Security Vulnerabilities of Caches

Unveiling your keystrokes: A Cache-based Side-channel Attack on Graphics Libraries

Daimeng Wang*, Ajaya Neupane*, Zhiyun Qian*, Nael Abu-Ghazaleh*, Srikanth V. Krishnamurthy*
Edward J. M. Colbert†, and Paul Yu‡
*University of California Riverside. {dwang030, ajaya, zhiyunq, nael, krish}@cs.ucr.edu
†Virginia Tech. ecolbert@vt.edu
‡U.S. Army Research Lab (ARL). paul.l.yu.civ@mail.mil

***Abstract*—Operating systems use shared memory to improve performance. However, as shown in recent studies, attackers can exploit CPU cache side-channels associated with shared memory to extract sensitive information. The attacks that were previously attempted typically only detect the presence of a certain operation and require significant manual analysis to identify and evaluate**

i.e., different virtual pages are mapped to the same physical pages. This creates an opportunity for a malicious process to infer graphics-related activities of a victim process.

Our intuition of the attack is that the performance of graphics rendering is critical for user experience across a

Image credit: https://www.ndss-symposium.org/ndss-paper/unveiling-your-keystrokes-a-cache-based-side-channel-attack-on-graphics-libraries/

Outline

- On studying attacks and security
- The premise of the attack
- Introducing the exploitable memory system…

Disclaimer on Teaching Attacks…

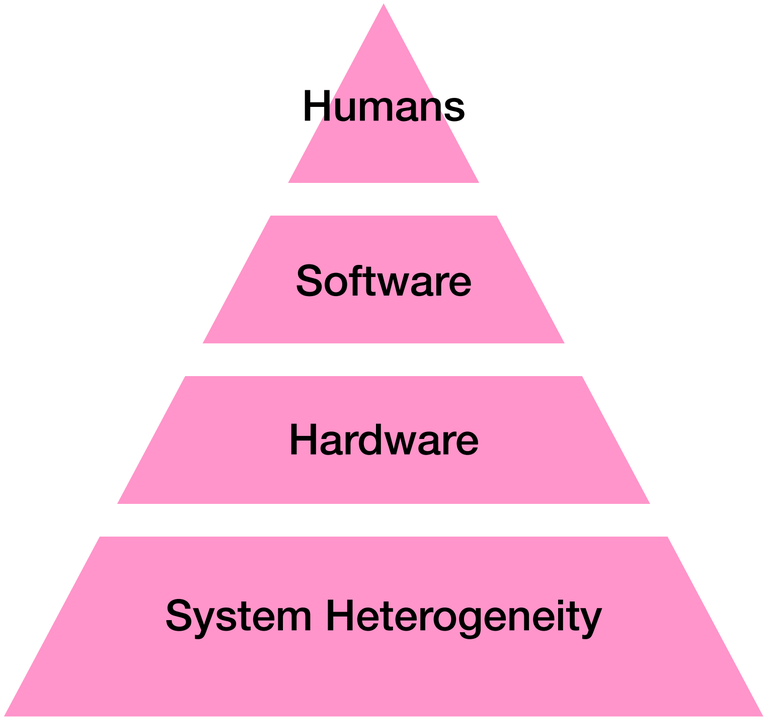
- We are going to describe published, well-studied literature about dangerous and powerful attacks
- These attacks are intuitive and easy to perform (they do not require high levels of privilege or complicated code to execute)
- One of the learning goals of this class is to study the trade-offs of design decisions made
  - ➡️ security is an often neglected component of this design space trade-off, so we will use it as a means to think deeply about the concepts that we have covered and learned thus far

Chat with your neighbor(s)!

Share your comfort levels of describing and learning about attacks.
1️⃣ How security conscious do you feel you are?
2️⃣ Do you feel like studying and publishing attacks should be encouraged (e.g., more awareness towards building defenses) or discouraged (e.g., don’t notify attackers of unknown vulnerabilities)?

