CSE 484/M584: Computer Security (and Privacy)

Spring 2025

David Kohlbrenner
dkohlbre@cs

UW Instruction Team: David Kohlbrenner, Yoshi Kohno, Franziska Roesner, Nirvan Tyagi. Thanks to Dan Boneh, Dieter Gollmann, Dan Halperin, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials

Admin

- Lab 4 : A+B Due Today
  - Separate assignments, so separate late day usage.
  - If this is not what you expected and it throws off your late day usage, email.
- Lab 4 : C Due in a week
  - These patches are slightly trickier
    - An ideal patch for each is (usually) more than just a couple lines of code.
    - With proper documentation sometimes a very short patch gets a great score.
  - You *must* pass the autograder if you want to get a reasonable grade.
    - It is very rare that a patch that doesn’t get a perfect autograder score does well.
- Next week’s in-class:
  - Due on Friday, not 1-week.

Final Exam

- June 9th, 8:30am here in G20.
  - There may be assigned seating, we're working that out.
- Next Thursday section will do exam prep/discussion
  - Don't wait until then to review materials.
- We will release some amount of sample exam questions in time for section, or before.
  - We're still writing the final, so can't release them yet.

Final Exam study notes

- Binary Security:
  - Go over any code examples we did in class, understand the fundamental problem with each.
  - Review your Lab 1 writeups (understand why those bugs work.)
  - Practice drawing out and reasoning about simple function stack diagrams.
  - Review the optional Lab 1 exercises (buffer exercise) if you didn’t before.
  - Review the discussion of defenses and hardening mechanisms.

  - In-class discussion questions are great starting point for many of these.

Final Exam study notes

- Cryptography:
  - Understand the fundamental security properties we discussed, and how each type of cryptographic tool relied on them.
  - Review hashing and hashing properties.
  - Review RSA and Diffie-Hellman operation.
  - Be able to do a simple RSA or DH operation in Z_p* (very small numbers)
  - Review block cipher properties and why/how we build block cipher modes.

  - In-class discussion questions are great starting point for many of these.
  - Lab 2 short answer questions as well.

Final Exam study notes

- Web Security:
  - Review the same-origin model.
  - Understand what cookies are used for, how they are set, etc.
  - Work through the XSS and CSRF flows we've done, and be able to identify the actors and actions for each.
  - Review your Lab 3 writeups. Be able to articulate how to launch an XSS or CSRF attack and what the 'naïve.com' site needs to have for that to work.

  - In-class discussion questions are great starting point for many of these.

Final Exam study notes

- Tracking, Anonymity and Usability:
  - Review what anonymity means, and when you can have it.
  - Review tracking approaches and defenses.
  - Understand the basics of how Tor and mixnets provide privacy properties (and when they don't.)
  - Be able to discuss the usability problems with security indicators, and why different security mechanisms have different ways to communicate their status.

  - In-class discussion questions are great starting point for many of these.

Side channels

Side-channels: conceptually

- A program’s implementation (that is, the final compiled version) is different from the conceptual description
- Side-effects of the difference between the implementation and conception can reveal unexpected information
  - Thus: Side-channels

Attacker Model

```
PwdCheck(RealPwd, CandidatePwd)  // both 8 chars always
    for(int i=0; i<8; i++){
        if (RealPwd[i] != CandidatePwd[i])
            return FALSE;
    }
    return TRUE;
```

- Attacker can guess CandidatePwds through some standard interface
- Naive: Try all 256^8 = 18,446,744,073,709,551,616 possibilities
- Is it possible to derive password more quickly?

Detour: Covert-channels

- We’ll see many unusual ways to have information flow from thing A to thing B
- If this is an *intentional* usage of side effects, it is a covert channel
- *Unintentional* means it is a side-channel
- The same *mechanism* can be used as a covert-channel, or abused as a side-channel

Detour: Covert-channels

Sender:

```
Msg = [0,1,1,1,0]
For bit in msg:
    if bit == 0:
        sleep(1);
    else:
        use_disk_for_1s();
```

Receiver:

Detour: Covert-channels -- Gradescope

Sender:

Receiver:

```
Msg = [0,1,1,1,0]
For bit in msg:
    if bit == 0:
        sleep(1);
    else:
        use_disk_for_1s();
```

Detour: Covert-channels

Sender:

```
Msg = [0,1,1,1,0]
For bit in msg:
    if bit == 0:
        sleep(1);
    else:
        use_disk_for_1s();
```

Receiver:

```
msg=[]
for i in range(5):
    msg[i] = is_disk_busy();
    sleep(1);
```

Side Channel Attacks

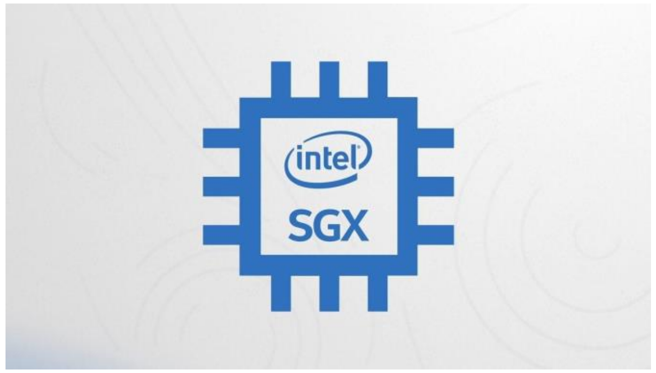
- Most commonly discussed in the context of cryptosystems
- But also prevalent in many contexts
    - E.g., we discussed the concept of a timing channel on password comparison
    - E.g., we discussed browser fingerprinting

Why should we care about side-channels?

