CPU Fuzzing for Discovering Hardware-caused Information Leakage

Michael Schwarz

January 2022

CISPA Helmholtz Center for Information Security

Who am I

Michael Schwarz

Faculty @ CISPA Helmholtz Center for Information Security

Research on CPU security and side channels

michael.schwarz@cispa.de

BBC

Intel Zombieload bug fix to slow data centre computers

THE VERGE

ZombieLoad attack lets hackers steal data from Intel chips

FORTUNE

'Zombieload' Flaw Lets Hackers Crack Almost Every Intel Chip Back to 2011. Why's It Being Downplayed?

How-To Geek

Only New CPUs Can Truly Fix ZombieLoad and Spectre


FIREEYE CYBER
THREAT MAP
DEVELOPING STORY
COMPUTER CHIP FLAWS IMPACT BILLIONS OF DEVICES
LIVE
CNN
DAX ▲164.69
NEWS STREAM

Introduction

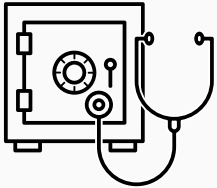
Microarchitectural side channels are powerful attack techniques

- Attack cryptographic implementations
- Spy on user behavior
- Augment traditional software exploits
- Building blocks for transient-execution attacks

Side-channel Attacks

$$M \equiv C^d \pmod{n}$$

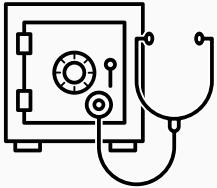
Description



Side-channel Attacks

$$M \equiv C^d \pmod{n}$$

Description

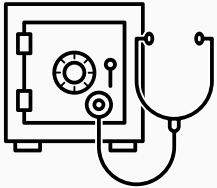


$$C^n = \begin{cases} (C^2)^{\frac{n}{2}} & \text{if } n \equiv 0 \mod 2 \\ C \cdot (C^2)^{\frac{n-1}{2}} & \text{if } n \equiv 1 \mod 2 \end{cases}$$

Software

Execution time

Side-channel Attacks



Description

$$M \equiv C^d \pmod{n}$$

Software

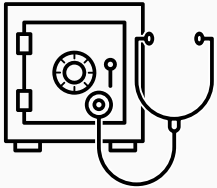
$$C^n = \begin{cases} (C^2)^{\frac{n}{2}} & \text{if } n \equiv 0 \mod 2 \\ C \cdot (C^2)^{\frac{n-1}{2}} & \text{if } n \equiv 1 \mod 2 \end{cases}$$

Hardware

Execution time

Power consumption

Side-channel Attacks



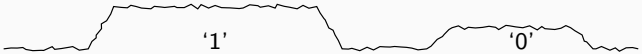
Description

$$M \equiv C^d \pmod{n}$$

Software

$$C^n = \begin{cases} (C^2)^{\frac{n}{2}} & \text{if } n \equiv 0 \mod 2 \\ C \cdot (C^2)^{\frac{n-1}{2}} & \text{if } n \equiv 1 \mod 2 \end{cases}$$

