Deepen the Defenses: A Case for Microarchitectural Isolation

Jo Van Bulck

Cybersec Europe, FutureLab Stage, Brussels, May 11, 2022

imec-DistriNet, KU Leuven, Belgium jo.vanbulck@cs.kuleuven.be jovanbulck

DistriNet
KU LEUVEN

- Postdoctoral **researcher** @imec-DistriNet, KU Leuven, Belgium
  - → PhD "Microarchitectural Side-Channel Attacks for Privileged Software Adversaries"
- **Trust across the system stack:** App > compiler > OS > CPU > μ-arch

Side-channel analysis

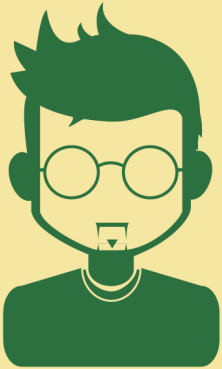
Transient-execution attacks
(Intel x86 SGX)

Embedded trust
(TI MSP430)

Hardware (noun.) — *The part of a computer that you can kick.*

Software (noun.) — *The reason you want to kick the hardware.*

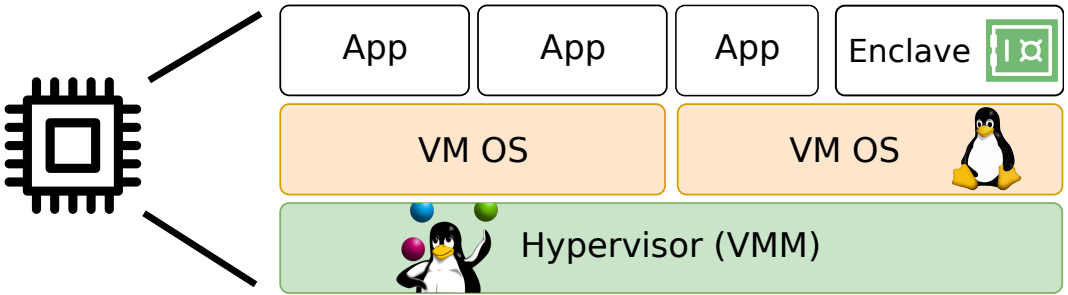
Software Engineer vs Hardware Engineer



Job Title	Software engineer	Hardware engineer
Job Description	Develop, design and test software or construct, maintain computer networks and programs	Research, develop and test hardware or computer equipment
Education	Software Engineering or Computer Science Degree	Electrical & Computer Engineering Degree
Skill Sets	Technology Design, Complex Problem Solving, Critical Thinking, etc.	Troubleshooting, Problem Solving, Systems Evaluation, etc.
Salary	$107,840	$112,760
Number of Jobs	>1,128,000	>87,000

