CSE 484 / CSE M 584: Computer Security and Privacy

Autumn 2019

Tadayoshi (Yoshi) Kohno
yoshi@cs.Washington.edu

Thanks to Dan Boneh, Dieter Gollmann, Dan Halperin, John Manferdelli, John Mitchell, Franzi Roesner, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...

Announcements

- My office hours
  - 11/20 (Wed), 2:30pm, CSE1 403
  - 11/27 (Wed), None
  - 12/4 (Wed), 12:30pm, CSE1 403
- Final Project checkpoints looked great! Next Final Project deadline Nov 22
  - Outline + references
  - Doesn't need to be super-detailed
- Lab 2: Nov 22

Announcements

- Quiz section this week:
  - Lab 2 discussion (briefly)
  - Lab 3 discussion (please attend)
- Nov 22: Charlie Reis (Google)
- Nov 27: See website for alternate video lecture
- Dec 4: Seattle PD + US Secret Service

Review: Side Channel Attacks

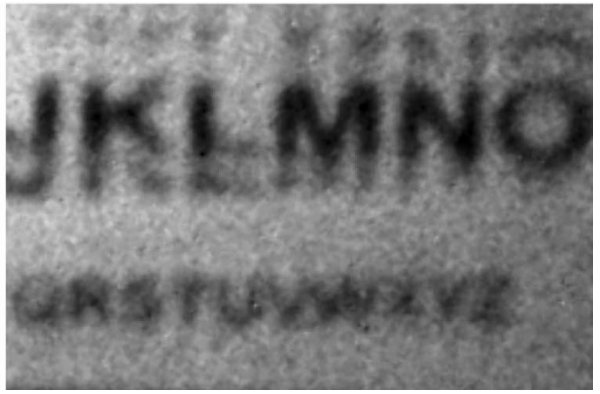
- Attacks based on information that can be gleaned from the physical implementation of a system, rather than breaking its theoretical properties
  - Most commonly discussed in the context of cryptosystems
  - But also prevalent in many contexts
    - E.g., we discussed browser fingerprinting
    - E.g., we discussed history sniffing
    - E.g., we also discussed the TENEX password verification system

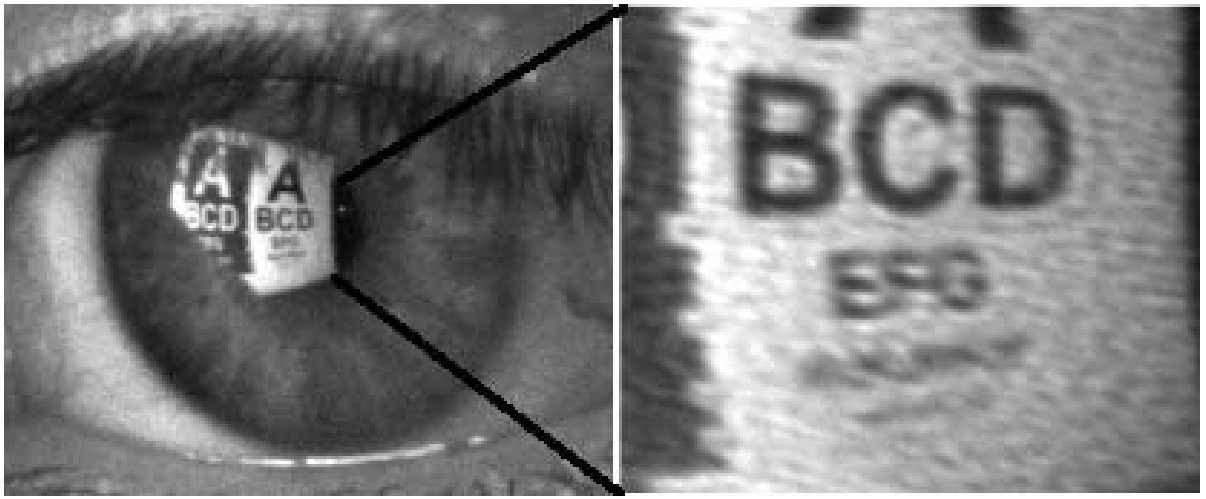
Review: Keyboard Eavesdropping

Zhuang et al. "Keyboard Acoustic Emanations Revisited" CCS 2005
Vuagnoux et al. "Compromising Electromagnetic Emanations of Wired and Wireless Keyboards" USENIX Security 2009

[Backes et al.]

