Covert and Side Channels

Mengjia Yan
Spring 2022

Based on slides from Christopher W. Fletcher
Before We Start

• Recitation Prize

• HotCRP Demo
  • Review submission interface
  • Bid papers

• Announce 3 Talks
What is Covert and Side Channel?

• Gather information by measuring or exploiting indirect effects of the system or its hardware -- rather than targeting the program or its code directly.

• Covert channel:
  • Intended communication between two or more security parties

• Side channel:
  • Unintended communication between two or more security parties

• In both cases:
  • Communication should not be possible, following system semantics
  • The communication medium is not designed to be a communication channel
Side Channels Are Almost Everywhere
Daily Life Examples

• Acoustic side channels
  • Monitor keystrokes
  • You only need: a cheap microphone + an ML model

• Network traffic contention side channel
  • If you want to be an active attacker, try stress test
“Hear” The Screen

Sound Spectogram

Genkin et. al. Synesthesia: Detecting Screen Content via Remote Acoustic Side Channels. S&P’19
“Hear” The Screen

(A) is the LCD panel, (B) is the screen’s digital logic and image rendering board and, (C) is the screen’s power supply board.
Network Side Channels

• Website Fingerprinting

• Response dependent:
  • iSideWith.com

• Real-time feedback:
  • Google Search auto-complete

Lescisin et. al. Tools for Active and Passive Network Side-Channel Detection for Web Applications. WOOT’18
Cai et. al. Touching from a distance: Website fingerprinting attacks and defenses. CCS’12.
Physical v.s. Timing v.s. uArch Channel

• What can the adversary observe?

Attacker requires measurement equipment → physical access
Power Analysis

Cryptographic device
(e.g., smart card and reader)

Control, Cyphertexts

Oscilloscope

Control, Waveform data

Computer

from https://en.wikipedia.org/wiki/Power_analysis
Victim Application: RSA

• Square-and-multiply based exponentiation

Input: base $b$, modulo $m$, exponent $e = (e_{n-1} ... e_0)_2$
Output: $b^e \mod m$

$r = 1$
for $i = n-1$ down to 0 do
  $r = \text{sqr}(r)$
  $r = \text{mod}(r, m)$
  if $e_i == 1$ then
    $r = \text{mul}(r, b)$
    $r = \text{mod}(r, m)$
  end
end
return $r$
Power Analysis

- Various signal processing techniques to de-noise.
- More advanced: differential power analysis (DPA)
Benign Usage: Non-intrusive Software Monitoring

• How to efficiently monitor application for anomaly detection?
• EM side channel can trace back to Van Eck phreaking in 1985

Sehatbakhsh et al. Spectral Profiling: Observer-Effect-Free Profiling by Monitoring EM Emanations. MICRO’16
Van Eck phreaking https://en.wikipedia.org/wiki/Van_Eck_phreaking
What can you do with these channels?

• Violate privilege boundaries
  • Inter-process communication
  • Infer an application’s secret

• (Semi-Invasive) application profiling

• What makes it more threatening compared to traditional software or physical attacks?
  • Stealthy. Sophisticated mechanisms needed to detect channel
  • Usually no permanent indication one has been exploited
Physical v.s. Timing v.s. uArch Channel

• What can the adversary observe?

**Physical channels**
- Power, EM, sound, etc.
- Attacker requires measurement equipment → physical access

**Timing channels**
- Response time
- Attacker may be remote (e.g., over an internet connection)
Victim Application: AES

- **SubBytes:**
  \[ S[i] = Ttable[S[i]] \]
Physical v.s. Timing v.s. uArch Channel

- What can the adversary observe?

**Physical channels**
- Power, EM, sound, etc.
- Attacker requires measurement equipment → physical access

**Timing channels**
- Response time
- Attacker may be remote (e.g., over an internet connection)

**Microarchitectural channels**
- Microarch events (e.g., timing, perf. counters, etc.)
- Attacker may be remote, or be co-located
uArch Side Channels
Recap: Process Isolation

Virtual Address Space (Programmer's View)

Physical Address Space (limited by DRAM size)

Page Table per process

How to communicate across processes?