Hardware

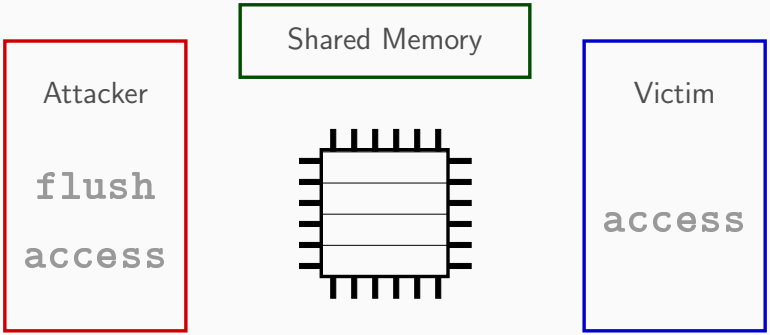


Execution time

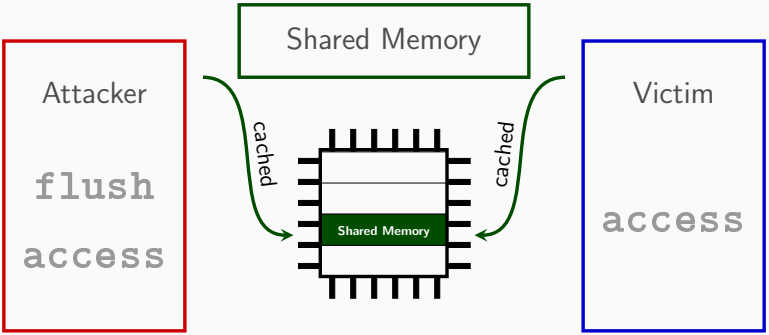
Power consumption

CPU caches

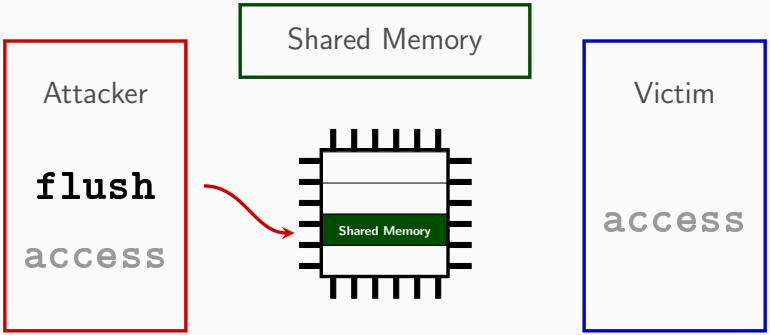
Flush+Reload



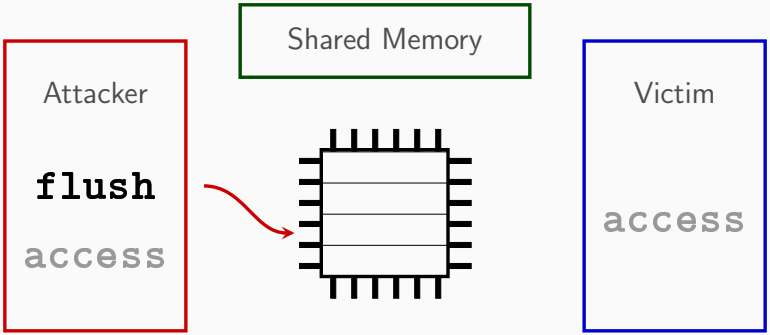
Flush+Reload



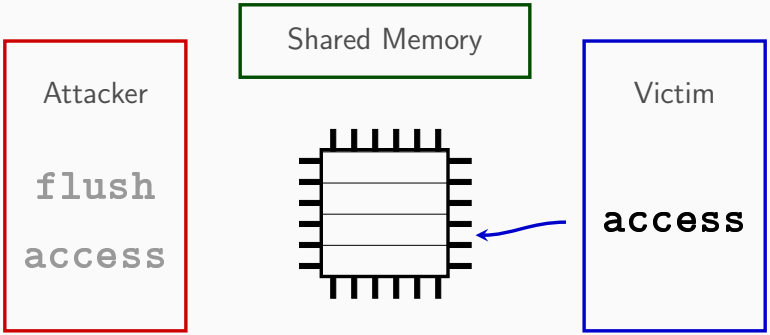
Flush+Reload



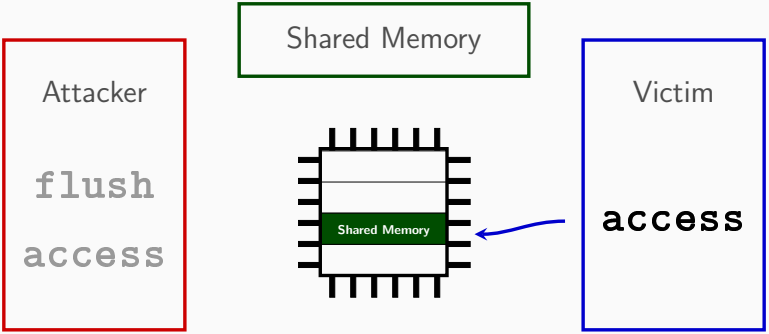
Flush+Reload



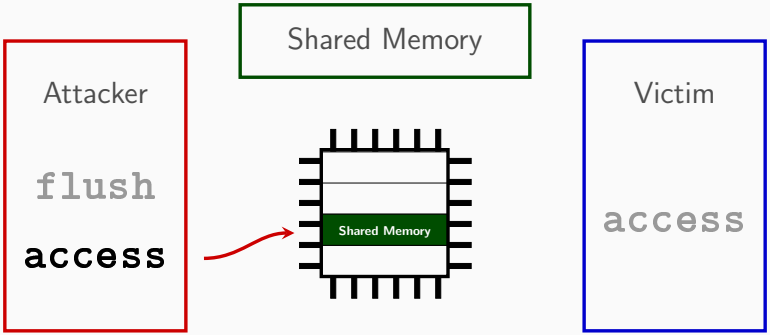
Flush+Reload



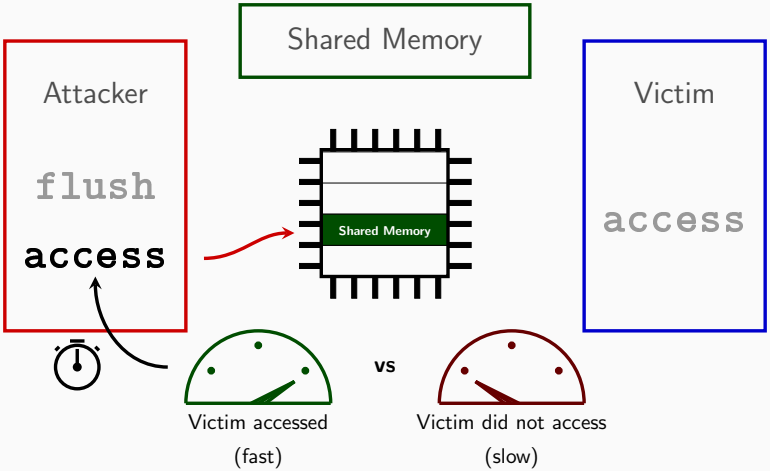
Flush+Reload



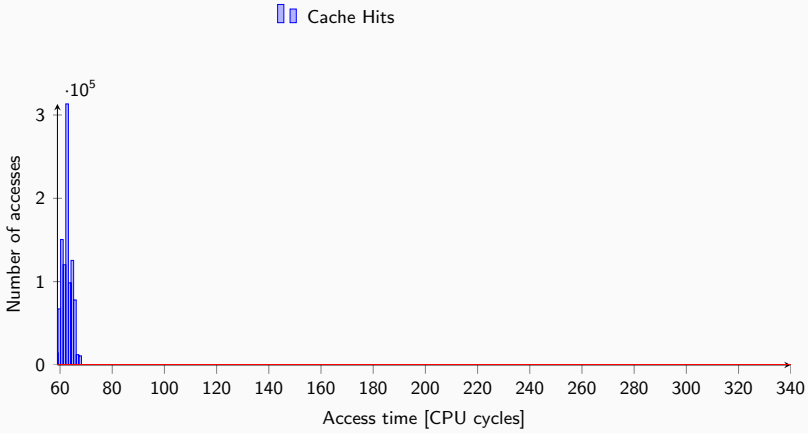
Flush+Reload



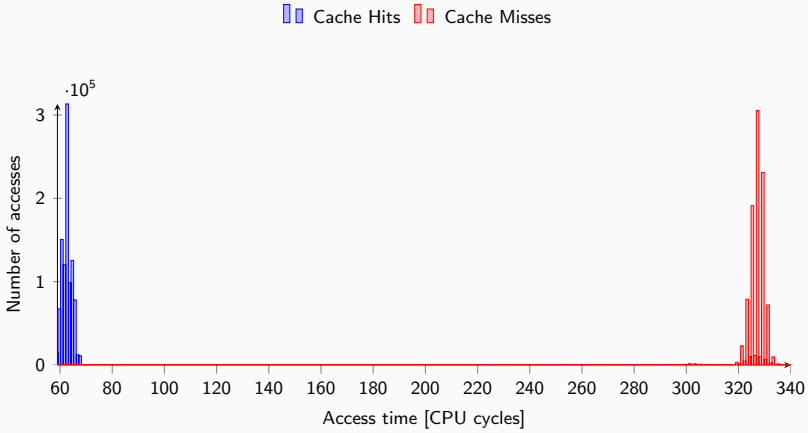
Flush+Reload



Flush+Reload



Flush+Reload



Flush+Reload on Square-and-Multiply

$$M = C^{d} \bmod n$$

Flush+Reload on Square-and-Multiply

$$M = C^{\boxed{d}} \bmod n$$

1	1	0	0	1	1	0	···

Result = C

Flush+Reload on Square-and-Multiply

$$M = C^{\boxed{d}} \bmod n$$

1	1	0	0	1	1	0	$\cdots$

$$\boxed{\text{Result}} = \underbrace{\boxed{\text{Result}} \times \boxed{\text{Result}}}_{\text{square}} \times \underbrace{\boxed{C}}_{\text{multiply}}$$