Compromising Reflections





Audio from Video

Davis et al. “The Visual Microphone: Passive Recovery of Sound from Video” SIGGRAPH 2014

Identifying Web Pages: Traffic Analysis

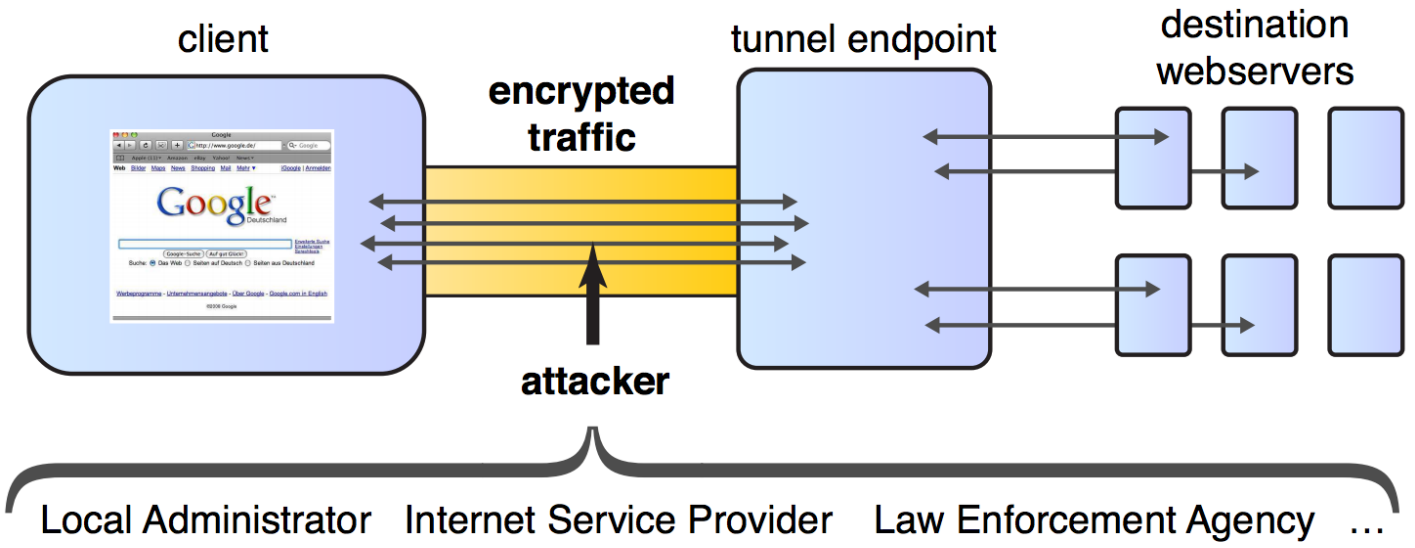
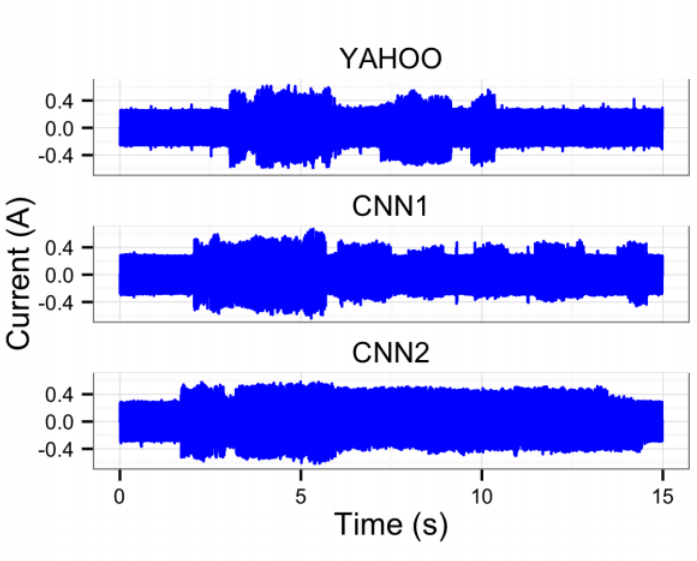


