MicroScope: Enabling Microarchitectural Replay Attacks

Dimitrios Skarlatos, Mengjia Yan, Bhargava Gopireddy, Read Sprabery,
Josep Torrellas, and Christopher W. Fletcher

Presented by Mengjia Yan
MIT 6.888 Fall 2020

Why this paper?

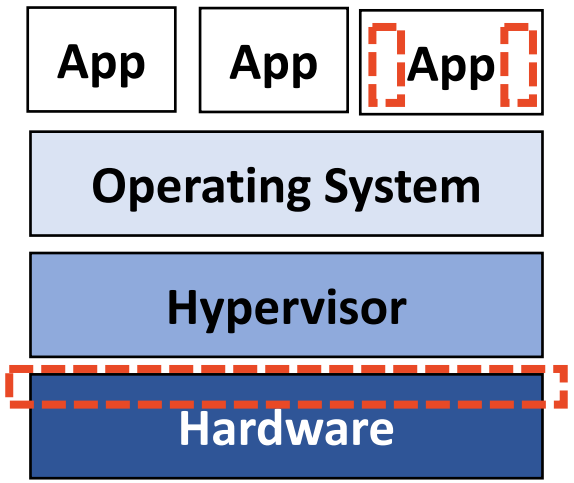
We have read a couple of attack papers, e.g., Spectre/Meltdown, Prime+Probe.

Why read this paper? What is new here at a high level?

Threat Model: Trusted Computing with SGX

- OS/Hypervisor are untrusted
  - OS/Hypervisor cannot introspect/tamper enclave
  - Unfortunately, OS/Hypervisor still manages demand paging

Attack Surface With Enclaves

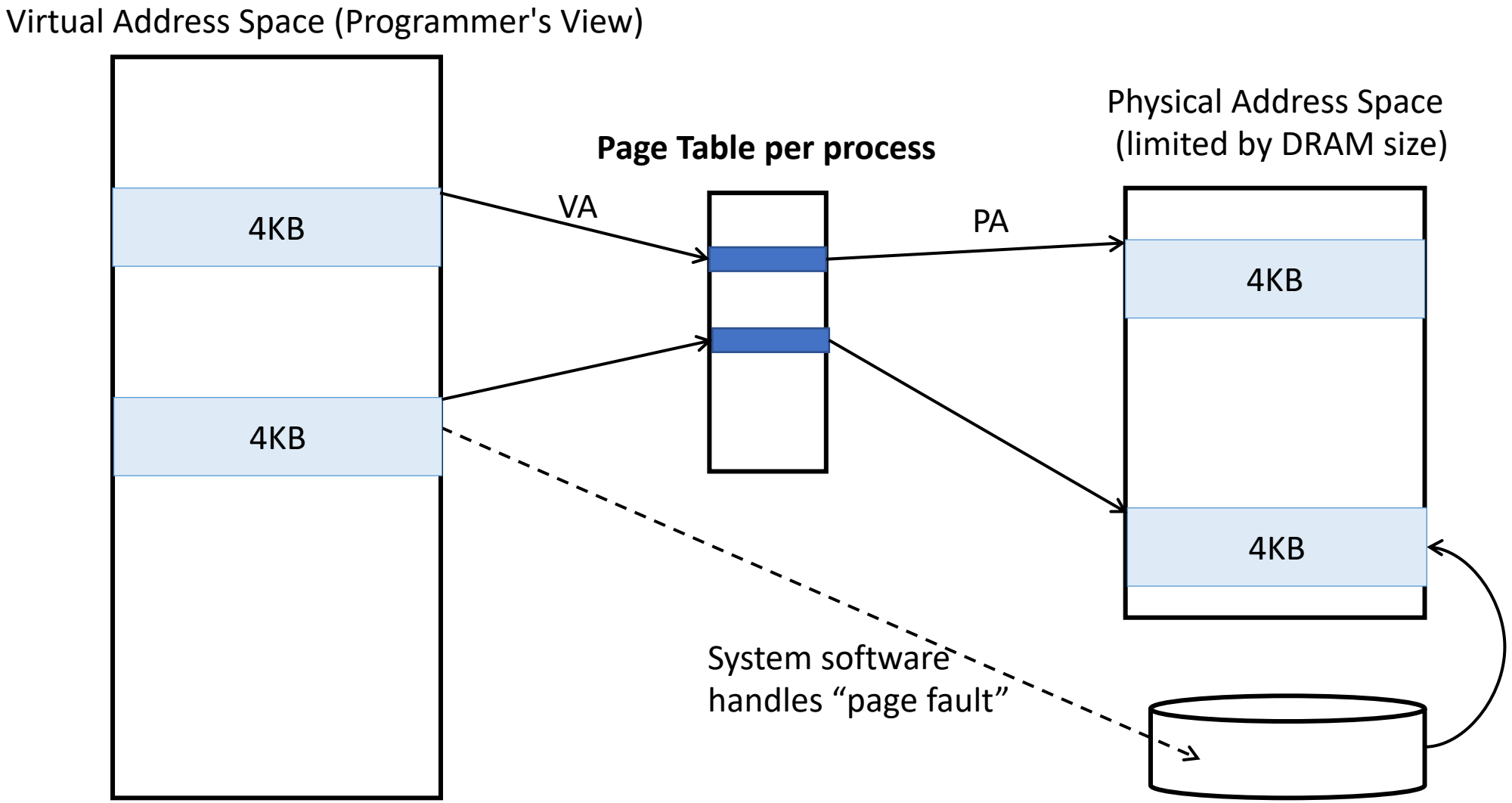


Attacker (OS) can:

- Manage page tables
- Evict TLB entries
- Evict page walk cache entries
- Monitor side channels

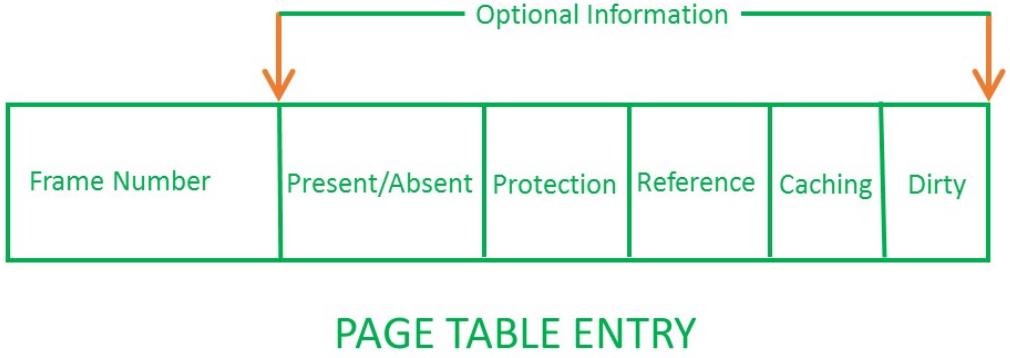
Attack Surface

Recap: Address Translation



Background: Page Fault

- Page fault: access to a page that is
  - Unmapped
  - Invalid
  - Wrong access rights



- Exception is generated → Run page fault handler
  - Page fault handler = Operating system (untrusted)

Controlled Side Channels

- OS can monitor enclaves access pattern at the granularity of page
  - After enclave start, remove access from all process pages (mark page not present)
  - Access will cause a page fault
  - Upon receiving a fault, the handler:
    - Logs the requested page
    - Enables access to the page
    - Removes access to the previous page

```
if (secret = 1)
    access page A
else
    access page B
```

Controlled-Channel Attacks: Deterministic Side Channels for Untrusted Operating Systems; Xu et al. S&P'15

Microscope Overview

Motivation: Leakage over side channels

Victim:

```
if (secret)
  use resource
else
  don't use resource
```

Attacker:

```
for ..
   t1 = time()
        use resource
   t2 = time()
```

- Need repeated measurements to be confident → Denoise
- However, many applications run only once → Attacker gets 1 measurement
- Can attackers really extract secrets?

Overview: Microarchitectural Replay Attacks

- Attacker leverages speculative execution
  - To repeatedly replay a snippet of victim code
  - That runs only once

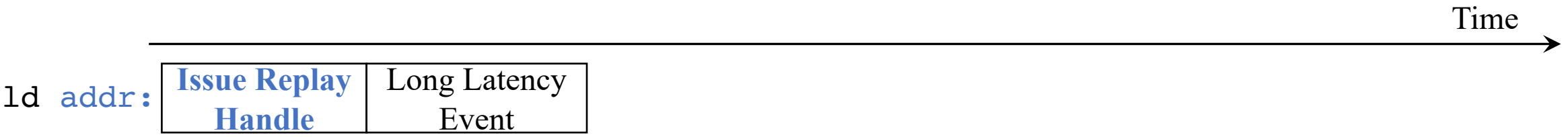
} Primitive to denoise arbitrary side channels

Victim:

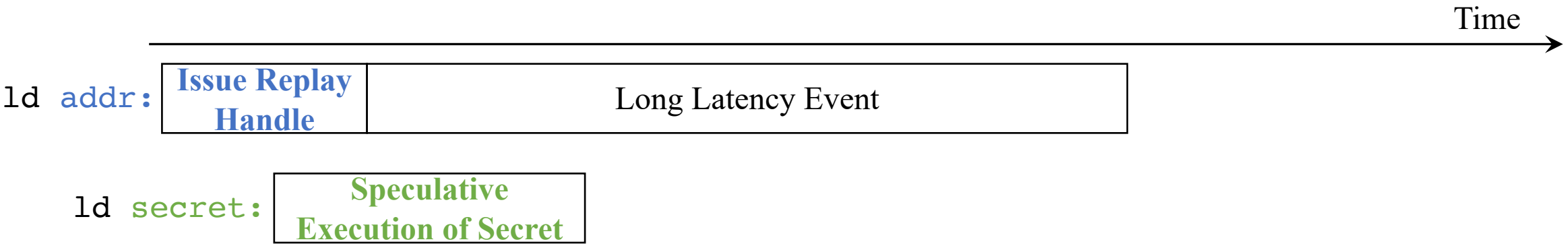
```
ld addr // “replay handle”
…
ld secret // secret the attacker tries to leak
```