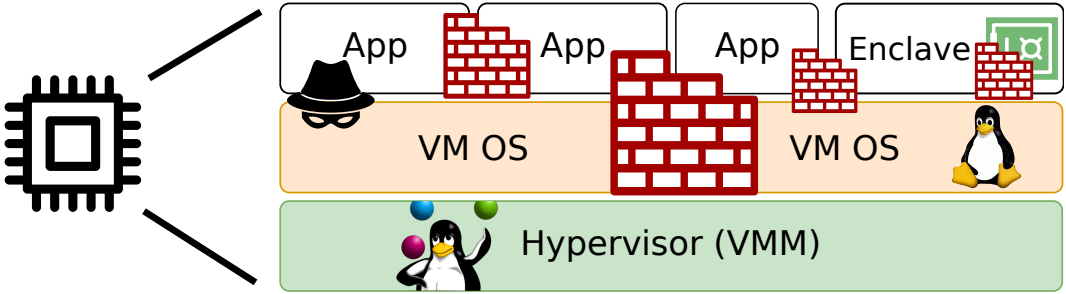
ComputerCareers.org

Processor security: Hardware isolation mechanisms



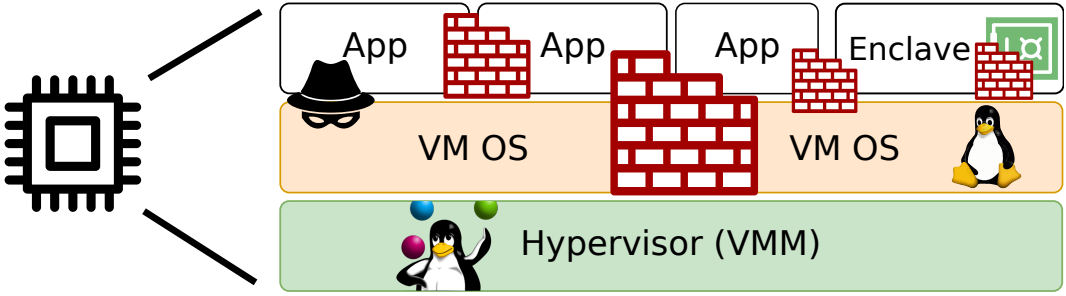
- Different software **protection domains:** Processes, VMs, enclaves

Processor security: Hardware isolation mechanisms



- Different software **protection domains:** Processes, VMs, enclaves
- CPU builds "walls" for **memory isolation** between apps and privilege levels

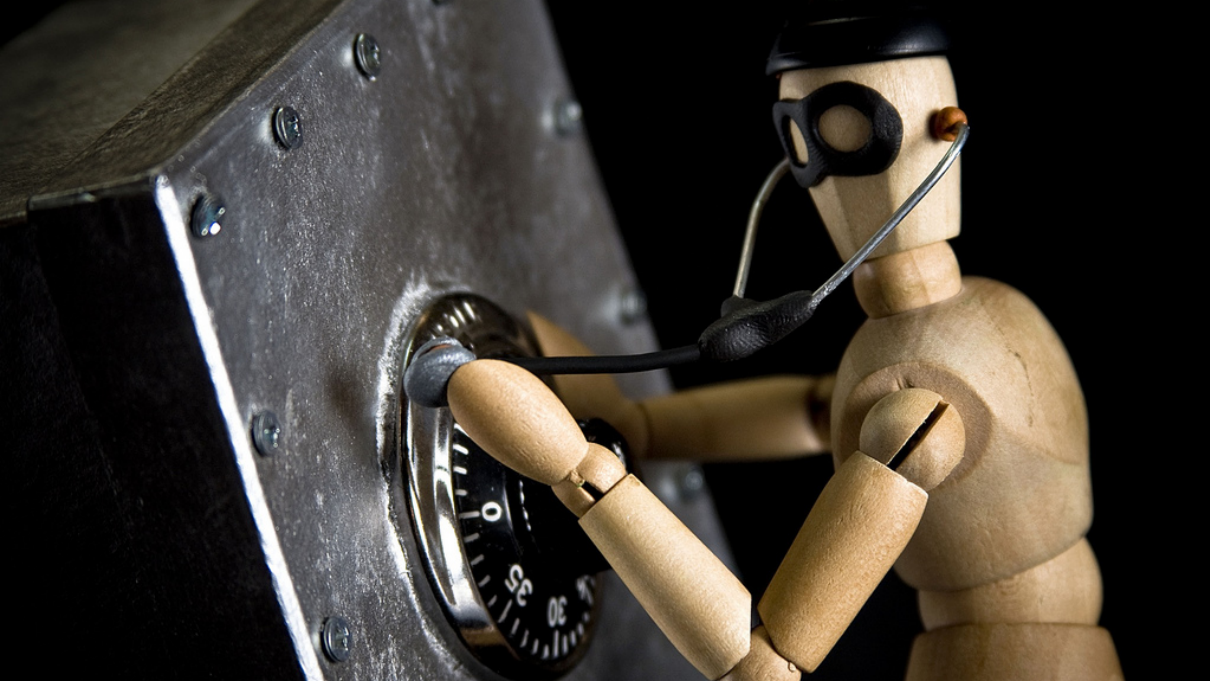
Processor security: Hardware isolation mechanisms