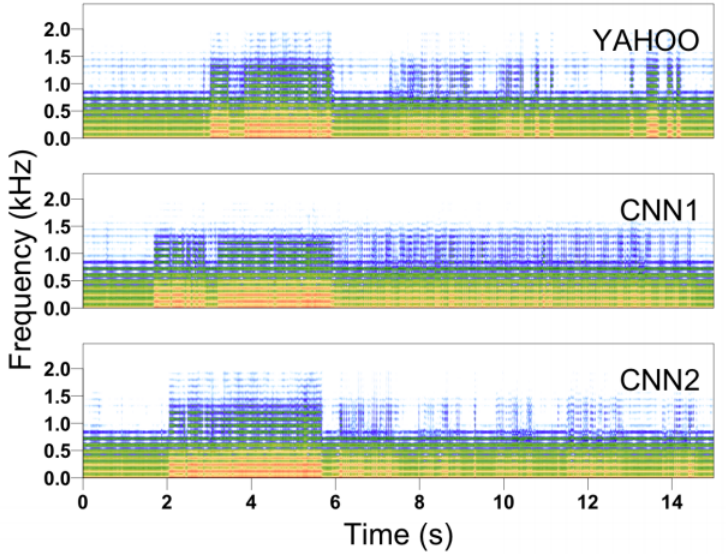
Figure 1: Website fingerprinting scenario and conceivable attackers

Herrmann et al. "Website Fingerprinting: Attacking Popular Privacy Enhancing Technologies with the Multinomial Naïve-Bayes Classifier" CCSW 2009

Identifying Web Pages: Electrical Outlets



(a) Time-domain plots



(b) Spectrogram plots

Fig. 1: Time- and frequency-domain plots of several power traces as a MacBook loads two different pages. In the frequency domain, brighter colors represent more energy at a given frequency. Despite the lack of obviously characteristic information in the time domain, the classifier correctly identifies all of the above traces.

Clark et al. "Current Events: Identifying Webpages by Tapping the Electrical Outlet" ESORICS 2013

Powerline Eavesdropping

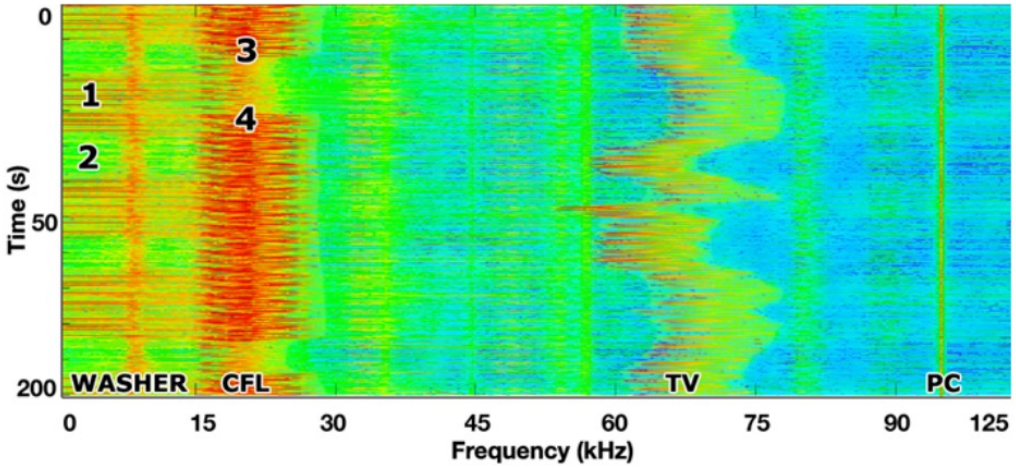


Figure 1: Frequency spectrogram showing various electrical appliances in the home. Washer cycle on (1) and off (2). CFL lamp turning off briefly (3) and then on (4). Note that the TV's (Sharp 42" LCD) EMI shifts in frequency, which happens as screen content changes.

Enev et al.: Televisions, Video Privacy, and Powerline Electromagnetic Interference, CCS 2011

Spectre

- Exploit speculative execution and cache timing information to extract private information from the same process
  - Example: JavaScript from web page trying to extract information from Browser