{ Memory operation that will cause a squash and re-execute }

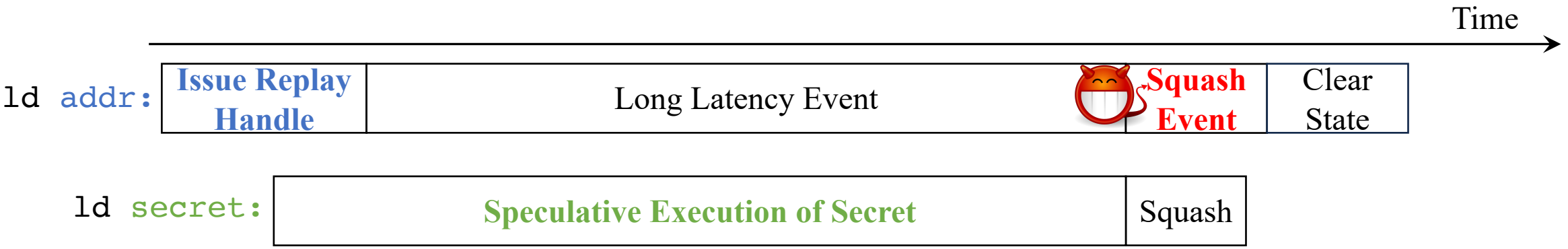
Contribution: Microarchitectural Replay Attacks



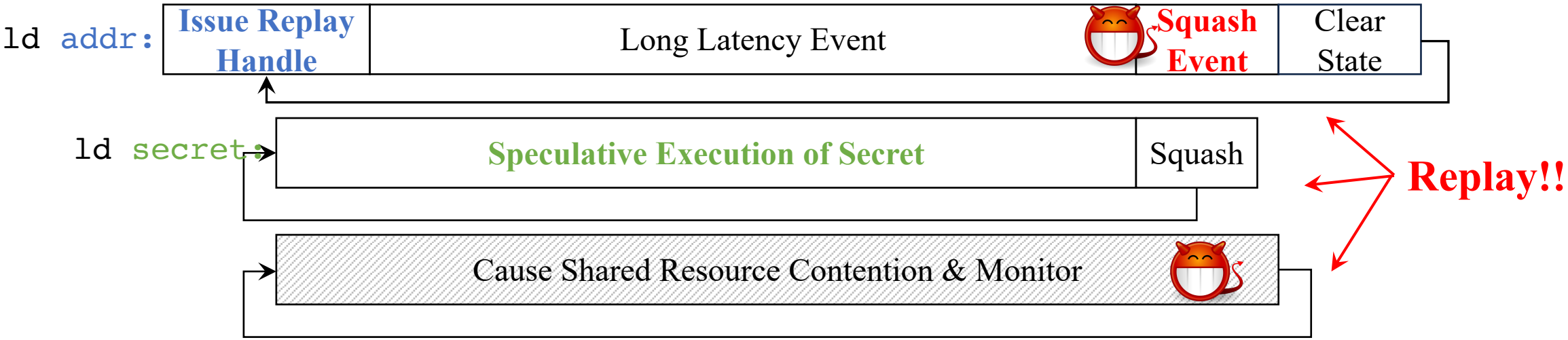
Contribution: Microarchitectural Replay Attacks



Contribution: Microarchitectural Replay Attacks



Contribution: Microarchitectural Replay Attacks



Strengths

- Opens large new attack surface (for noisy side channels)

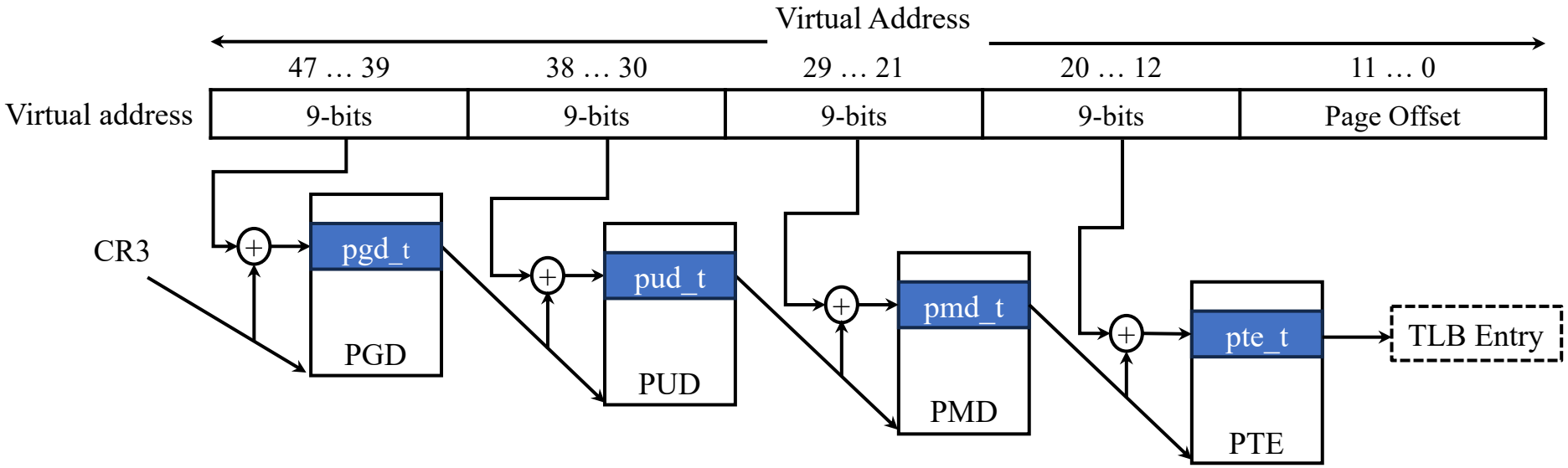
- Exploits vulnerabilities of correct speculation
  - Dynamic instructions can be replayed through controlled squashes
  - Different from Spectre/Meltdown that exploits incorrect speculation