The Security Stack

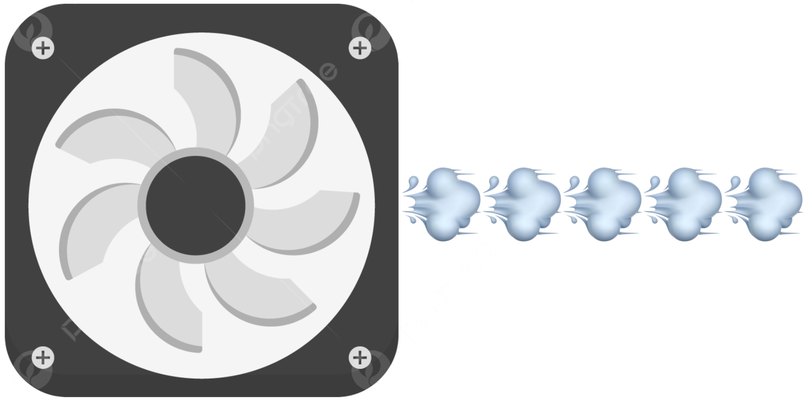
- In order to understand how to build secure systems, we should think about ways in which they are vulnerable
- Different levels of threat imply different degrees of defenses
- In general, defenses at one level of abstraction do not apply to the next level of vulnerability
- Different degrees of threat may apply to different computing contexts



Defining Side Channels

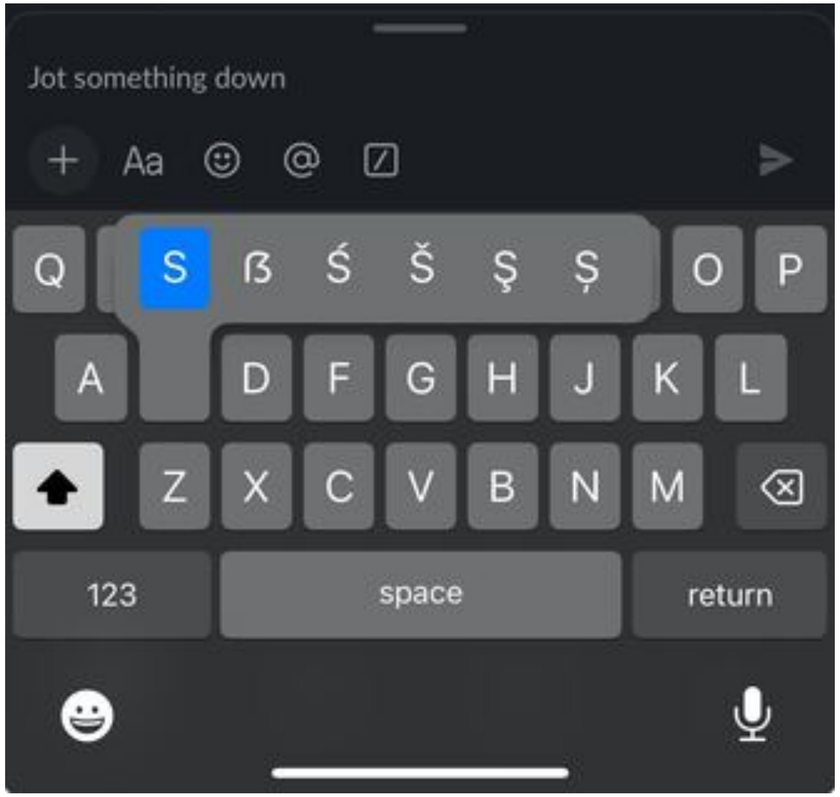
- A *side channel* describes incidental information leakage that can be inferred from observing normal execution
- How does hardware leak information?
  - Noise! 🐓🕺
  - Heat (and power dissipation)! ☀️
  - Timing! ⏱️
- Adversaries need to have some notion of meaning associated with the information that is leaked by the behavior

```
for (;;) {
  // super intense computation!
}
```



The Premise of the Attack

- If an adversary can learn your keystrokes, then they can know anything that you type (e.g., passwords, unsubmitted searches, etc)!
- When pressing keys on a virtual keyboard (i.e., on a touchscreen), typically the pressed key is highlighted
- To perform the “highlight” operation, the keyboard includes and executes code from a graphics library that will update the display