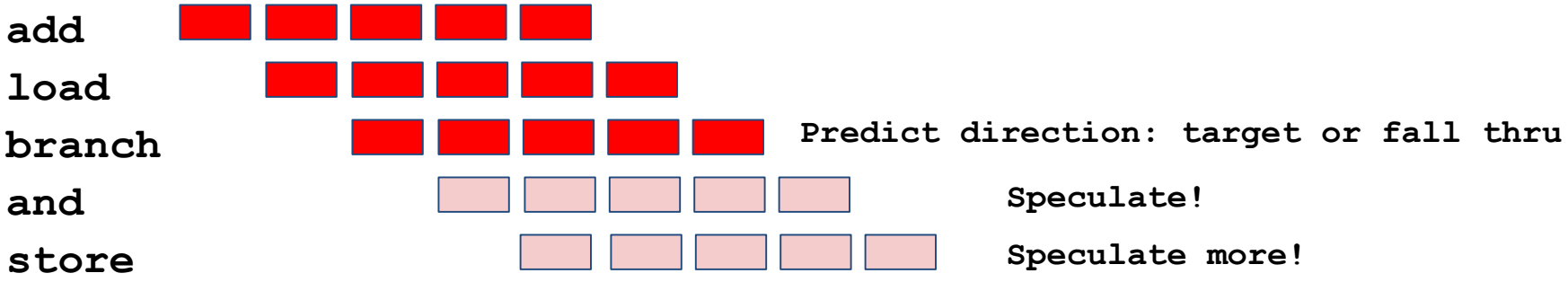
- Architecture Background:
  - Hardware architecture provides "promises" to software
  - Those proposes focus on the functional properties of the software, not performance properties
  - Architectures do a lot to try to increase performance

Material from http://research.cs.wisc.edu/multifacet/papers/hill_mark_wisconsin_meltdown_spectre.pptx

Instruction Speculation Tutorial

Many steps (cycles) to execute one instruction; time flows left to right →

Go Faster: Pipelining, branch prediction, & instruction speculation



Speculation correct: Commit architectural changes of **and** (register) & **store** (memory) go fast!

Mis-speculate: Abort architectural changes (registers, memory); go in other branch direction

Material from http://research.cs.wisc.edu/multifacet/papers/hill_mark_wisconsin_meltdown_spectre.pptx

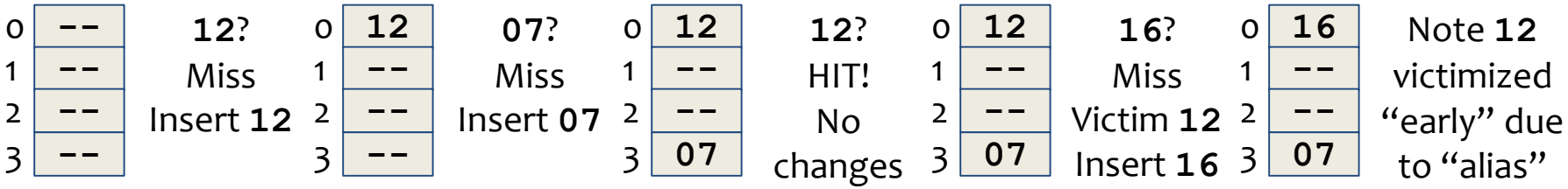
Hardware Caching Tutorial

Main Memory (DRAM) 1000x too slow

Add Hardware Cache(s): small, transparent hardware memory

- Like a software cache: speculate near-term reuse (locality) is common
- Like a hash table: an item (block or line) can go in one or few slots

E.g., 4-entry cache w/ slot picked with address (key) modulo 4



Spectre (Worksheet)

- Consider this code, running as a kernel system call or as part of a cryptographic library.

```
if (x < array1_size)
        y = array2[array1[x] * 256];
```

- Suppose:
  - That an adversary can run code, in the same process.
  - That an adversary can control the value x.
  - That an adversary has access to array2.
  - That the adversary's code cannot just read arbitrary memory in the process.
  - That there is some secret value, elsewhere in the process, that the adversary would like to learn.

- Can you envision a way that an adversary could use their own code, to call a vulnerable function with the above code, to learn the secret information? Leverage branch prediction and cache structure / timing.

Spectre: Key Insights

- Train branch predictor to follow one branch of a conditional
- After branch predictor trained, make the followed branch access information that the code should *not* be allowed to access
- That access information will be loaded into the cache
- After the hardware determines that the branch was incorrectly executed, the logic of the program will be rolled back *but* the cache will still be impacted
- Time reads to cache, to see which cache lines are read more efficiently

Attacker Steps

- Attacker: Execute code with valid inputs, train branch predictor to assume conditional is true
- Attacker: Invoke code with x outside of array1 , array1_size and array2 not cached, but value at array1+x cached // Attacker goal: read secret memory at address array1+x
- CPU: CPU guesses bounds check is true, speculatively reads from array2[array1[x]*256] using malicious x
- CPU: Read from array2 loads data into cache at an address that depends on array1[x] using malicious x
- CPU: Change in cache state not reverted when processor realizes that speculative execution erroneous
- Attacker: Measure cache timings for array2; read of array2[n*256] will be fast for secret byte n (at array1+x)
- Attacker: Repeat for other values of x

Other Types of Side Channels?