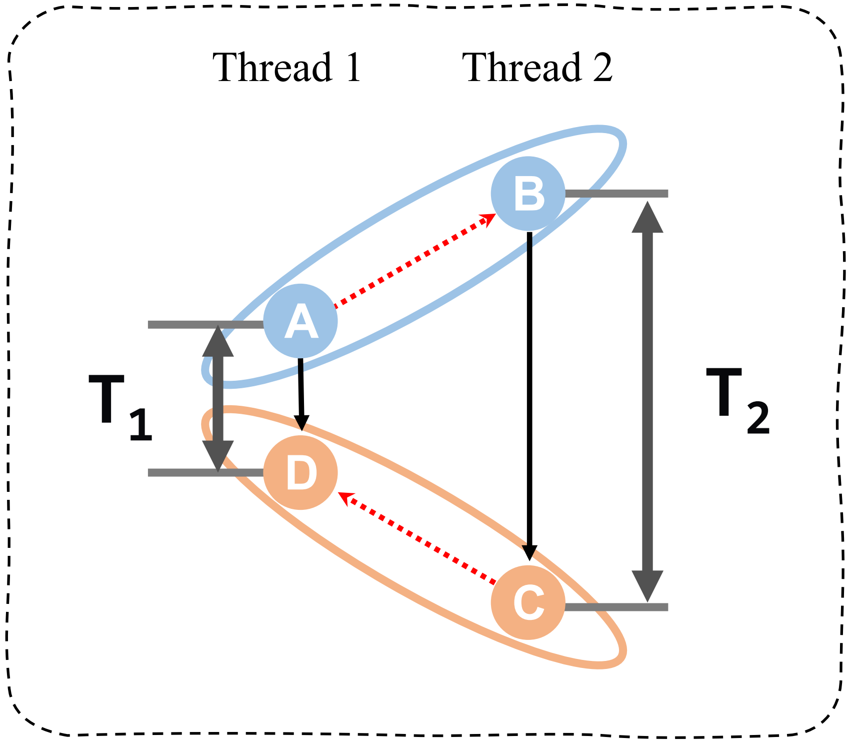
Problem : Multi-Variable Race

RACE CVE	Bruteforce work?
CVE-2019-11486	✔
CVE-2017-7533	✔
CVE-2017-2636	✔
CVE-2016-8655	✔
CVE-2019-6974	✗
CVE-2019-2025	✗
CVE-2019-1999	✗
CVE-2017-15265	✗

