463.16 Side-Channel Attack

Computer Security II
CS463/ECE424
University of Illinois
Some of the top ransomware gangs are deploying a new pressure tactic to push more victim organizations into paying an extortion demand: Emailing the victim's customers and partners directly, warning that their data will be leaked to the dark web unless they can convince the victim firm to pay up.

The same extortion pressure email has been going out to people associated with the University of California, which was one of several large U.S. universities that got hit with Clop ransomware recently. Most of those university ransomware incidents

"Sadly, regardless of whether a ransom is paid, consumers whose data has been stolen are still at risk as there is no way of knowing if ransomware gangs delete the data as they promise," Abrams wrote.
Side Channel Attacks: Two Case Studies

- Keyboard spy via acoustic side channels
- Information leakage via hardware side channels
Extracting Information from Side Channels

- Inferring words typed on the keyboard by analyzing the sound

Keyboard Acoustic Emanations Revisited, Li Zhuang, Feng Zhou, J. D. Tygar, CCS 2005
Intuition: Why could this possibly work?

• Different keystrokes make different sounds
  – Locations
  – Underlying hardware
Threat Model and Challenges

• Attacker has a microphone recording the victim’s typing
  – **Assumptions**: typing English text, **no labeled input**
  – **Goals**: recovering the English text, **inferring random text** (e.g., password)

• Challenges
  – Hard to obtain labeled training data --- no cooperation from the victim
  – Typing patterns can be keyboard specific
  – Typing patterns can be user specific
Threat Model and Challenges

• Attacker has a microphone recording the victim’s typing
  – **Assumptions**: typing English text, **no labeled input**
  – **Goals**: recovering the English text, **inferring random text** (e.g., password)

• Challenges
  – Hard to obtain labeled training data --- no cooperation from the victim
  – Typing patterns can be keyboard specific

**Key Intuition:** the typed text is often not random.
• English words limits the possible temporal combinations of keys
• English grammar limits the word combinations.
How The Attack Works

• Key idea: generating training data automatically
  – Labelling the audio of a key stroke with the actual key

[Diagram showing steps of the attack process]

- Audio
- Keystroke detection
  - Touch Peak
  - Hit Peak
  - Push Peak
  - Release Peak
- Spectrum feature extraction
- Clustering
- Group keystrokes into classes
- Language models
- Match classes with keys
- Re-train on original audio

Supervised learning to train new keystroke classifier
A Combination of Different Learning Methods

- **Unsupervised Learning**
  - Spectrum feature extraction
  - Clustering
  - Group keystrokes into classes

- **Supervised Learning**
  - Re-train on original audio

- **Data Labelling**
  - Language models
  - Match classes with keys

- **Keystroke detection**
  - Audio
  - Supervised learning to train new keystroke classifier
Step 1: Unsupervised Learning

- Unsupervised clustering
  - Feature generation
    - Cepstrum features
  - Clustering into K classes
    - K > N (actual number of keys used)

- Output
  - K unlabeled classes

- Spectrum feature extraction
- Clustering

Group keystrokes into classes

this is the best pizza in town

this is the best pizza in town
Step 2: Context-based Language Model

• Need to label the clusters: which key they represent?

• Assume the victim is typing English text
  – Characters follow certain frequency
  – Actual content follows English spelling and grammar

• Advantages:
  – Use 2-character combination frequency to match classes to keys
  – Use language model (spelling, grammar) to correct mistakes
Details: Context-based Language Model

• Character-level mapping:
  – Hidden Markov Model
  – Produce a probability of keys assigned to classes.
  – Example: “th” vs. “tj”

• Word-level correction:
  1. Spell check
  2. Grammar
     o Tri-gram
Before spelling and grammar correction:

the big money fight has drawn the support of dozens of companies in the entertainment industry as well as attorneys general in states, who fear the film sharing software will encourage illegal activity, stem the growth of small artists and lead to lost jobs and diminished sales tax revenue.