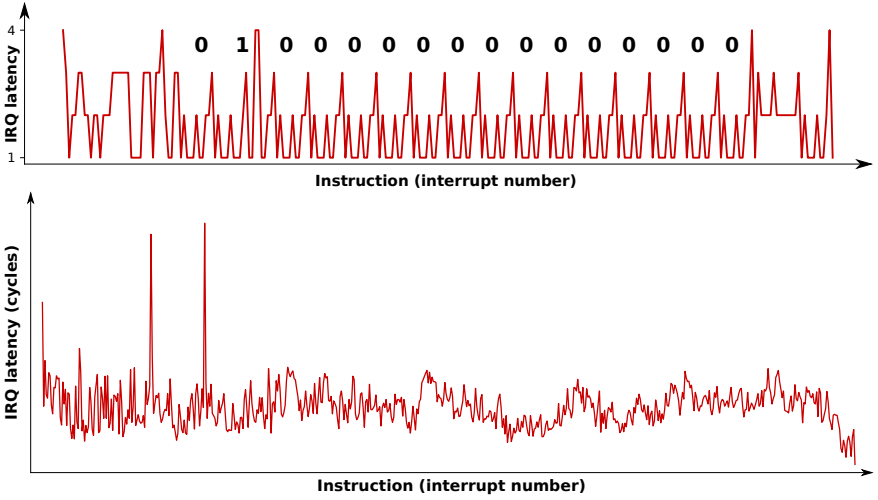
- Different software **protection domains:** Processes, VMs, enclaves
- CPU builds "walls" for **memory isolation** between apps and privilege levels

↔ Architectural protection walls permeate **microarchitectural side channels!**


VAULT DOOR
WEIGHT 22 1/2 Tons
THICKNESS 22 Inches
STEEL 11 Layers of Special Cutting and Drill Resistant
LOCKS 4 Hamilton Watch Movements for Time Locks



Microarchitectural timing leaks in practice

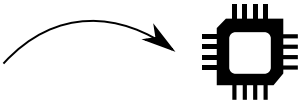


Van Bulck et al. "Nemesis: Studying Microarchitectural Timing Leaks in Rudimentary CPU Interrupt Logic", CCS 2018

Example: CPU cache timing side channel

Cache principle: CPU speed ≫ DRAM → *cache code/data*

```
while true do
    maccess(&a);
endwh
```



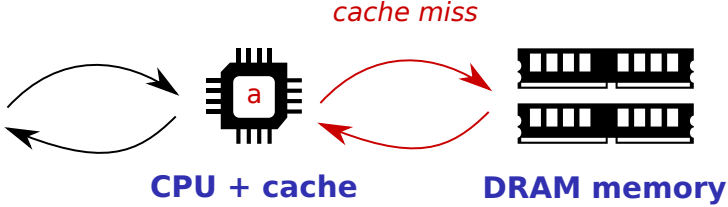
CPU + cache

DRAM memory

Example: CPU cache timing side channel

Cache miss: Request data from (slow) DRAM upon first use

```
while true do
    maccess(&a);
endwh
```

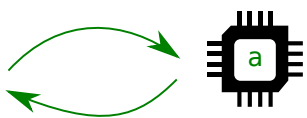


Example: CPU cache timing side channel

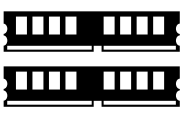
Cache hit: No DRAM access required for subsequent uses

```
while true do
    maccess(&a);
endwh
```