- Demonstrate attacks on notoriously noisy side channels
  - Make impractical attacks possible

Weaknesses

- Is it really practical?
  - Attacker side:
    - Malicious OS
    - Control TLB/page mapping
  - Victim side:
    - The replay handler and the transmitter need to be in the ROB simultaneously
    - The replay handler and the transmitter needs to access different pages

Page Tables Background



- Page tables stored in memory
- On a TLB Miss → "*page walk*" = memory accesses
  - Each step of page walk = cache hit/miss.
  - Page walk cache (PWC): hardware cache of translations
- If Present bit in pte_t is cleared → Page Fault, invoke OS

Attack Examples

Victim Code

```
1. //public address
2. handle(pub_addr);
3. ...
4. transmit(secret);
5. ...
```

Loop Victim Code:

```
1. for i in ...
2.   handle(pub_addrA);
3.   ...
4.   transmit(secret[i]);
5.   ...
6.   memOp(pub_addrB);
7.   ...
```

Terminology

Victim Code

```
1. //public address
2. handle(pub_addr);
3. ...
4. transmit(secret);
5. ...
```

Replay handle:

- Load to a public address (known to OS)

Transmitter:

- Any instruction(s) whose execution reveals secret through some side channel
- Occurs < ROB length from Replay Handle

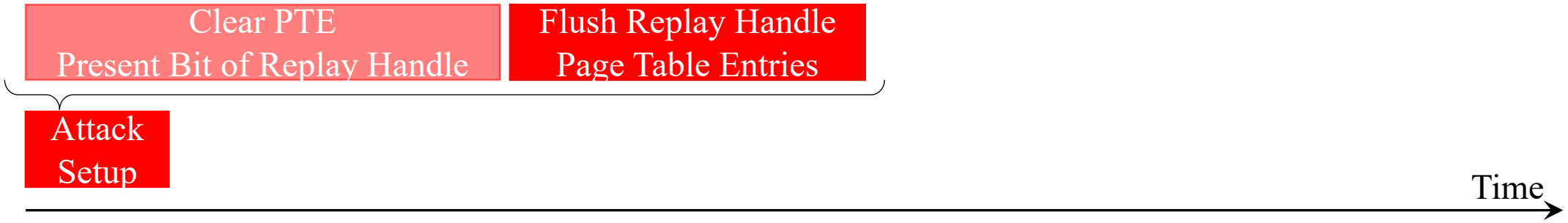
Timeline of a MicroScope Attack - Setup

Attacker Victim

Timeline of a MicroScope Attack - Setup

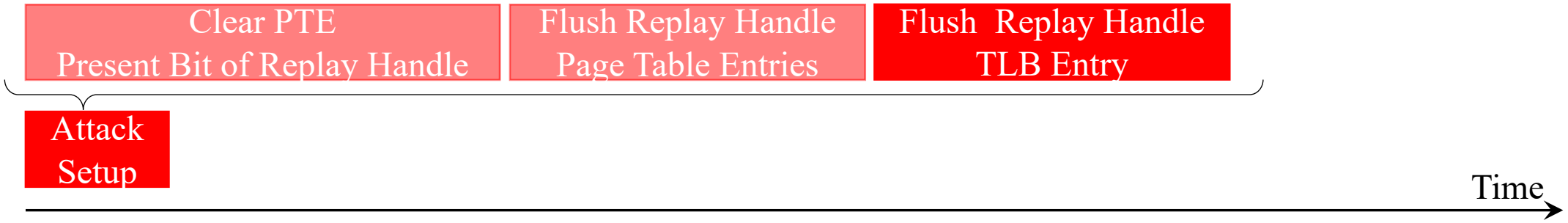
Attacker Victim

Timeline of a MicroScope Attack - Setup



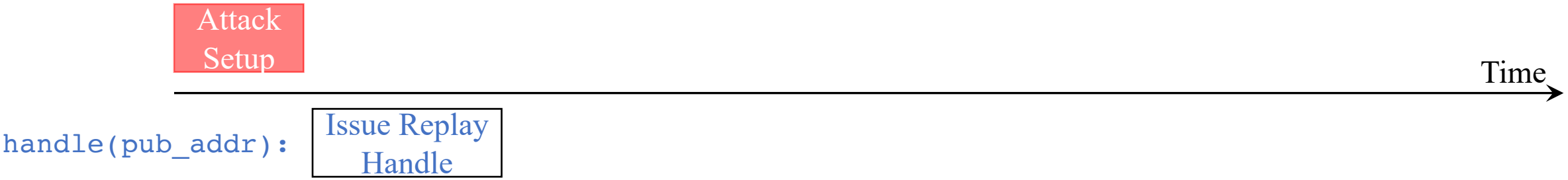
Attacker Victim

Timeline of a MicroScope Attack - Setup



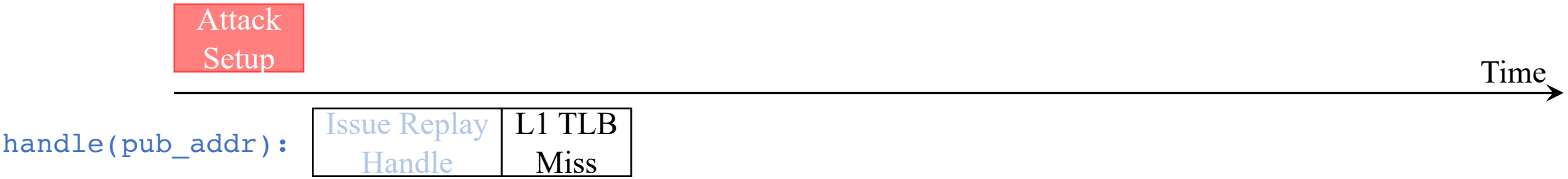
Attacker Victim

Timeline of a MicroScope Attack



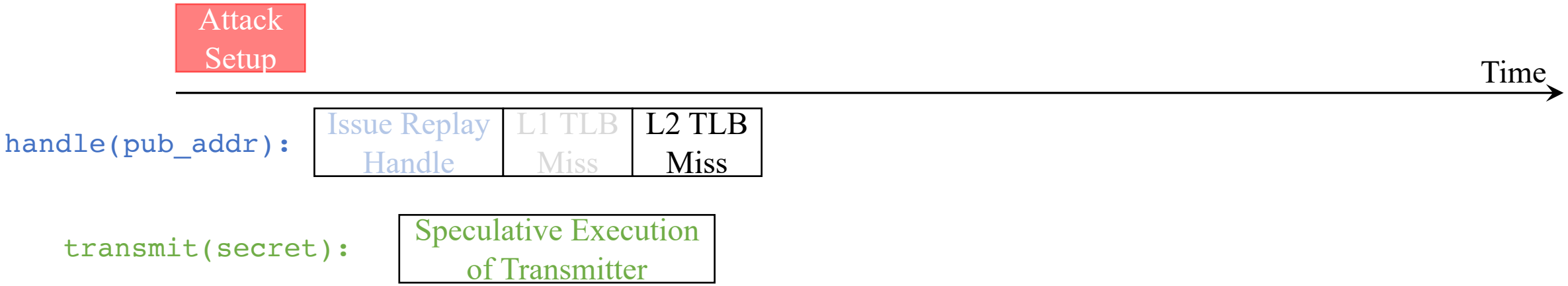
Attacker Victim

Timeline of a MicroScope Attack



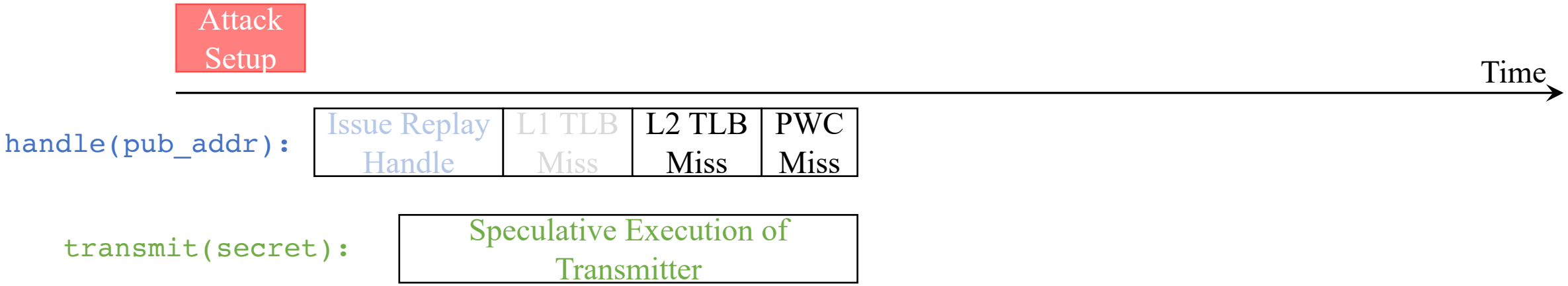
Attacker Victim

Timeline of a MicroScope Attack



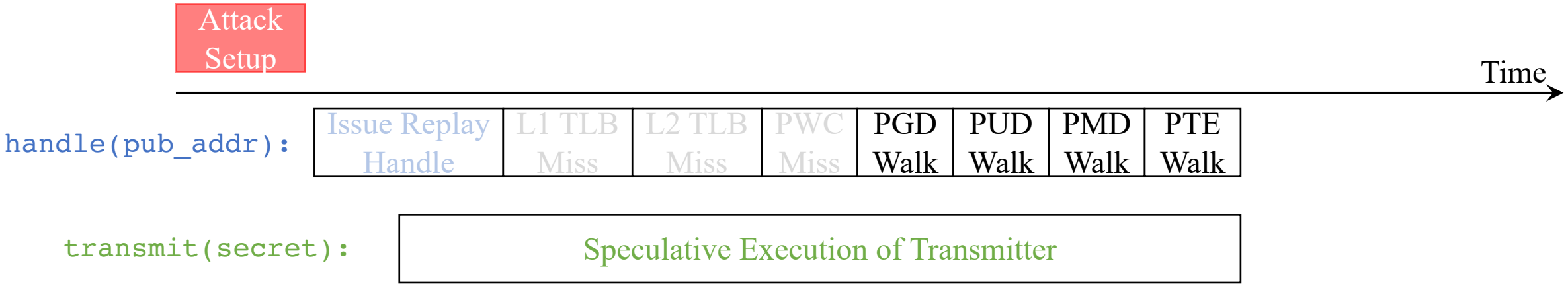
Attacker Victim

Timeline of a MicroScope Attack



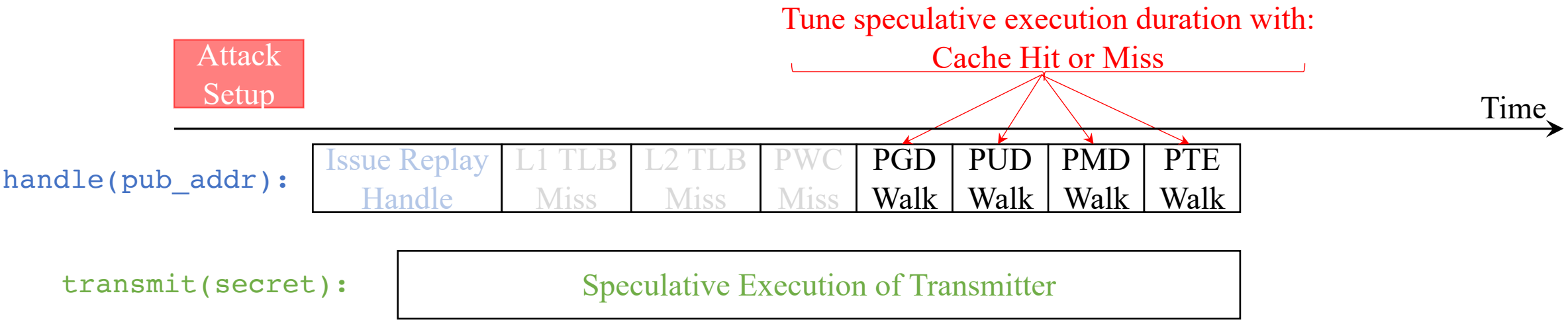
Attacker Victim