After spelling and grammar correction:

the big money fight has drawn the support of dozens of companies in the entertainment industry as well as attorneys general in states, who fear the film sharing software will encourage illegal activity, stem the growth of small artists and lead to lost jobs and finished sales tax revenue.
A Combination of Different Learning Methods

Unsupervised Learning

- Spectrum feature extraction
- Clustering
  Group keystrokes into classes

Data Labelling

- Language models
  Match classes with keys

Supervised Learning (Feedback-based training)

- Re-train on original audio

Audio

Keystroke detection

Supervised learning to train a keystroke classifier
Feedback based Training

- A keystroke classifier (for inferring random text)
  - Given a keystroke, produce the label of the key

- Training
  - Input: noisy training data
    - Only a subset of labeled data from the language models
    - Choose those with fewer corrections by the language model (quality indicator)
  - Output: a not so accurate keystroke classifier

- Testing
  - Use the trained classifier to classify the training data again
  - Use the language model to correct the classification result
  - Use the corrected label for re-training
Feedback based Training (Con’t)

Not 100% accurately labeled

Training audio → Standard Training → Classifier

Training

Not so accurate

Classifier → Old training audio → text

Testing

More accurate Labels

Language correction
## Evaluation

<table>
<thead>
<tr>
<th></th>
<th>Set 1</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>words</td>
<td>chars</td>
<td>words</td>
<td>chars</td>
<td>words</td>
</tr>
<tr>
<td>unsupervised learning</td>
<td>34.72</td>
<td>76.17</td>
<td>38.50</td>
<td>79.60</td>
<td>31.61</td>
</tr>
<tr>
<td></td>
<td>74.57</td>
<td>87.19</td>
<td>71.30</td>
<td>87.05</td>
<td>56.57</td>
</tr>
<tr>
<td>1st supervised feedback</td>
<td>58.19</td>
<td>89.02</td>
<td>58.20</td>
<td>89.86</td>
<td>51.53</td>
</tr>
<tr>
<td></td>
<td>89.73</td>
<td>95.94</td>
<td>88.10</td>
<td>95.64</td>
<td>78.75</td>
</tr>
<tr>
<td>2nd supervised feedback</td>
<td>65.28</td>
<td>91.81</td>
<td>62.80</td>
<td>91.07</td>
<td>61.75</td>
</tr>
<tr>
<td></td>
<td>90.95</td>
<td>96.46</td>
<td>88.70</td>
<td>95.93</td>
<td>82.74</td>
</tr>
<tr>
<td>3rd supervised feedback</td>
<td>66.01</td>
<td>92.04</td>
<td>62.70</td>
<td>91.20</td>
<td>63.35</td>
</tr>
<tr>
<td></td>
<td>90.46</td>
<td>96.34</td>
<td>89.30</td>
<td>96.09</td>
<td>83.13</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>23.22</td>
<td>67.67</td>
<td>37.84</td>
<td>82.02</td>
<td>45.36</td>
</tr>
<tr>
<td></td>
<td>73.22</td>
<td>88.60</td>
<td>78.42</td>
<td>91.49</td>
<td></td>
</tr>
<tr>
<td></td>
<td>48.22</td>
<td>86.58</td>
<td>79.51</td>
<td>92.49</td>
<td></td>
</tr>
</tbody>
</table>

**Table 2:** Text recovery rate at each step. All numbers are percentages.
Evaluation

Audio Keystroke Detection

- Spectrum feature extraction
- Clustering

Group keystrokes into classes

Language models

Match classes with keys

Supervised learning to train a keystroke classifier

Sub-set of the output

1st 58%
2nd 65%
3rd 66%

Language model only
Keystroke: 37%
Language: 74%

1st 89%
2nd 91%
3rd 90%
Other Key Results

• Works for random text
  – Inferring passwords that contain English letters only
  – 90% of 5-character random passwords: < 20 attempts
  – 80% of 10-character random passwords: <75 attempts

• Works for multiple types of keyboards

• Even “low-quality” microphones can do the job
Possible Defenses

• Introduce noise into the system
  – Add (random) background noise to keystrokes
    o Remove the unique pattern for each key
  – Use quieter keyboards

• Other defenses
  – Two factor authentication (not just typing a password)
  – No microphone in your room?
Microarchitectural covert and side channels (how to share a secret)

Credit: Chris Fletcher (UIUC)
Process isolation + OS (CS 233)

... OS paging ...

Virtual memory

0x00000000

Process Memory

Communication to other processes via e.g., #include <sockets.h>, send(), recv()

...OS services...

0xffffffff

Threading, etc
Programs run on processors

- Processor that OS would have you see...

- Real processors (CS 433)

  - Core
  - Datapath
  - L1 I Cache
  - L1 D Cache
  - L2 Cache
  - L3 Cache
  - DRAM (and/or: stacked DRAM, HMC, NVMs)

OS swaps work on/off

Cache = on-chip memory, faster to access than DRAM

Multi-core

Multi-socket

w/ virtualization
**Programs run on processors**

- Processor that OS would have you see...