cache hit

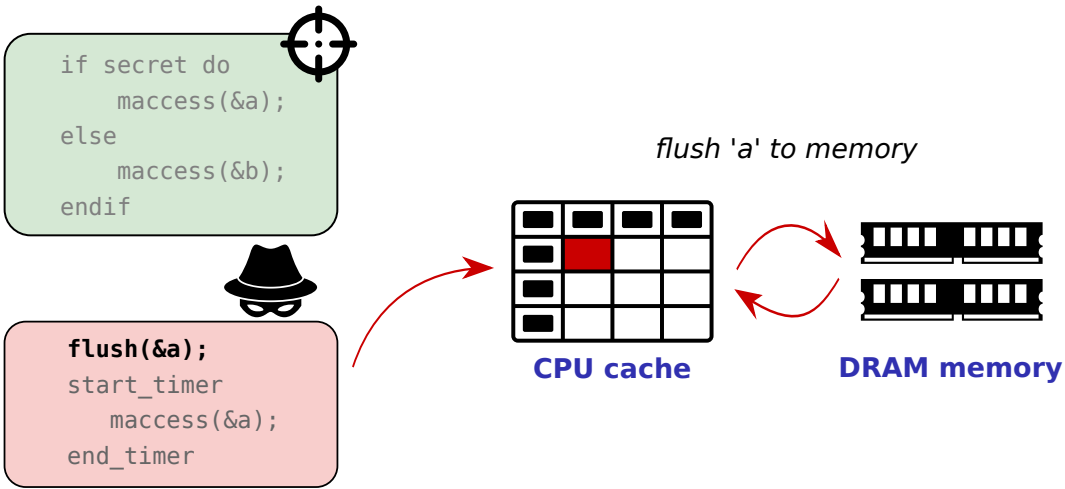


CPU + cache

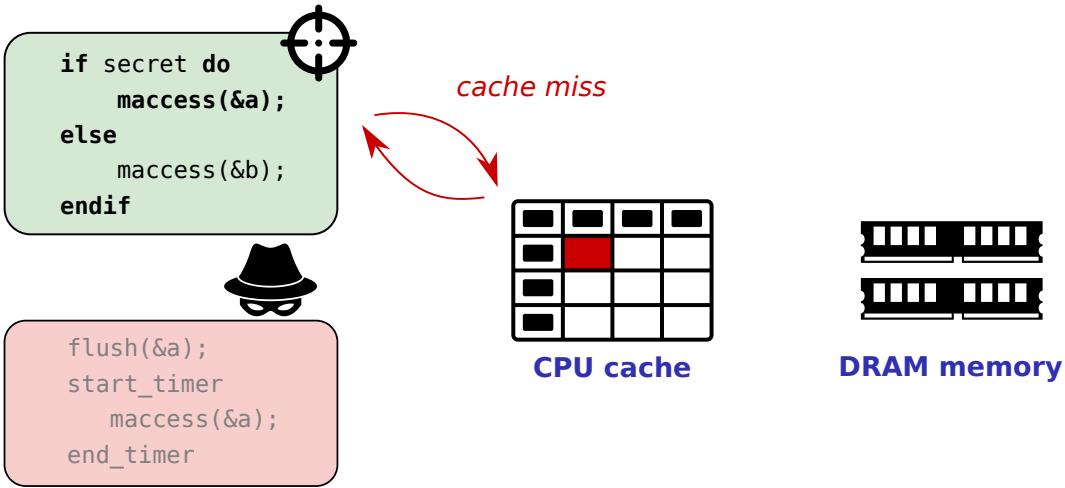


DRAM memory

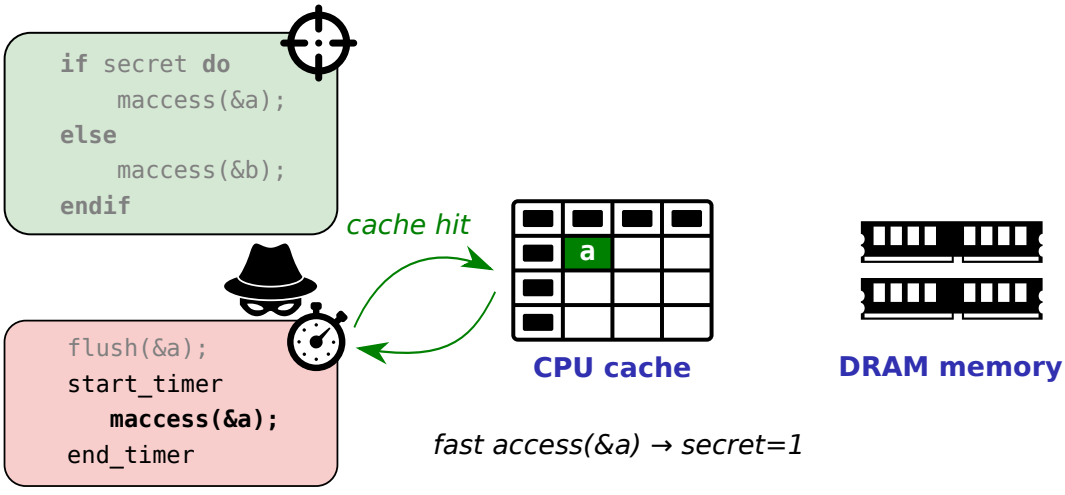
Cache timing attacks in practice: Flush+Reload