Timeline of a MicroScope Attack



Attacker Victim

Timeline of a MicroScope Attack



Attacker Victim

Timeline of a MicroScope Attack

Attacker Victim

Timeline of a MicroScope Attack

Attacker Victim

Timeline of a MicroScope Attack

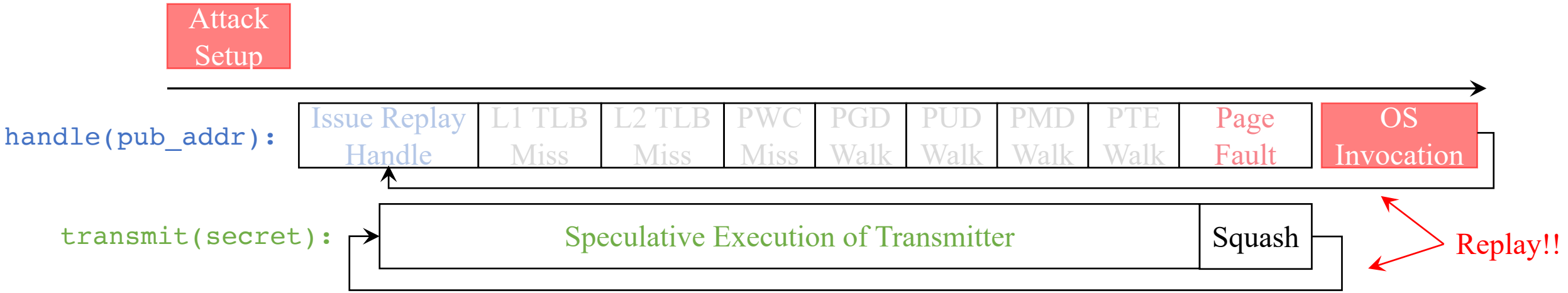
Attacker Victim

Timeline of a MicroScope Attack

Timeline of a MicroScope Attack

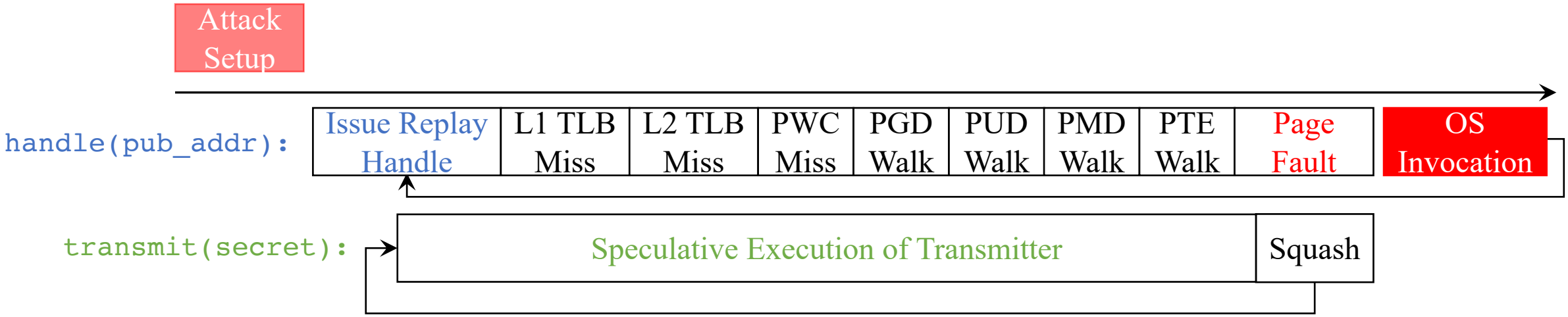
Attacker Victim

Timeline of a MicroScope Attack



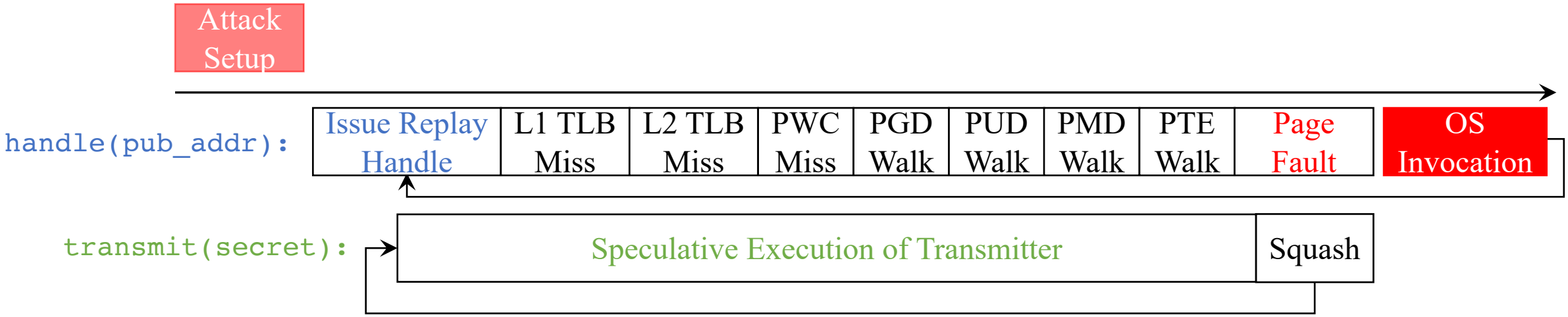
Attacker Victim

Timeline of a MicroScope Attack



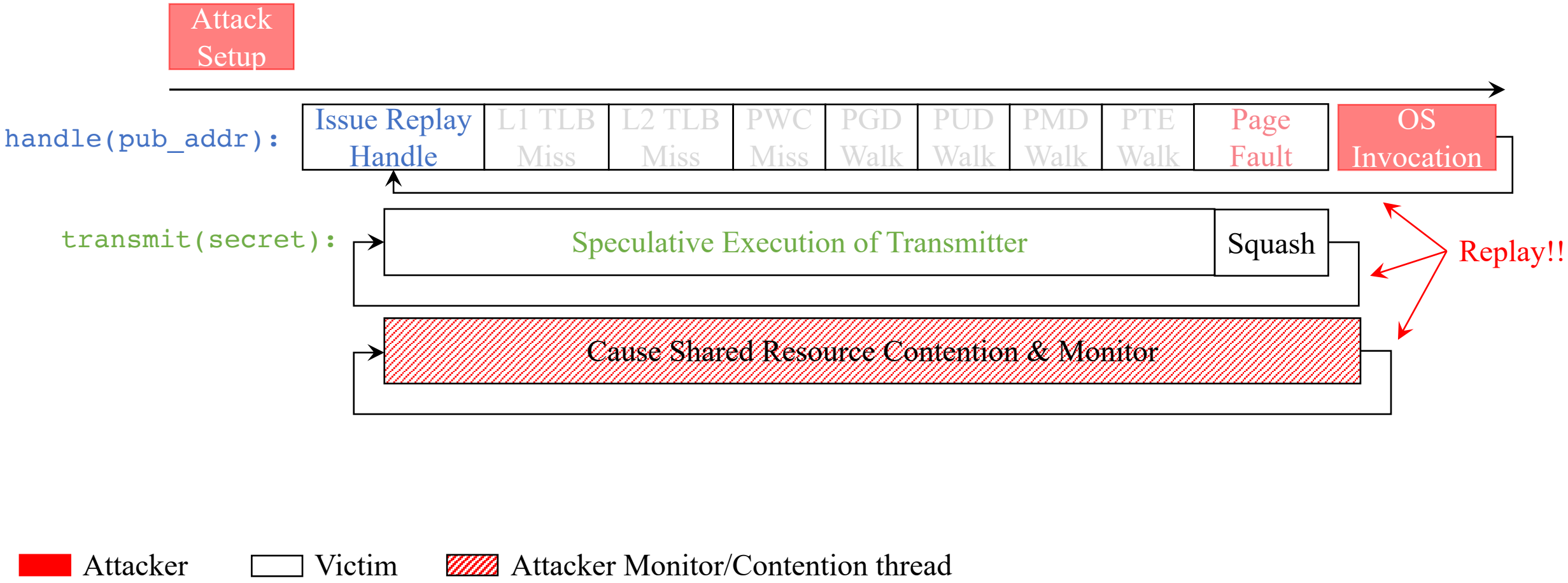
Attacker Victim

Timeline of a MicroScope Attack

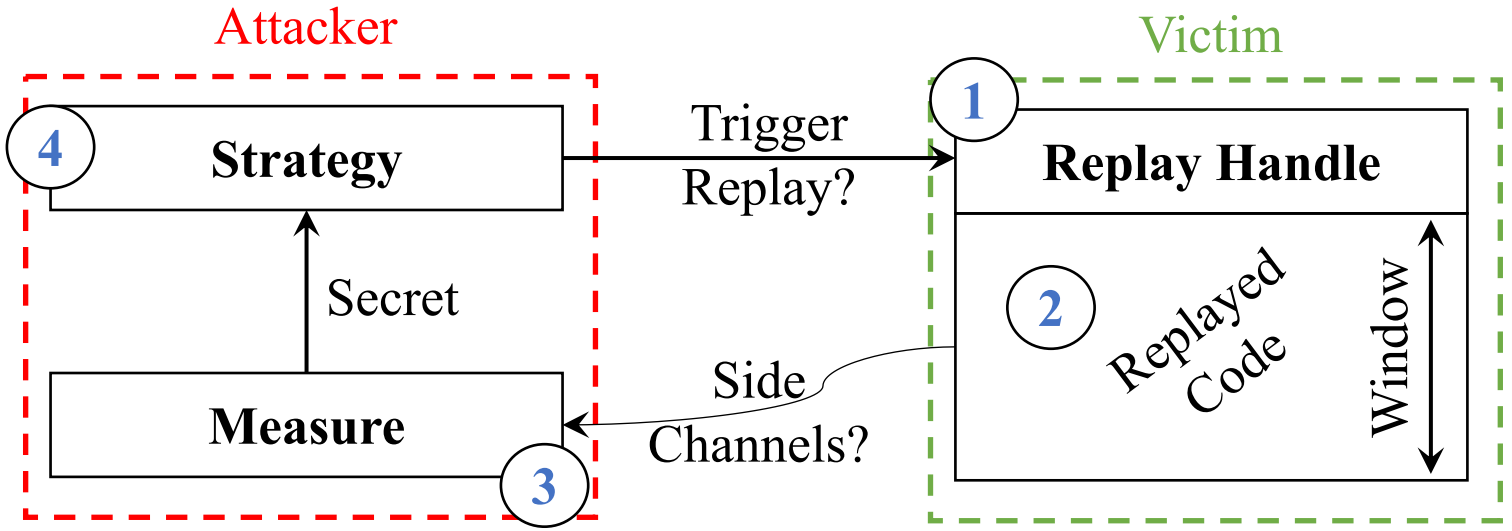


Attacker Victim

Timeline of a MicroScope Attack



Generalize Microarchitectural Replay Attacks



This work:

1. Replay Handle → Page fault-inducing load
2. Replayed Code → Leaky instruction
3. Side Channel → uarch structures
4. Attacker strategy → Page fault until denoise

Changing each can result in different attacks!!