Flush+Reload on Square-and-Multiply

$$M = C^{\boxed{d}} \bmod n$$

1	1	0	0	1	1	0	$\cdots$

$$\boxed{\text{Result}} = \underbrace{\boxed{\text{Result}} \times \boxed{\text{Result}}}_{\text{square}}$$

Flush+Reload on Square-and-Multiply

$$M = C^{\boxed{d}} \bmod n$$

1	1	0	0	1	1	0	$\cdots$

$$\boxed{\text{Result}} = \underbrace{\boxed{\text{Result}} \times \boxed{\text{Result}}}_{\text{square}}$$

Flush+Reload on Square-and-Multiply

$$M = C^{\boxed{d}} \bmod n$$

1	1	0	0	1	1	0	$\cdots$

$$\boxed{\text{Result}} = \underbrace{\boxed{\text{Result}} \times \boxed{\text{Result}}}_{\text{square}} \times \underbrace{\boxed{C}}_{\text{multiply}}$$

Flush+Reload on Square-and-Multiply

$$M = C^{\boxed{d}} \bmod n$$

1	1	0	0	1	1	0	$\cdots$

$$\boxed{\text{Result}} = \underbrace{\boxed{\text{Result}} \times \boxed{\text{Result}}}_{\text{square}} \times \underbrace{\boxed{C}}_{\text{multiply}}$$

Flush+Reload on Square-and-Multiply

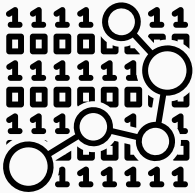
$$M = C^{\boxed{d}} \bmod n$$

1	1	0	0	1	1	0	···

$$\boxed{\text{Result}} = \underbrace{\boxed{\text{Result}} \times \boxed{\text{Result}}}_{\text{square}}$$

From Metadata to Data

Meta Data



Data

Transient Execution Attacks

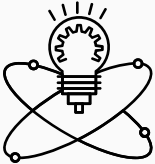
- Transient-execution attacks evolved from side-channel attacks
- Side channel is a building block
- Leak data, not only metadata
- Meltdown, Spectre, ZombieLoad, Foreshadow, Fallout, LVI, ...

Motivation

Problem

Finding side channels and vulnerabilities is a **complex** and **time-consuming** process