Cache timing attacks in practice: Flush+Reload



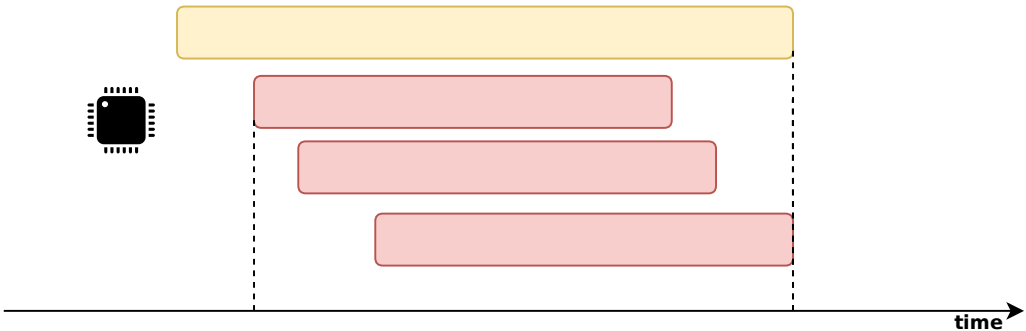
Cache timing attacks in practice: Flush+Reload



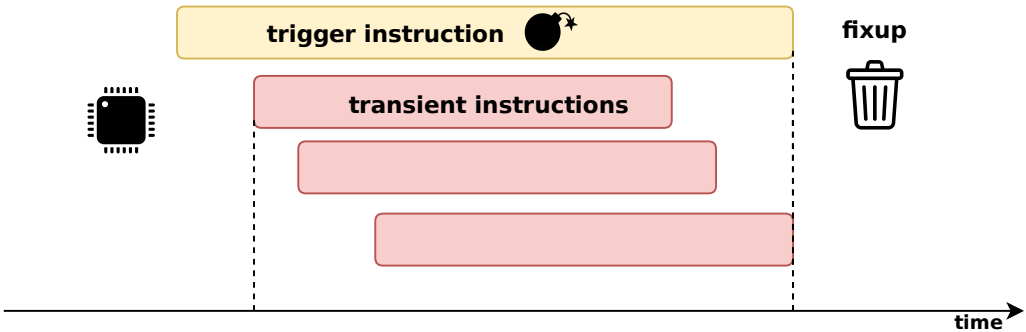

We can communicate across protection walls
using microarchitectural side channels!


WHAT IF I TOLD YOU
YOU CAN CHANGE RULES MID-GAME

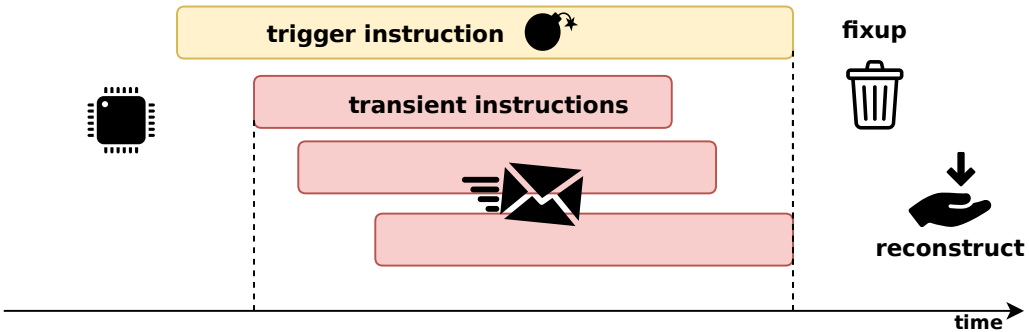
Abusing out-of-order and speculative execution