Process 1

Process 2

4KB

4KB

4KB

4KB

6.888 - L3 Covert and Side Channels
Inter-process communication

• File
• Socket
• Pipe
• Shared memory (shm in Linux)
• ...

All of these communication approaches are monitored by OS.
Normal Cross-process Communication

```c
#include <socket.h>

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

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

How to communication without letting OS know?

--> Use HW contention
Covert Channels 101: Through the Page Fault

Blackboard: page fault, on-demand paging
Covert Channels 101: Through the Page Fault

if (send ‘1’)
    accesses many pages
else
    idle

\[ t_1 = \text{rdtsc}() \]
Accesses many pages
\[ t_2 = \text{rdtsc}() \]

if (t_2 - t_1 > \text{THRESH})
    read ‘1’
else
    read ‘0’
Another Example of Using Caches

Process 1 (Sender)

if (send ‘1’)
   *Fill up the cache*
else
   *idle*

Process 2 (Receiver)

t1 = rdtsc()
*Fill up the cache*
t2 = rdtsc()

if (t2 – t1 > THRESH)
   read ‘1’
else
   read ‘0’
Potential Covert Channel Medium?

- Functional units inside the pipeline/core
- Main memory
- Network interface card (NIC)
- Hard disk drive
- GPUs
- PCIe bus
The Memory Hierarchy

- L1, L2
  - Shared by threads on the same core
- LLC:
  - Shared by threads on different cores
- DRAM row buffer:
  - Shared by .....
Flush+Reload in the Cache

• On blackboard: page deduplication, clflush
Protocol 101: Prime+Probe in the Cache

Sender

Receiver

Cache Set

# ways

Shared Cache

6.888 - L3 Covert and Side Channels
Prime+Probe

Sender

Receiver

# ways

Cache Set

Shared Cache

Sender line

Receiver line

Time

Prime

Cache Set

6.888 - L3 Covert and Side Channels
Prime+Probe – Send “1”
Prime+Probe – Receive “1”

Receive “1” = 8 accesses → 1 miss
Prime+Probe – Send “0”

- Sender
- Receiver
- Cache Set
- Shared Cache
- # ways
- Time
- Prime
- Wait
- NO Access
- Sender line
- Receiver line

Access Wait Time

Cache Set
Prime+Probe – Receive “0”

Receive “0” = 8 accesses → 0 miss
A Complete Protocol -- Synchronization

Sample window length

Sender

Prime
Wait
Probe

Prime
Wait
Probe

Receiver

Receiver

- Window size agreed on by sender and receiver
- Each window transmits some bits

Sender & receiver need to perform an window alignment at the start

Question: how to distinguish between noise and actual transmission?
Bandwidth

Error-free bitrate of \texttt{send()} \rightarrow \texttt{recv()}

\[
\text{send}(\text{msg}) \quad \text{Channel} \quad \text{recv()}
\]

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

- RDRAND unit:
- MemBus/AES-NI contention:
- LLC:
- Various structures on GPGPU:

6.888 - L3 Covert and Side Channels
From Covert $\rightarrow$ Side Channels

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

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

Side channel:
- $t1 = \text{rdtsc}()$
- Use resource
- $t2 = \text{rdtsc}()$

Side channel:
- if (t2 – t1 > THRESH)
- read ‘1’
- else
- read ‘0’

6.888 - L3 Covert and Side Channels
Summary

• What can the adversary observe?

Physical channels
- Power, EM, sound, etc.
- Attacker requires measurement equipment → physical access

Timing channels
- Response time
- Attacker may be remote (e.g., over an internet connection)

Microarchitectural channels
- Microarch events (e.g., timing, perf. counters, etc.)
- Attacker may be remote, or be co-located
Micro-arch Side Channel Generalization

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’
Next Lecture:
Practical Cache Side Channel Examples