- We found several races are exploited with bruteforce

- However, some races cannot be exploited with brutefoce

Problem : Multi-Variable Race



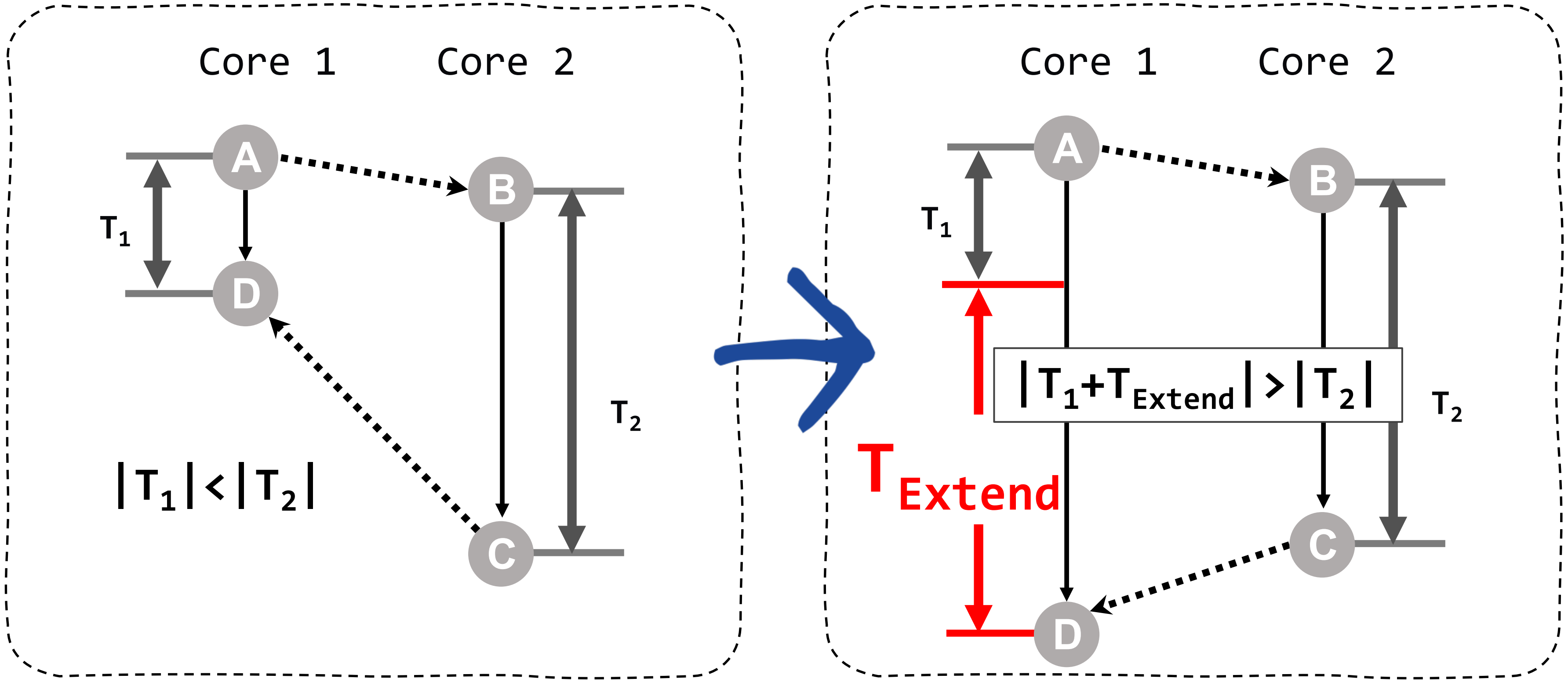
Access to memory M_1

Access to memory M_2

CVE	T_1		T_2
CVE-2019-6974	18		1210
CVE-2019-2025	50	<	600
CVE-2019-1999	150		1800
CVE-2017-15265	35		450

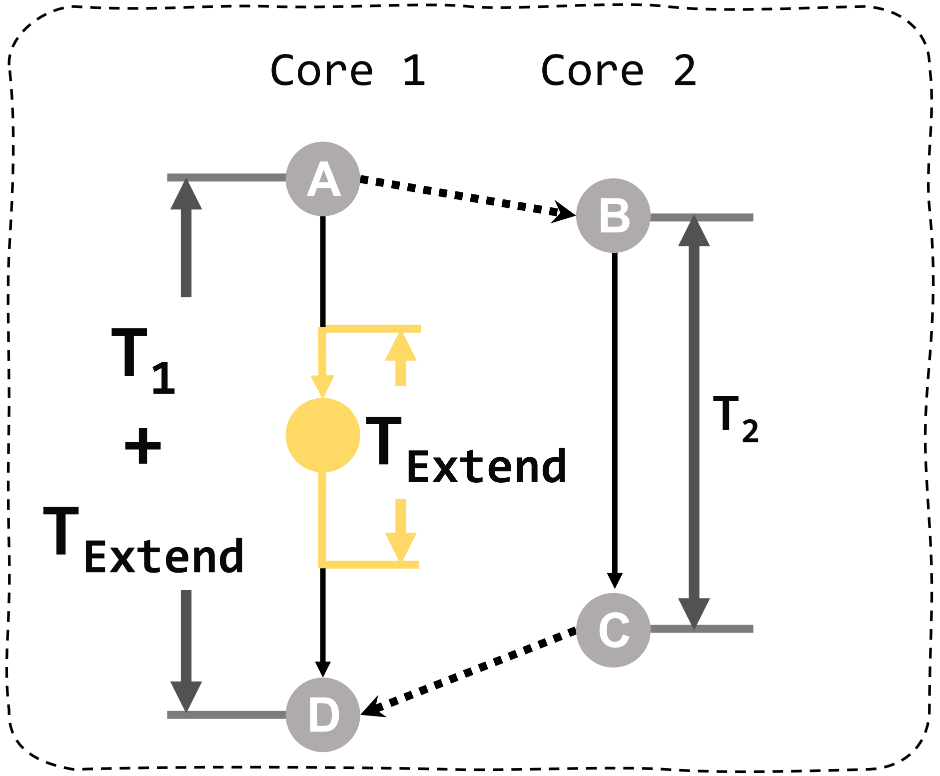
- Some races are (practically) impossible to exploit
  - If $T_1 < T_2$

Goal: How do we exploit Multi-variable Race?