Abusing out-of-order and speculative execution

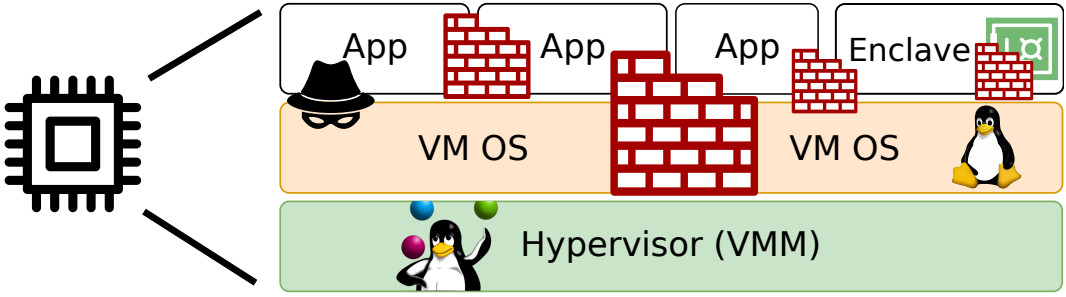


Abusing out-of-order and speculative execution

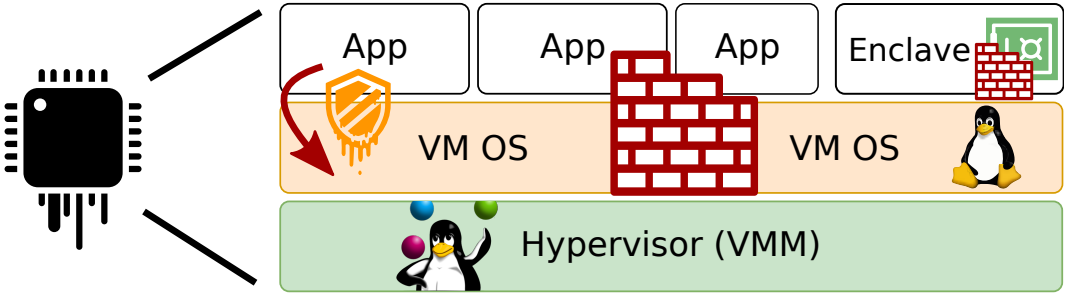


Transient-execution attacks: Welcome to the world of fun!