Motivation



- Apply bug-finding techniques from software

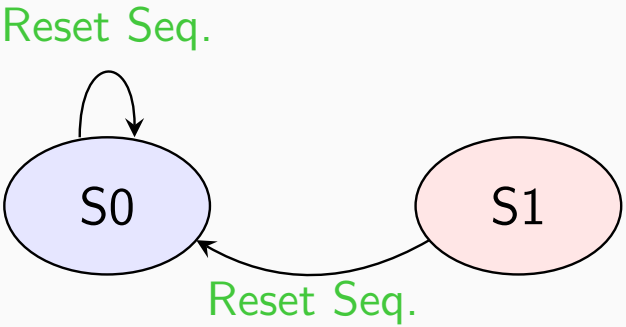
→ Fuzz for side channels

  - Input are code sequences
  - Detect timing differences

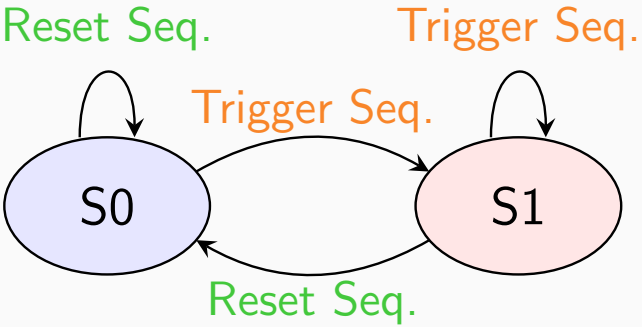
- Start with random (dumb) fuzzing

Sequence Triples

Sequence Triples

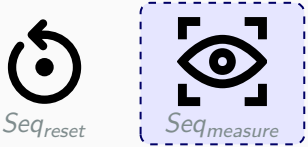


Sequence Triples



Testing A Sequence Triple

Testing A Sequence Triple



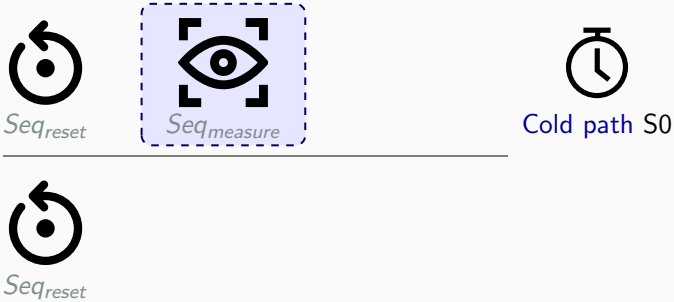
Testing A Sequence Triple

Seq_{reset}

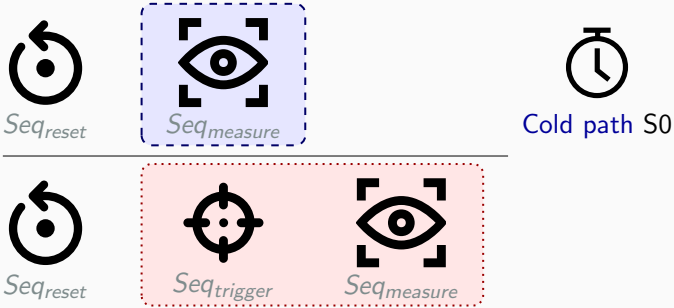
$Seq_{measure}$

Cold path S0

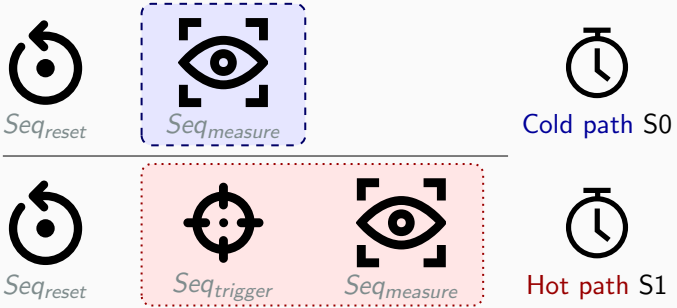
Testing A Sequence Triple



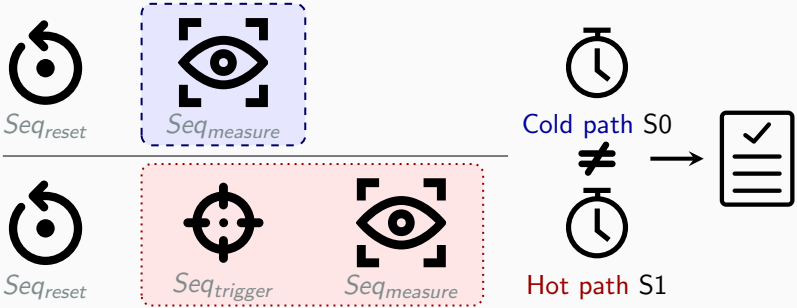
Testing A Sequence Triple



Testing A Sequence Triple

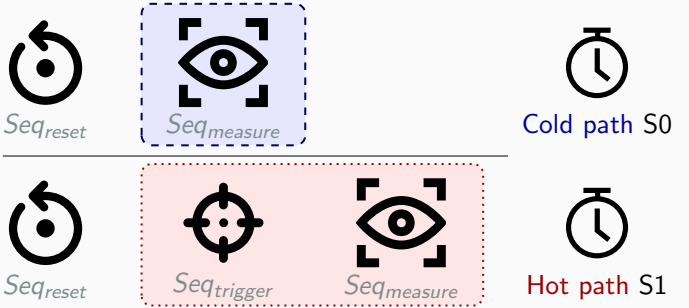


Testing A Sequence Triple



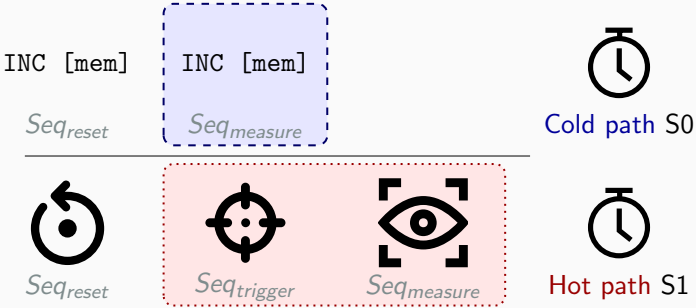
Testing A Sequence Triple

Example 1: $Seq_{measure} = Seq_{trigger} = Seq_{reset} =$ `INC [mem]`



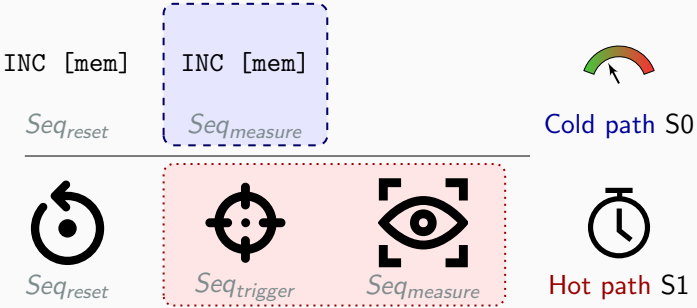
Testing A Sequence Triple

Example 1: $Seq_{measure} = Seq_{trigger} = Seq_{reset} = \texttt{INC [mem]}$



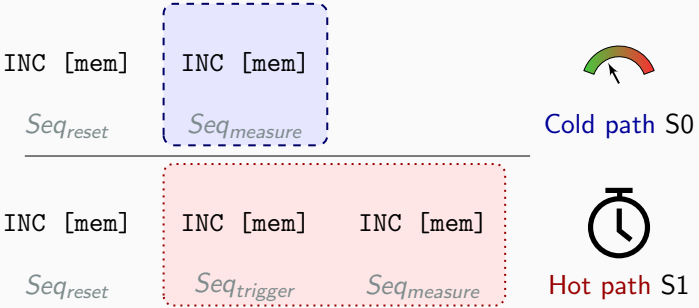
Testing A Sequence Triple

Example 1: $Seq_{measure} = Seq_{trigger} = Seq_{reset} = \texttt{INC [mem]}$



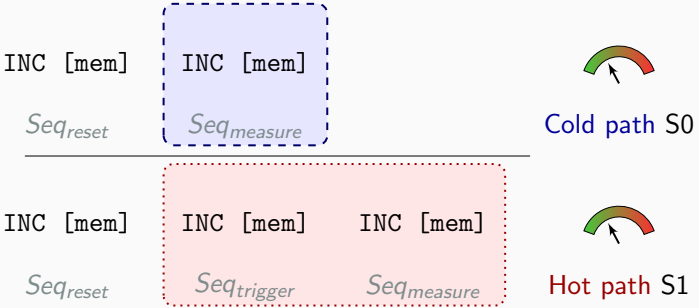
Testing A Sequence Triple

Example 1: $Seq_{measure} = Seq_{trigger} = Seq_{reset} = \texttt{INC [mem]}$



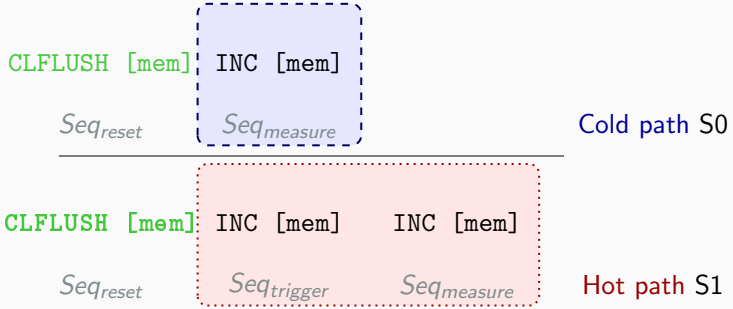
Testing A Sequence Triple

Example 1: $Seq_{measure} = Seq_{trigger} = Seq_{reset} = \texttt{INC [mem]}$



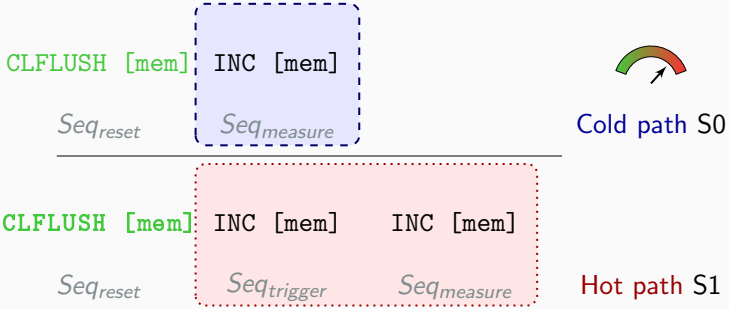
Testing A Sequence Triple

Example 2: $Seq_{measure} = Seq_{trigger} = \texttt{INC [mem]};$
$Seq_{reset} = \texttt{CLFLUSH [mem]}$