- Real processors (CS 433)

**Goal:** create a `send()`, `recv()` abstraction using Hardware contention (without using the OS/other sanctioned interfaces)
Covert Channels 101: Through the cache

- Cache fill for line A may cause another line B to be evicted
- Various mechanisms for owner of B to detect a hit or miss
- We like the cache: easy to measure, many types of sharing

**L1/L2 → Intra-core, inter-thread channels**

**LLC → inter-core channels**

**Directory → inter-core/inter-socket channels**

**DRAM row buffer → """"**
Processor caches

• Motivation
  – Programs have locality
  – Memory access cost \( \propto \) memory size
• Block placement/replacement policies tell us where blocks can live and when

Core-facing API:
  - Read(addr)
  - Write(addr, word)

Backend API:
  - Evict(addr)
  - Fill(addr, line)
Why is cache design relevant?

- Two processes can agree on “dead drops” on the processor hardware, to pass information under the OS’s nose.

```
Repeatedly accesses lines in set i

if (t2 \( - \) t1 > THRESH) read ‘1’
else read ‘0’
```
Normal communication

```c
#include <socket.h>

void send(bit msg) {
    socket.send(msg);
}

bit recv() {
    return socket.recv(msg);
}
```

Covert Channel communication

```c
void send(bit msg) {
    // pressure on cache
}

bit recv() {
    st = time();
    // pressure on cache
    return time() - st > THRESH;
}
```
We made a virtual “wire”, now what?

• Remember TCP?

• Virtual wire + de-noising + re-transmission + wrapper API = Cache pressure!
Fun! How else can I do this?

Sender

if (send ‘1’)
Use resource
else idle

Hardware resource

t1 = rdtsc()
Use resource
t2 = rdtsc()

Receiver

if (t2 − t1 > THRESH) read ‘1’
else read ‘0’
Many potential channels at our disposal:

- L1 I Cache
- L1 D Cache
- L2 Cache
- L3 Cache
- DRAM (and/or: stacked DRAM, HMC, NVMs)

Advanced techniques:

- Speculative execution [Spectre’18]
- Arithmetic timing [AKMJLS’15]
- Port contention [CBHGT’18]
- 4K aliasing [MES’17]
- Cache banking [YGH’16]
- Inclusive LLC [LYGHL’15]
- Non-inclusive LLC [YSGFCT’19]
- L3 Cache
- RAND unit [EP’16]
- DRAM [PGMSM’16]
- Multi-core
- Multi-socket w/ virtualization
Bandwidth

Error-free bitrate of send() → recv()

send(msg)  Channel  recv()

Depends on what hardware structure is used to build the channel.

- RDRAND unit: 7-200 Kbps [EP’16]
- Ld/st performance counters: ~75-150 Kbps [HKRVD'T'15]
- LLC: 1.2 Mbps [MNHF’15]
- Various structures on GPGPU: up to 4 Mbps [NKG’17]
Practical uses

• Talk to your friends for fun
• Malware can inter-communicate w/o OS realizing it
• Different VMs sharing the same box on (e.g.) Amazon AWS can talk

• Side channel attacks
  – Learn private information about co-resident processes
From covert → side channels

Covert channel:
- if (send ‘1’)
  - Use resource
- else
  - idle

Side channel:
- if (secret)
  - Use resource
- else
  - idle

Victim

Hardware resource

Attacker

$t_1 = \text{rdtsc}()$
- Use resource
- $t_2 = \text{rdtsc}()$

if ($t_2 - t_1 > \text{THRESH}$) read ‘1’
- else read ‘0’
Side channel attacks

- Shared resource pressure can also lead to side channel attacks
- E.g., RSA encryption \( \text{msg} = \text{Decrypt}_{\text{key}}(\text{Encrypt}_{\text{key}}(\text{msg})) \)

```plaintext
SquareMult(x, e, N):
    let \( e_n, \ldots, e_1 \) be the bits of \( e \)
    \( y \leftarrow 1 \)
    for \( i = n \) down to 1 {
        \( y \leftarrow \text{Square}(y) \) (S)
        \( y \leftarrow \text{ModReduce}(y, N) \) (R)
        if \( e_i = 1 \) then {
            \( y \leftarrow \text{Mult}(y, x) \) (M)
            \( y \leftarrow \text{ModReduce}(y, N) \) (R)
        }
    }
    return \( y \)
```
Discussion

• Any other examples of side channels you can think of to infer user information / steal data?

• What’s your thoughts on the future development of microarchitecture side channels (try to also think from the defender’s side of view)?