- Compromises happen via ‘simple’ methods
  - Phishing
  - Straight-forward attacks

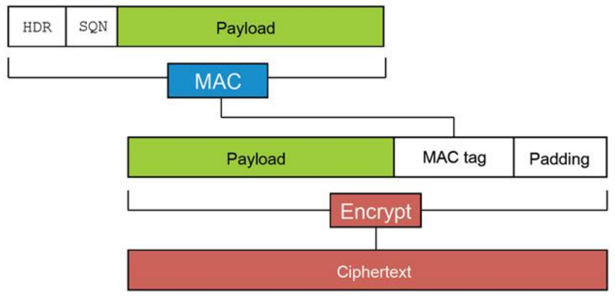
- Embedded systems *do* see side-channel attacks

- “High Security” systems *do* see side-channel attacks



Timing side-channels: round 2

- Cryptographic implementations fall down
  - #1 target for timing attacks
  - Extremely common to find vulnerabilities



- "Timing Attacks on Implementations of Diffe-Hellman, RSA, DSS, and Other Systems"
  - Was very far from the last paper on the topic

Attacking cryptography with side-channels

- ANY leakage is bad
  - E.g. 1 bit of key leaking is ‘catastrophic’

- Cryptographic implementations are complex
  - Many layers of protocols

RSA is deprecated

- Partially because of how hard it is to avoid implementation issues!
    - https://blog.trailofbits.com/2019/07/08/fuck-rsa/

Example Timing Attacks

- **RSA:** Leverage key-dependent timings of modular exponentiations
  - https://www.rambus.com/timing-attacks-on-implementations-of-diffie-hellman-rsa-dss-and-other-systems/
  - http://crypto.stanford.edu/~dabo/papers/ssl-timing.pdf
- **Block Ciphers:** Leverage key-dependent cache hits/misses

Cache side-channels

Cache side-channels

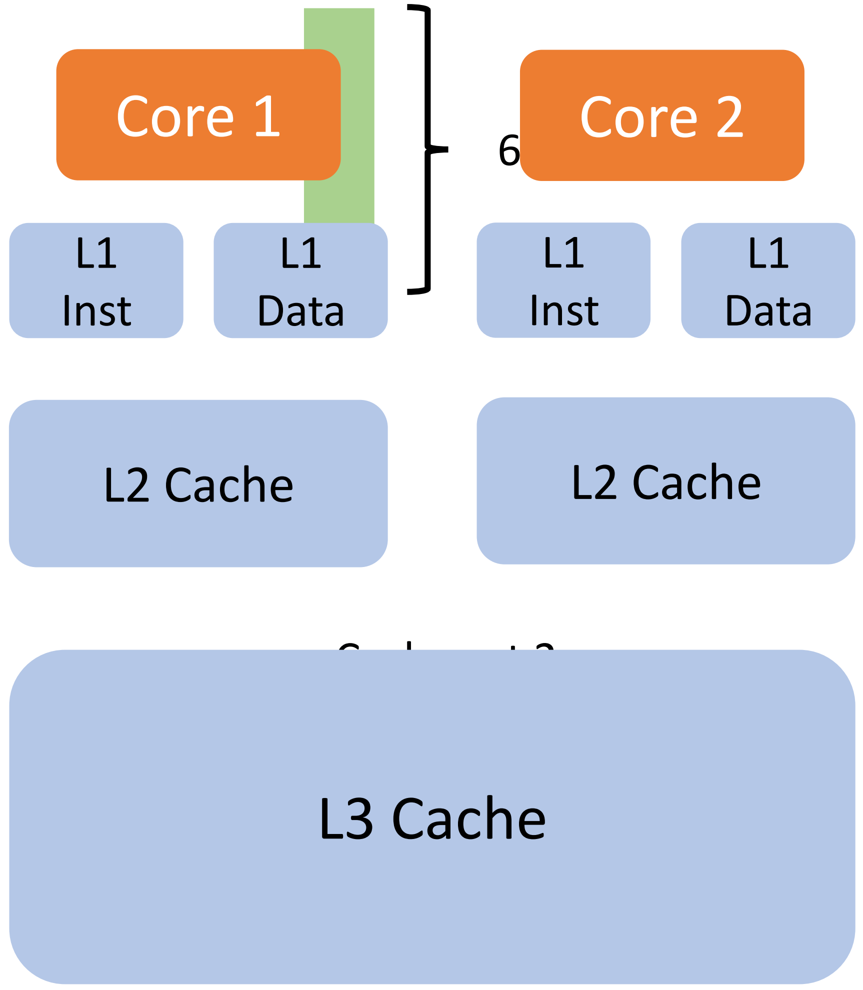
- **Idea**: The cache's current state implies something about prior memory accesses