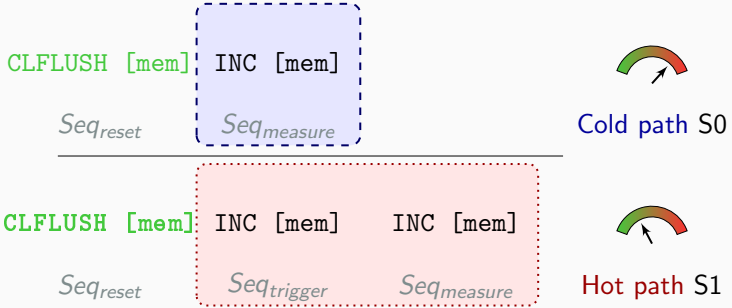
Testing A Sequence Triple

Example 2: $Seq_{measure} = Seq_{trigger} = \texttt{INC [mem]};$
$Seq_{reset} = \texttt{CLFLUSH [mem]}$



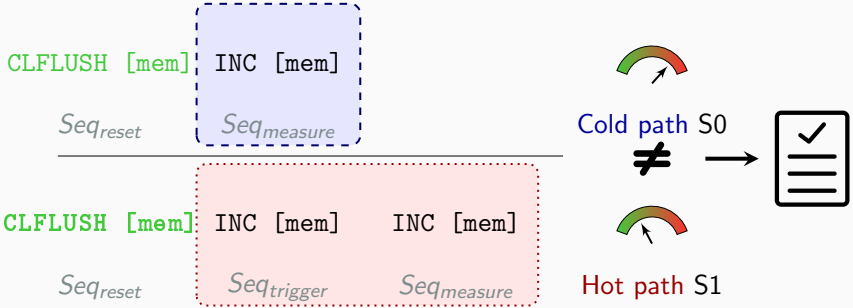
Testing A Sequence Triple

Example 2: $Seq_{measure} = Seq_{trigger} = \texttt{INC [mem]};$
$Seq_{reset} = \texttt{CLFLUSH [mem]}$



Testing A Sequence Triple

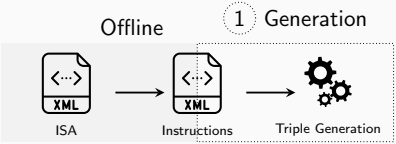
Example 2: $Seq_{measure} = Seq_{trigger} = \texttt{INC [mem]};$
$Seq_{reset} = \texttt{CLFLUSH [mem]}$



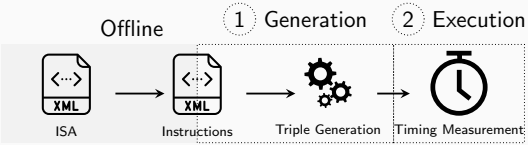
Osiris – Fuzzing x86 CPUs for Side Channels

Offline

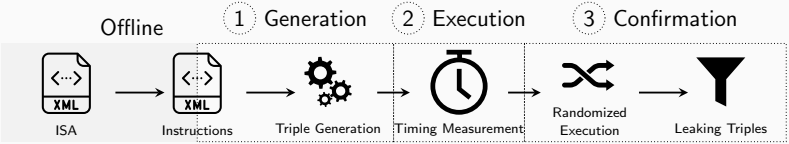
Osiris – Fuzzing x86 CPUs for Side Channels



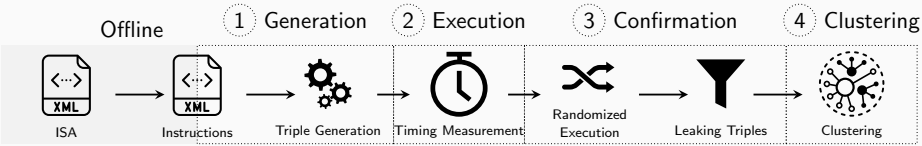
Osiris – Fuzzing x86 CPUs for Side Channels



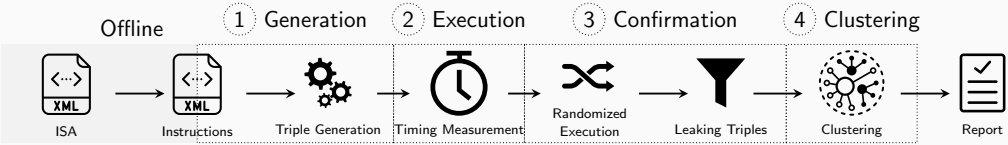
Osiris – Fuzzing x86 CPUs for Side Channels



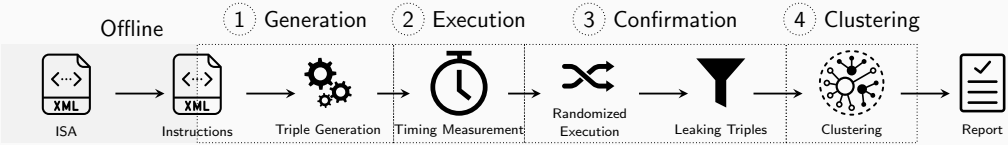
Osiris – Fuzzing x86 CPUs for Side Channels