Leaky processors: Breaking isolation mechanisms

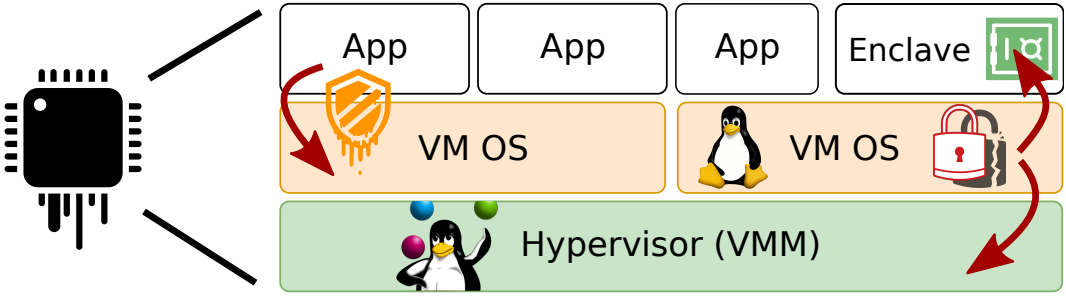


Leaky processors: Breaking isolation mechanisms



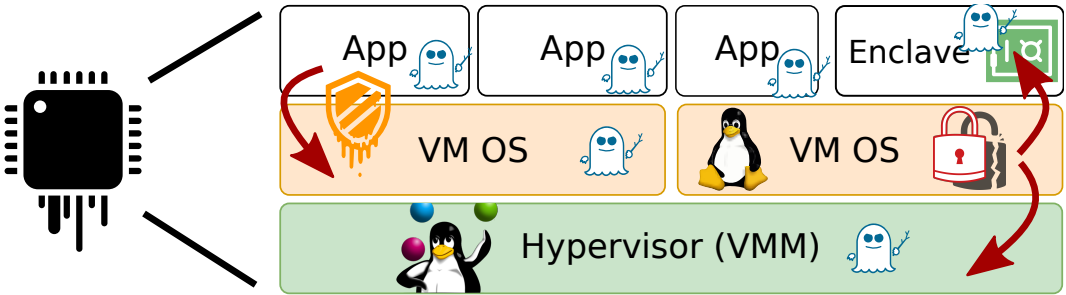
- Meltdown breaks user/kernel isolation

Leaky processors: Breaking isolation mechanisms



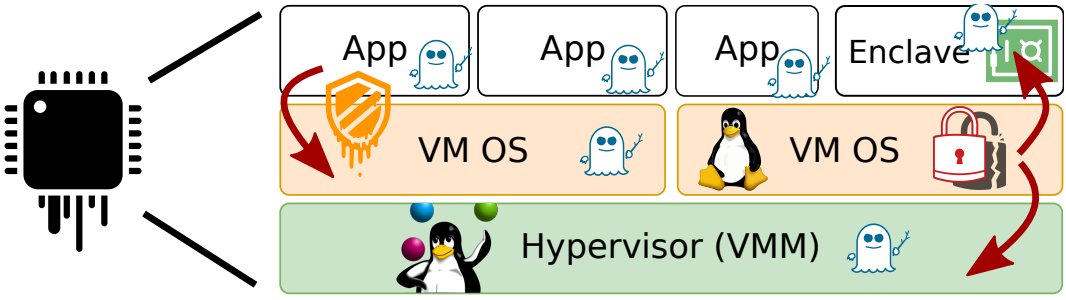
- Meltdown breaks user/kernel isolation
- Foreshadow breaks SGX enclave and virtual machine isolation

Leaky processors: Breaking isolation mechanisms



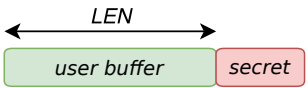
- Meltdown breaks user/kernel isolation
- Foreshadow breaks SGX enclave and virtual machine isolation
- Spectre breaks software-defined isolation on various levels

Leaky processors: Breaking isolation mechanisms



- Meltdown breaks user/kernel isolation
- Foreshadow breaks SGX enclave and virtual machine isolation
- Spectre breaks software-defined isolation on various levels
- . . . many more – but all exploit the same underlying insights!

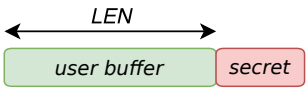
Spectre v1: Speculative buffer over-read




```
if (idx < LEN)
{
  s = buffer[idx];
  t = lookup[s];
  ...
}
```

- Programmer *intention:* no out-of-bounds accesses

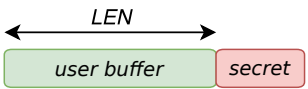
Spectre v1: Speculative buffer over-read




```
if (idx < LEN)
{
  s = buffer[idx];
  t = lookup[s];
  ...
}
```

- Programmer *intention:* no out-of-bounds accesses
- **Mistrain gadget** to speculatively "ahead of time" execute with $idx \geq LEN$ in the transient world

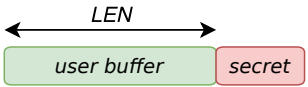
Spectre v1: Speculative buffer over-read




```
if (idx < LEN)
{
  s = buffer[idx];
  t = lookup[s];
  ...
}
```

