Abusing hardware for fun and profit
Agenda

- Cache-based Covert channels w/ demo
- Spectre and Meltdown from covert channels
Process isolation + OS (CS 423)

...OS paging...

0x00000000

Virtual memory

0xffffffff

Process Memory

...OS services...

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

Processor that OS would have you see ...

• Real processors

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

OS swaps work on/off

Core

Memory

Core

Core

Core

Core

Datapath

L1 I Cache

L1 D Cache

L2 Cache

L3 Cache

DRAM (and/or: stacked DRAM, HMC, NVMs)
Hardware Covert Channels

Talk to your friends without the OS’s help or knowledge
No header files → no socket/etc, no OS-sanctioned communication

Exploit properties of your hardware 😊

L3 cache shared by all processes running on system!
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)

Which set?
2 or more \(L_{i+1}\) cache sets statically map to

Which way?
Determined by replacement policy.
Why is cache design relevant?

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

\[ t_1 = \text{rdtsc()} \]
Repeatedly accesses lines in set i

\[ t_2 = \text{rdtsc()} \]
If \((t_2 - t_1 > \text{thresh})\) read ‘1’
Else read ‘0’
We made a virtual “wire”, now what?

Remember TCP?

Virtual wire +
de-noising +
re-transmission +
wrapper API

= Cache pressure!
Demo
Fun! How else can I do this?

Processes share...
branch predictors, cores, caches, RNG modules, DRAM, ...

All of which can (and have) been turned into “virtual wires”
And they are pretty fast (~ 1 Mb/sec on the high end)
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
Side channel attacks

Shared resource pressure can also lead to side channel attacks.

E.g., RSA encryption

\[ \text{msg} = \text{Decrypt}_{\text{key}}(\text{Encrypt}_{\text{key}}(\text{msg})) \]
Ingredients

Cover channel
Speculation
OS mapped to process address space (for Meltdown)
Branch prediction (for Spectre)
Out of order, speculative processor core

```assembly
xor sum, 0, 0
xor d, 0, 0
loop:
  add $t0, d, &P1
  lw P1d, 0($t0)
  add $t0, d, &P2
  lw P2d, 0($t0)
  sub $t0, P1d, P2d
  mul $t0, $t0, $t0
  add sum, sum, $t0
  addi d, d, 1
  ble loop, d, LEN
post:
  blt end, best, sum
  add best, sum, 0
end:
```