Osiris – Fuzzing x86 CPUs for Side Channels



Osiris – Fuzzing x86 CPUs for Side Channels

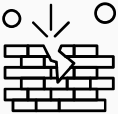


- Fuzzed on 5 different CPUs
- AMD and Intel

Osiris Results

~4 days per CPU

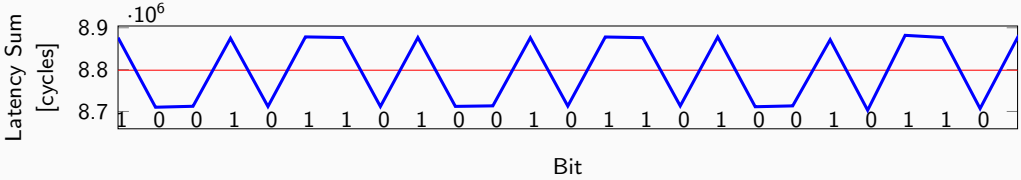
2 side channels rediscovered



4 new side channels

2 new attacks

RDRAND Covert Channel



- `RDRAND` cross-core interference

→ Cross-core cross-VM covert channel

- Tested on the AWS cloud

RDRAND Covert Channel - Properties

AMD and Intel

VM and native

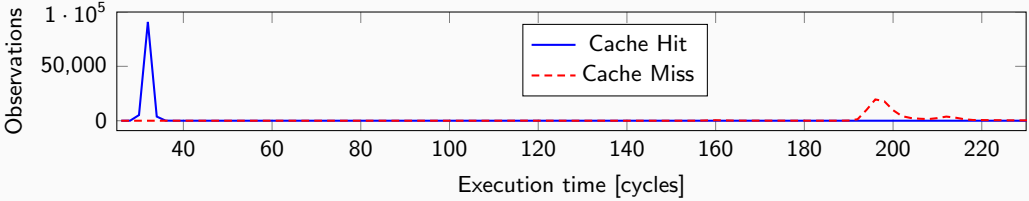
1000 bit/s

No memory

No detection

No mitigation

MOVNT Side Channel



- `MOVNT` can replace `CLFLUSH`
- Flushes data from all cache levels

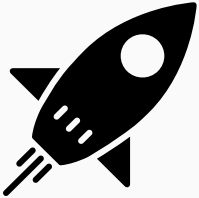
MOVNT Side Channel - Properties

Faster reload

Stealthy

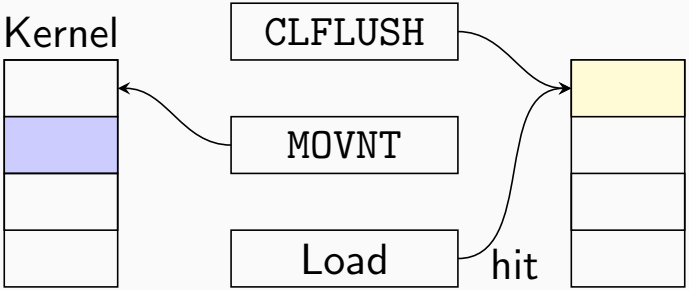
Not prevented by any cache design

MOVNT-based Meltdown

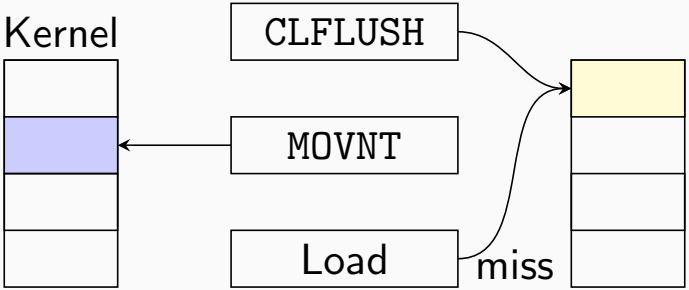


- Faster reload time → more leakage per transient window
- Previously: max. 3 bytes at once
- Meltdown PoC with `MOVNT`: 7.83 bytes at once

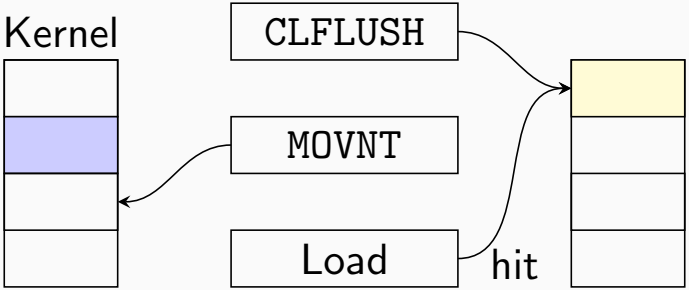
MOVNT KASLR Break



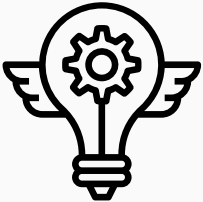
MOVNT KASLR Break



MOVNT KASLR Break



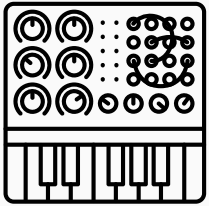
Transient Execution Attacks



- Improves implementation of transient execution attacks
- Can we find new transient execution attacks too?

→ Analyze them like the side channels

MDS Analysis



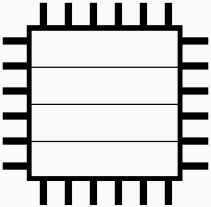
- Many Microarchitectural Data Sampling (MDS) attacks
  → ZombieLoad, RIDL, Fallout, Meltdown-UC
- Different variants and leakage targets
- Complex to reproduce and test all variations
- Common: require a fault or microcode assist

MDS

User Memory

	A	B
C	D	E
F	G	H
I	J	K
L	M	N
O	P	Q
R	S	T
U	V	W
X	Y	Z

```
char value = faulting[0]
```



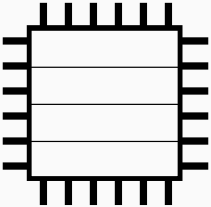
MDS

User Memory

	A	B
C	D	E
F	G	H
I	J	K
L	M	N
O	P	Q
R	S	T
U	V	W
X	Y	Z

```
char value = faulting[0]
```