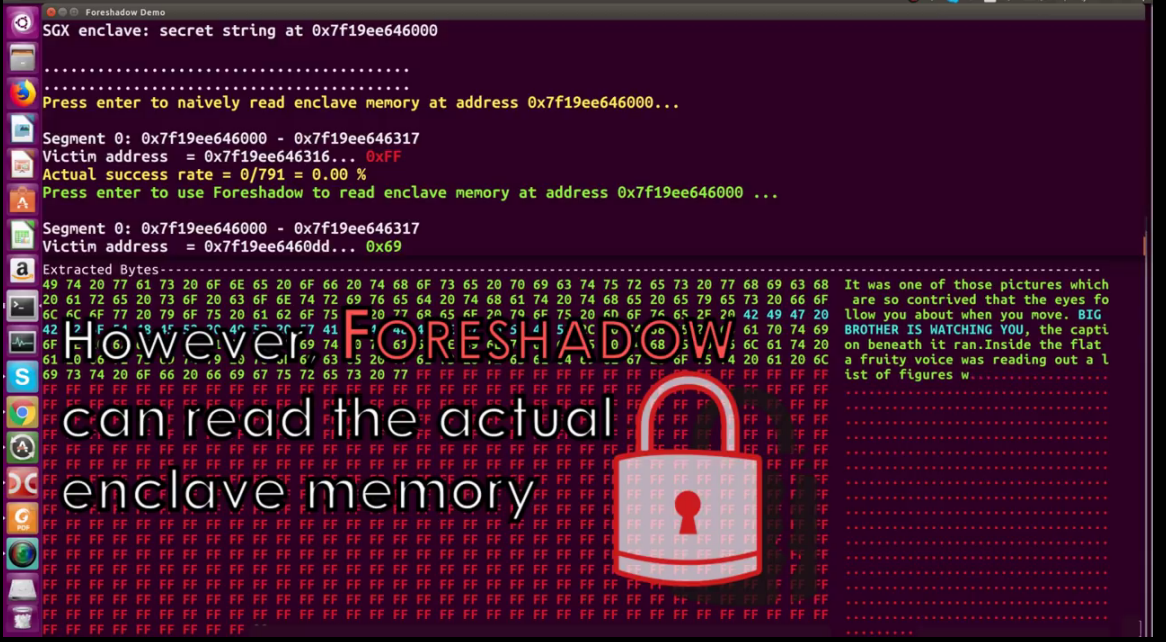
- Programmer *intention:* no out-of-bounds accesses
- **Mistrain gadget** to speculatively "ahead of time" execute with $idx \geq LEN$ in the transient world
- **Side channels** may leave traces after roll-back!

Spectre v1: Speculative buffer over-read




```
if (idx < LEN)
{
  asm("lfence\n\t");
  s = buffer[idx];
  t = lookup[s];
  ...
}
```

- Programmer *intention:* no out-of-bounds accesses
- **Mistrain gadget** to speculatively "ahead of time" execute with $idx \geq LEN$ in the transient world
- **Side channels** may leave traces after roll-back!
- Insert explicit **speculation barriers** to tell the CPU to halt the transient world...


Terminal
Foreshadow Demo
SGX enclave: secret string at 0x7f19ee646000
Press enter to naively read enclave memory at address 0x7f19ee646000...
Segment 0: 0x7f19ee646000 - 0x7f19ee646317
Victim address = 0x7f19ee646316... 0xFF
Actual success rate = 0/791 = 0.00 %
Press enter to use Foreshadow to read enclave memory at address 0x7f19ee646000 ...
Segment 0: 0x7f19ee646000 - 0x7f19ee646317
Victim address = 0x7f19ee6460dd... 0x69
Extracted Bytes
However FORESHADOW
can read the actual
enclave memory


SHARING IS NOT CARING
SHARING IS LOSING YOUR STUFF TO OTHERS
imgflip.com

A new golden age for computer architecture?

Conclusions and take-away

Hardware + software patches: Update your systems!

Conclusions and take-away

Hardware + software patches: Update your systems!

⇒ New emerging and powerful class of **transient-execution** attacks

⇒ Importance of fundamental **side-channel research;** no silver-bullet defenses

⇒ Security **cross-cuts** the system stack: hardware, OS, VMM, compiler, app