Countermeasures

- Fence after pipeline squash
- Defenses against Spectre/Meltdown style of attacks
- Rewrite victim code to make replay handler and target code reside in the same page
- etc

Discussion Questions

Discussion Questions on Countermeasures

- What is required to prevent this in hardware? Would a form of page fault counter be appropriate, where if a specific instruction page faulted some number of times in a row, the application terminates? Or is this a common scenario in a real process, that a single page may fault repeatedly?

- Would something like FaCT for the SGX application help prevent a significant subset of the available side channels? Or really any other way to make sure that the instruction trace is always constant...

- Are the weaknesses of SGX things that can be patched over as new attacks are demonstrated or is there a more fundamental problem with an untrusted OS? More out of curiosity, but are there adversaries out there trying to exploit these kinds of vulnerabilities right now, and if so how and in what context?

Discussion Questions on Countermeasures

- When referring to page fault protection schemes, why can’t we control how the present bit is set? A key component of this attack is the attacker’s ability to clear the Present bit so would it not be possible to focus on this aspect?

- The paper mentioned that T-SGX terminates the program after N=10 consecutive failed page faults as a potential defense to this type of attack. Was 10 chosen arbitrarily? How did they guarantee that this wouldn't interfere with existing programs? If they set it to a smaller number in order to prevent replay attacks, how would they ensure it would continue to let existing programs work?

- How difficult is it to manipulate where replay handles occur? Can user code force secrets to be contained within a single page? Can user code avoid speculatively affecting side channels by adding data dependency across pages?