Fault



MDS

User Memory

	A	B
C	D	E
F	G	H
I	J	K
L	M	N
O	P	Q
R	S	T
U	V	W
X	Y	Z

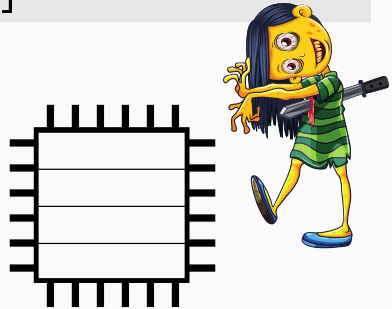
```
char value = faulting[0]
```

Fault

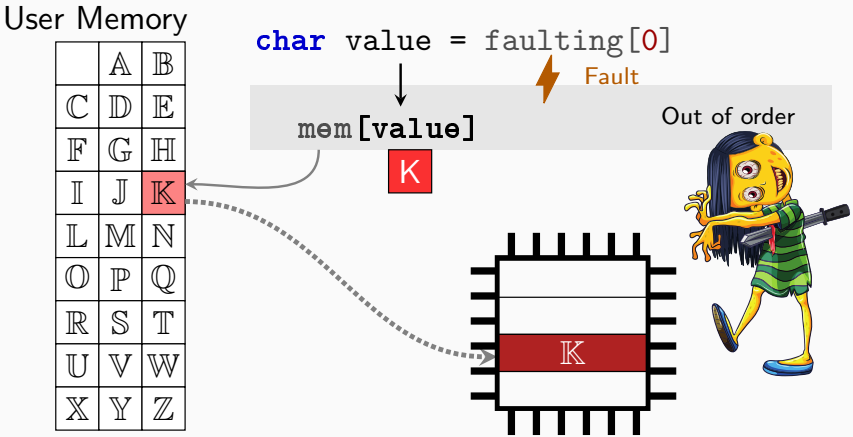
Out of order

```
mem[value]
```

K



MDS

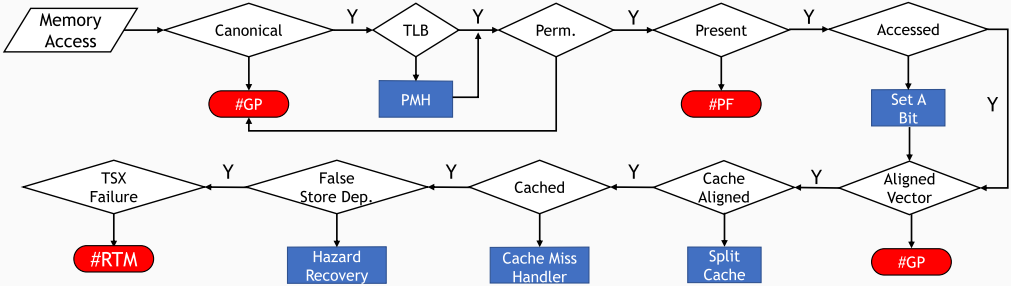

User Memory
A B
C D E
F G H
I J K
L M N
O P Q
R S T
U V W
X Y Z
char value = faulting[0]
Fault
Out of order
mem[value]
K
K

Memory Access Checks (simplified)

- Many possibilities for faults

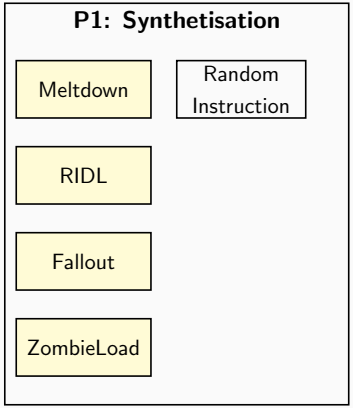
Memory Access Checks (simplified)