Identifying Vulnerable Code

Source from libcairo.so which is used in Ubuntu Linux at the beginning and end of calling "renderStart" and "renderEnd"!

```
static void D32_LCD32_Opaque(...) {
    ...
    do {
        blit_lcd32_opaque_row(dstRow, srcRow, color, width);
        dstRow = (SkPMColor*)((char*)dstRow + dstRB);
        srcRow = (const SkPMColor*)((const char*)srcRow + maskRB);
    } while (--height != 0);
}
```

```
static void blit_lcd32_opaque_row(dst, src, color, width) {
    ...
    for (int i = 0; i < width; i++) {
        if (0 == src[i]) {
            continue;
        }
        ...
    }
}
```

Image credit: https://www.ndss-symposium.org/ndss-paper/unveiling-your-keystrokes-a-cache-based-side-channel-attack-on-graphics-libraries/

Memory access by calling src[i]

Asymmetric timing depending on the state of the data!

Identifying Vulnerable Code

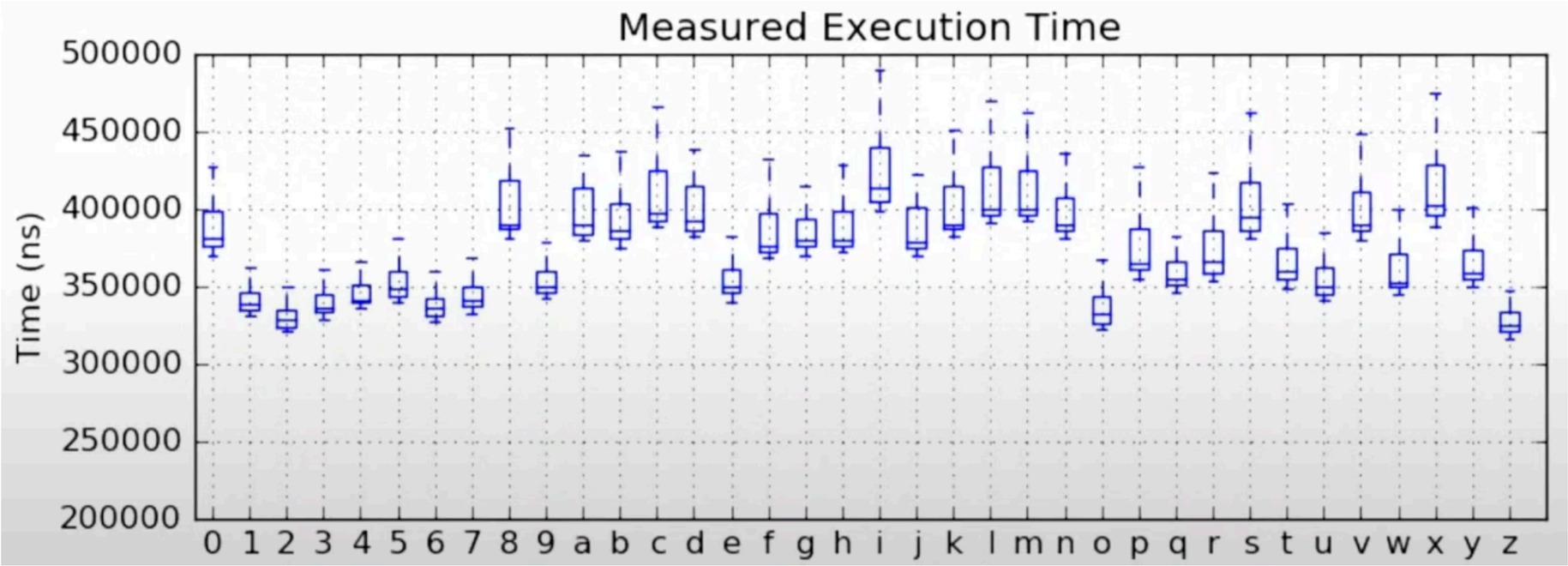


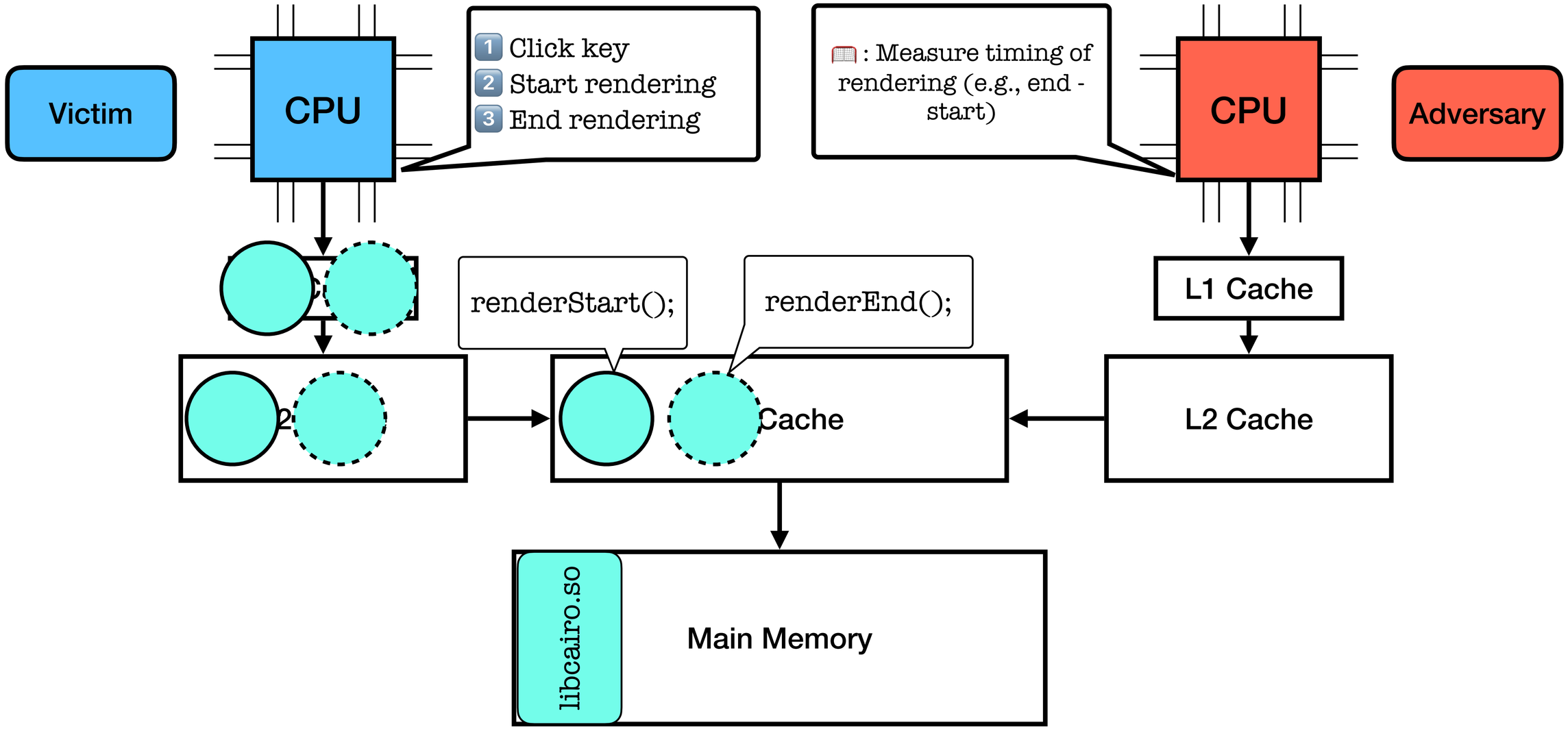
Image credit: https://www.ndss-symposium.org/ndss-paper/unveiling-your-keystrokes-a-cache-based-side-channel-attack-on-graphics-libraries/