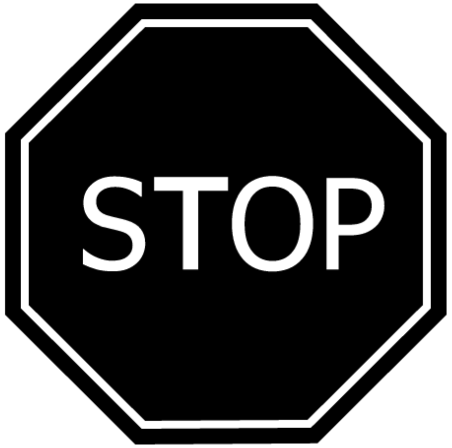
Idea: Extend the time window to make $|T_1 + T_{Extend}|$ is larger than T_2

Idea: Extending the time window?

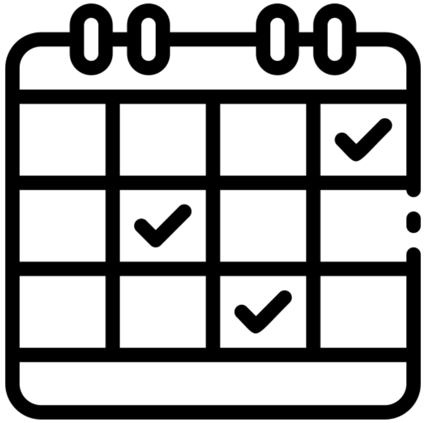


Preempting the thread execution

Typical preemption methods



Kernel breakpoint



Kernel thread schedule

interrupt

Challenge:
Preemption is not under user's control

Kernel breakpoint

Kernel thread schedule

interrupt

Back to OS basics:

Users cannot control preemption methods

User-controlled Preemption through Interrupts

interrupt

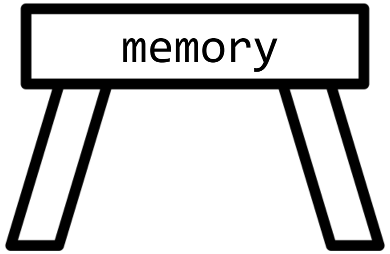
Fact: Users **CANNOT** raise interrupts **directly**

We found that users **CAN** raise interrupts **indirectly**

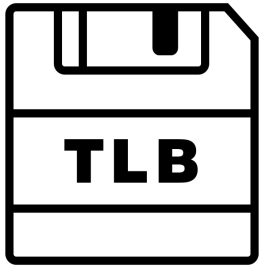
User-controlled Preemption through Interrupts