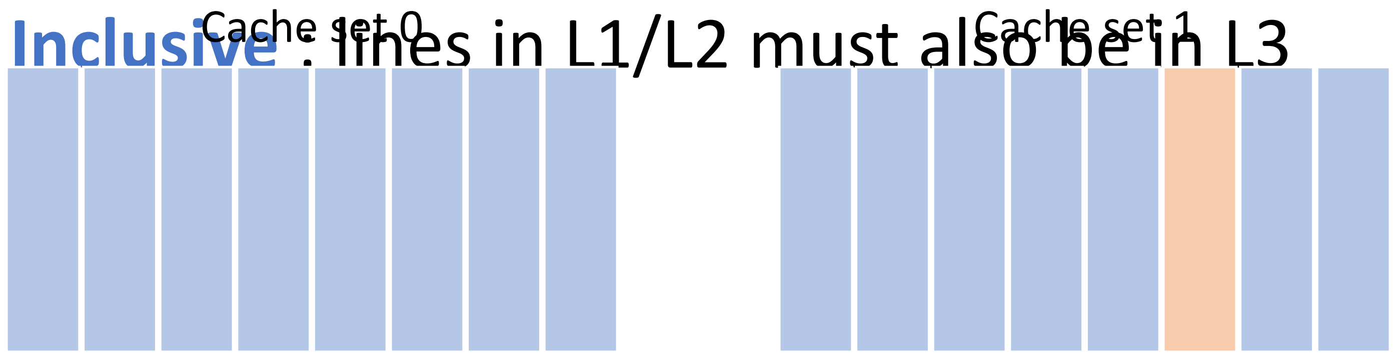
- **Insight**: Prior memory accesses can tell you a lot about a program!

Many thanks to Craig Disselkoen for the animations.

Cache Basics

- **Cache lines** : fixed-size units of data
- **Cache set** : holds multiple cache lines
- **Set index** : assigns cache line to cache set
- **Eviction** : removing cache lines to make room
- **L1, L2, L3** : different levels of cache
- **Inclusive** : lines in L1/L2 must also be in L3





Many thanks to Craig Disselkoen for the animations.

Cache Attacks: Structure

Pre-Attack

Active Attack

Analysis

Many thanks to Craig Disselkoen for the animations.

PRIME+PROBE FLUSH+RELOAD

(requires shared memory)

Pre-Attack

Timing threshold

Eviction set

Active Attack

Prime targeted set

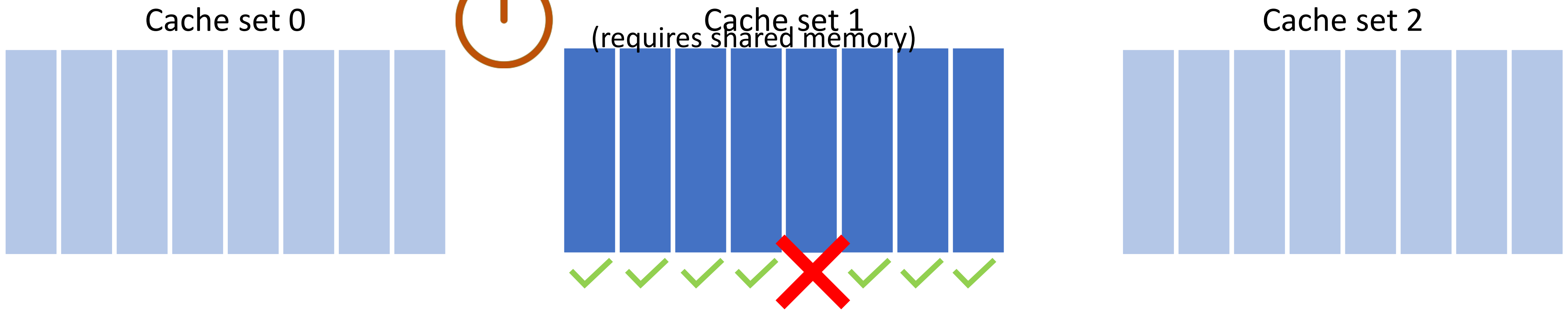
Wait

Victim accesses targeted set

[Timed] Prime targeted set

Analysis

Victim access if time > threshold



🟦 Pre-existing data

🟦 Attacker's data

🟧 Victim's data

FLUSH + RELOAD

- Even simpler!
- Kick line L out of cache
- Let victim run
- Access L
  - Fast? Victim touched it
  - Slow? Victim didn’t touch it

Cache attacks wrapup

- Cache attacks are a core element of many side-channels
- Generally “assumed to work” these days
- New variations/tricks/mitigations published constantly

What is a cache attack good for?

A research topic: microarchitectural side-channels

- What if
  - `mul rax, rcx`
- Takes variable CPU cycles depending on the value of `rcx`
- 2 cycles for not-zero
- 1 cycle for zero
- …