Different characters to render means that there are different amounts of whitespace, so the timing to render different characters will be distinct

Profiling the Attacker (Threat Model)

- Goal: extract sensitive information that might otherwise be protected via encryption or other cryptographic encoding while stored or in transmission
- Capability: access to the libraries and hardware versions of the victim
  - ➡️ libraries are often open source, so it is easy to know what will be run
  - ➡️ this allows the adversary to do behavioral profiling offline before deploying the attack
- Capability: be able to run legitimate guest code alongside the victim's process
  - ➡️ this can be in the website's code itself, a third-party cookie, an external application, etc…

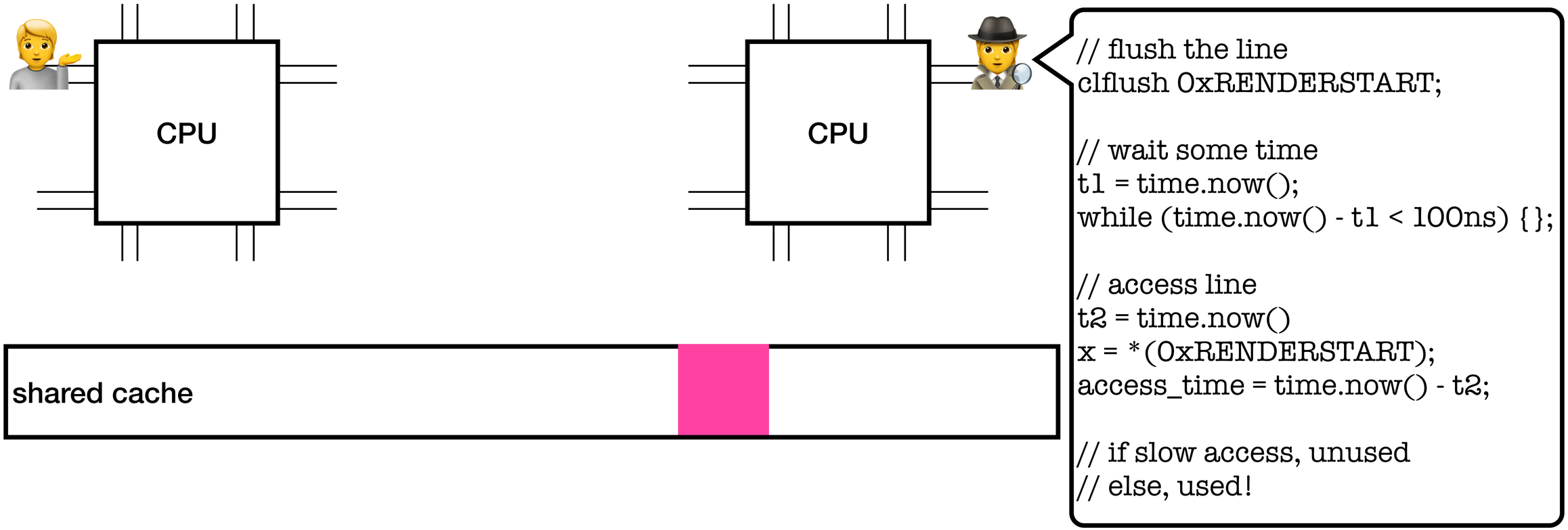
Shared Memory Model



Chat with your neighbor(s)!

What are some of the challenges for an adversary who wants to measure when a user calls `renderStart` and `renderEnd`?

Flush + Reload Attack



Flush + Reload Attack

- To implement such an attack, an adversary needs to be able to use very precise timers (nanosecond granularity) and be able to execute completely in parallel with the victim
- x86 gives access to the `clflush` instruction to be able to explicitly interact with long term storage and write certain data through to non-volatile storage
- Potential mitigation: What happens if the ISA and/or hardware doesn't allow for explicit flushing instructions or this granularity of timing?

Prime + Probe Attack

- Unfortunately, we are not safe with these ISA defenses!
- While, we may not be able to time individual accesses, but we can measure the impact of misses due to *cache contention*