We found **four methods to indirectly raise interrupts**

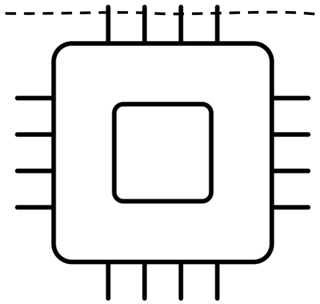
- **Reschedule** interrupt



- **Membarrier** interrupt



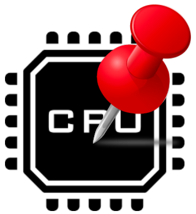
- **TLB shootdown** interrupt



- **Hardware** interrupt

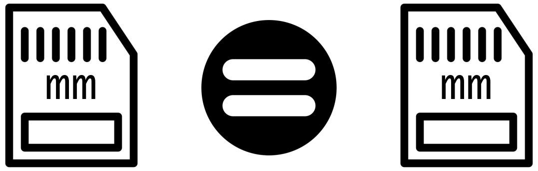
Requirement: Precise Interrupt Control

`We should send an interrupt` **`to a desired CPU core`**`!`



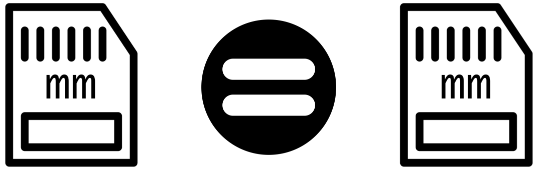
CPU pinning

- **Reschedule** interrupt



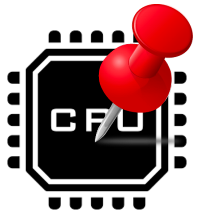
Same memory map

- **Membarrier** interrupt



Same memory map

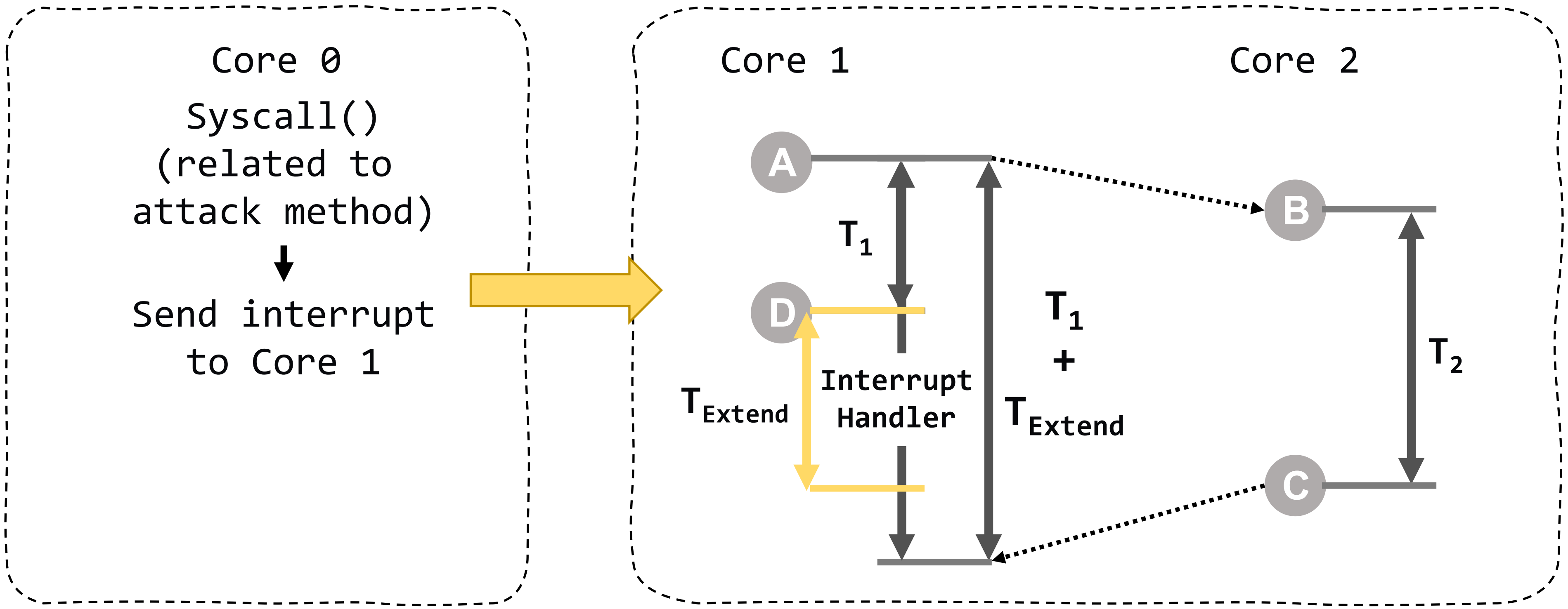
- **TLB shootdown** interrupt



CPU pinning

- **Hardware** interrupt

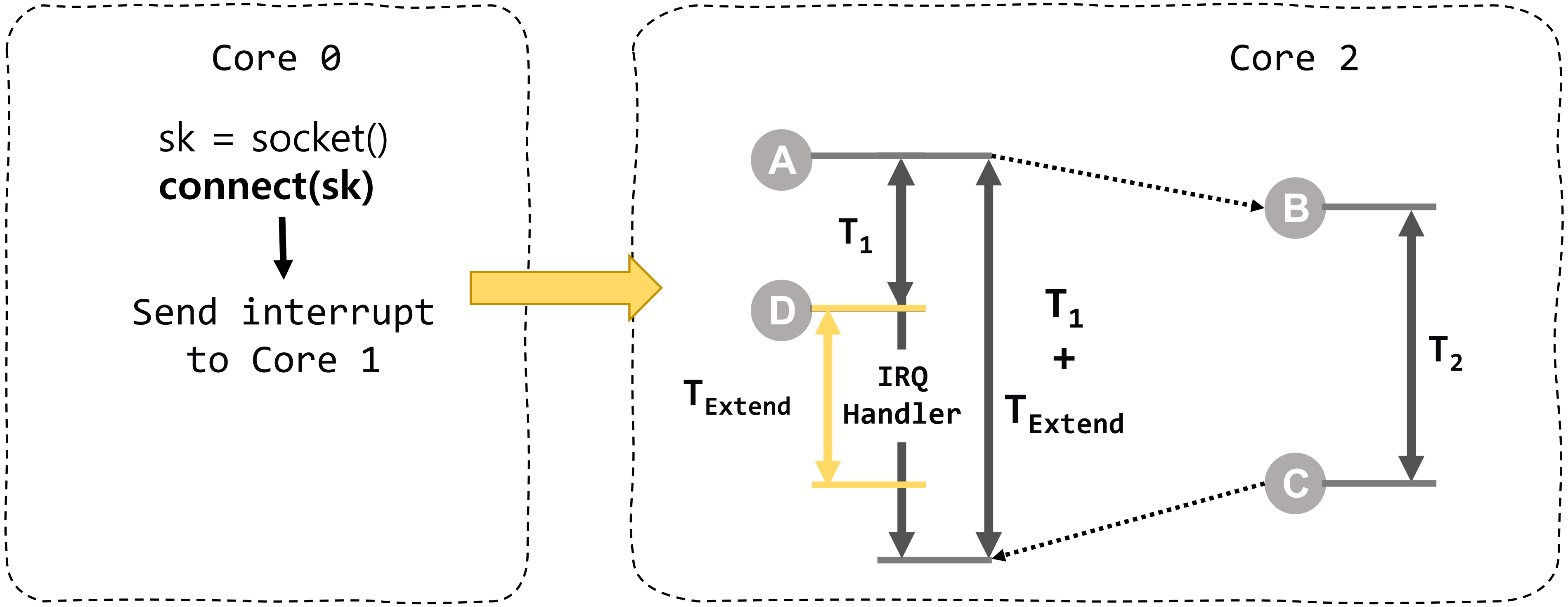
Our Approach : Interrupt



- The key insight behind EXPRACE is in intentionally enlarging race window using interrupt.

Example : Hardware Interrupt

```
yoochan@compsec:~$ cat  /proc/irq/121/smp_affinity_list