- Many possibilities for faults

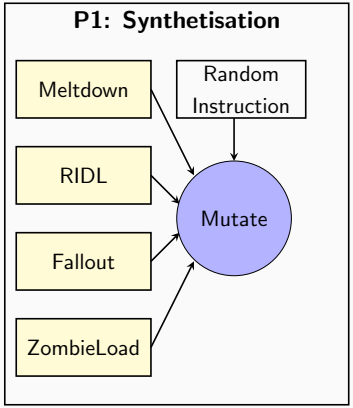


- Idea: mutation fuzzing for new variants

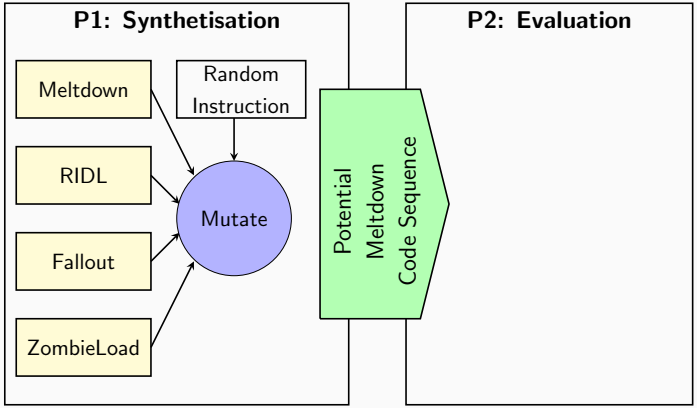
Transynther



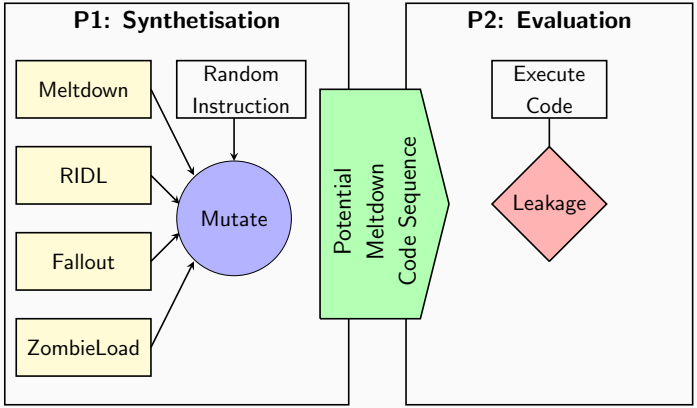
Transynther



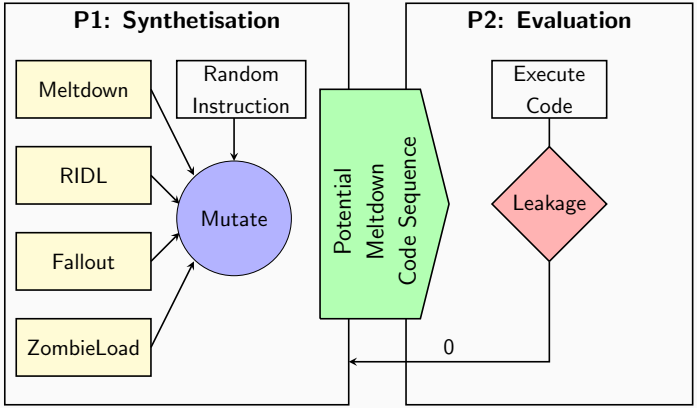
Transynther



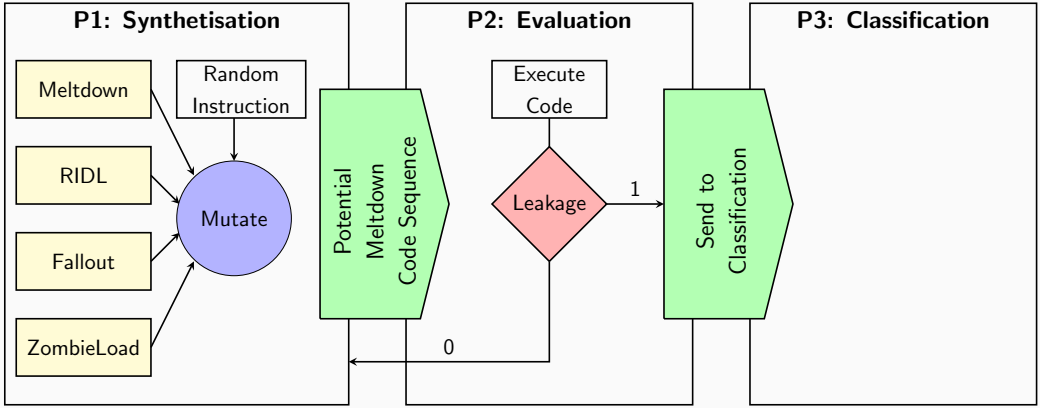
Transynther



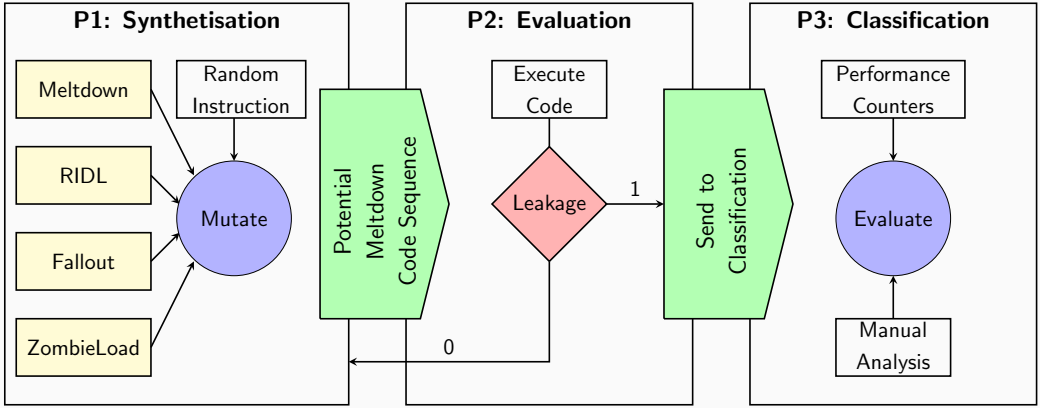
Transynther



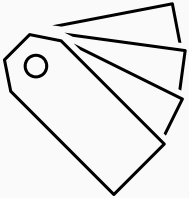
Transynther



Transynther



Buffer Grooming



- Fill microarchitectural buffers with known values

→ Rely on eviction sequences for buffers

- Leaked value indicates origin of leakage

Transynther Results

26 hours runtime

100 unique leakage patterns

7 attacks reproduced

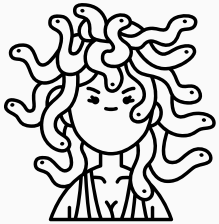
1 new vulnerability

1 regression

Medusa

- Medusa: new variant of ZombieLoad

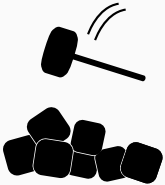
Medusa



- Medusa: new variant of ZombieLoad
- Leaks from write-combining buffer, i.e., `REP MOV`
- Used for fast memory copy, e.g., in OpenSSL or kernel

→ Leaked RSA key while decoding in OpenSSL

Ice Lake Regression

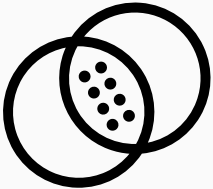


- Ice Lake microarchitecture reported no vulnerabilities
- Transynther found a regression via a small mutation

→ Re-enabled a "mitigated" variant

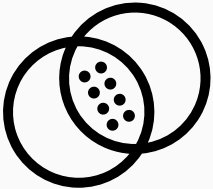
- Fixed via microcode update

Small Specialized Fuzzers



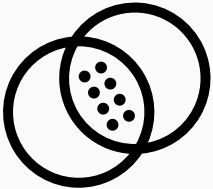
- Only cover small field of possible vulnerabilities

Small Specialized Fuzzers



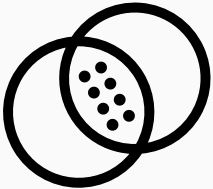
- Only cover small field of possible vulnerabilities
- Fuzzers are still simple

Small Specialized Fuzzers



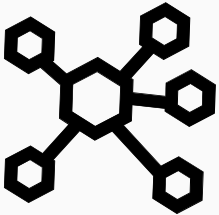
- Only cover small field of possible vulnerabilities
- Fuzzers are still simple
  - Narrow scope
  - No complex sequences
  - No guidance

Small Specialized Fuzzers



- Only cover small field of possible vulnerabilities
- Fuzzers are still simple
  - Narrow scope
  - No complex sequences
  - No guidance
- Very specialized fuzzers

Other CPU Fuzzer



- Some other specialized CPU fuzzers
  - Sandsifter: undocumented x86 instructions
  - ABSynthe: same-core contention side channels
  - FastSpec: Spectre variants
  - CrossTalk: cross-core transient execution attacks

Software vs. Hardware Fuzzing

- All low-hanging fruit
- Approximately as sophisticated as software fuzzing in 1990
- Majority of fuzzers does not use any guidance
- More research on feedback necessary

Summary

- Simple models are sufficient to find leakage
- Dumb fuzzers find leakage within hours
  - New vulnerability variants
  - New side channels
  - Regression in new CPUs
- Prediction: smarter fuzzers $\rightarrow$ more vulnerabilities

Open Source

https://github.com/CISPA/Osiris

USENIX'21

Daniel Weber, Ahmad Ibrahim, Hamed Nemati, Michael Schwarz, Christian Rossow.
Osiris: Automated Discovery of Microarchitectural Side Channels.

https://github.com/vernamlab/Medusa

USENIX'20

Daniel Moghimi, Moritz Lipp, Berk Sunar, Michael Schwarz.
Medusa: Microarchitectural Data Leakage via Automated Attack Synthesis.

FUZZ

ALL THE THINGS

CPU Fuzzing for Discovering Hardware-caused Information Leakage

Michael Schwarz

January 2022

CISPA Helmholtz Center for Information Security

References

C. Domas. Breaking the x86 ISA, v. 2017-07-27. In: Black Hat US (2017).

B. Gras, C. Giuffrida, M. Kurth, H. Bos, and K. Razavi. ABSynthe: Automatic Blackbox Side-channel Synthesis on Commodity Microarchitectures. In: NDSS. 2020.

D. Moghimi, M. Lipp, B. Sunar, and M. Schwarz. Medusa: Microarchitectural Data Leakage via Automated Attack Synthesis. In: USENIX Security Symposium. 2020.

H. Ragab, A. Milburn, K. Razavi, H. Bos, and C. Giuffrida. CrossTalk: Speculative Data Leaks Across Cores Are Real. In: S&P. 2021.

References ii

M. C. Tol, K. Yurtseven, B. Gulmezoglu, and B. Sunar. FastSpec: Scalable Generation and Detection of Spectre Gadgets Using Neural Embeddings. In: arXiv:2006.14147 (2020).

D. Weber, A. Ibrahim, H. Nemati, M. Schwarz, and C. Rossow. Osiris: Automated Discovery Of Microarchitectural Side Channels. In: USENIX Security Symposium. 2021.