> Core 7
```



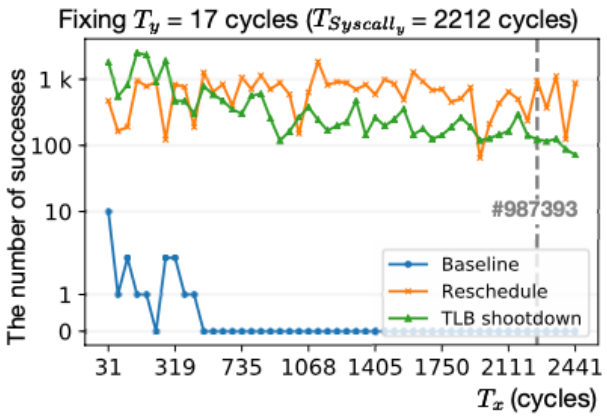
Real-World Races in Linux

Vulnerability	Baseline	Reschedule	membarrier	TLB shootdown	HW interrupt
CVE-2019-6974	✗	✗	✗	✗	✓
CVE-2019-2025	✗	✓	✓	✓	✓
CVE-2019-1999	✗	✗	✗	✓	✓
CVE-2017-15265	✗	✓	✓	✓	✓
11eb85ec…	✗	✗	✗	✓	✓
1a6084f8…	✗	✗	✗	✓	✓
20f2e4c2…	✗	✗	✗	✓	✓
484298f…	✗	✓	✓	✓	✓
da1b9564…	✗	✗	✗	✗	✓
e20a2e9c…	✗	✗	✗	✓	✓

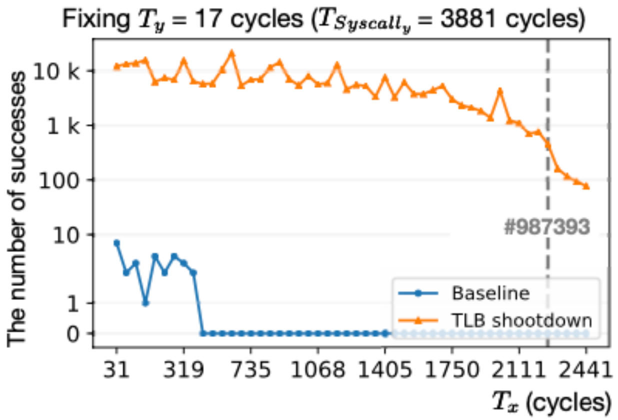
✗ denotes exploitation has failed for given 24 hours

All 10 cases are exploited with **EXPRACE**

Other OSes



Microsoft Windows



Mac OS X

- Reschedule and TLB shootdown has shown far more success numbers than baseline.

Conclusion

- We analyzed real-world kernel races and found an intrinsic condition separating easy-to-exploit and hard-to-exploit races.
- We developed EXPRACE, a generic race exploitation technique for Linux, Windows, OS X.
- EXPRACE demonstrated that it truly augments the exploitability of kernel races.