Attaques sur la micro-architecture

Clémentine Maurice, CNRS, CRIStAL
@BloodyTangerine
24 Novembre 2022—8ème édition de la journée Sécu Min2Rien
Attacks on micro-architecture

- **hardware** usually modeled as an abstract layer behaving correctly
Attacks on micro-architecture

- **hardware** usually modeled as an abstract layer behaving correctly, but possible attacks
Attacks on micro-architecture

- hardware usually modeled as an abstract layer behaving correctly, but possible attacks
  - faults: bypassing software protections by causing hardware errors
  - side channels: observing side effects of hardware on computations
Attacks on micro-architecture

- **hardware** usually modeled as an abstract layer behaving correctly, but possible attacks
  - faults: bypassing software protections by causing hardware errors
  - side channels: observing side effects of hardware on computations

**identification**

![Graph showing cache hits and cache misses over access time](image)

<table>
<thead>
<tr>
<th>Access time [CPU cycles]</th>
<th>Number of accesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>$10^1$</td>
</tr>
<tr>
<td>200</td>
<td>$10^2$</td>
</tr>
<tr>
<td>300</td>
<td>$10^3$</td>
</tr>
<tr>
<td>400</td>
<td>$10^4$</td>
</tr>
</tbody>
</table>
Attacks on micro-architecture

- **hardware** usually modeled as an abstract layer behaving correctly, but possible attacks
  - faults: bypassing software protections by causing **hardware errors**
  - side channels: observing **side effects** of hardware on computations

![Graph showing cache hits and cache misses](image)

**identification**

- Number of accesses
  - cache hits
  - cache misses

**attack**

- retrieving secret keys, keystroke timings
- bypassing OS security (ASLR)
Attacker model

Hardware-based attacks
a.k.a physical attacks

- Physical access to hardware
  → embedded devices

Software-based attacks
a.k.a micro-architectural attacks

- Co-located or remote attacker
  → complex systems
Focus on one fault attack: Rowhammer

Overview of side-channel attacks

Conclusions
Focus on one fault attack: Rowhammer
DRAM organization
DRAM organization
DRAM organization

channel 0

back of DIMM: rank 1

front of DIMM: rank 0

channel 1
DRAM organization

channel 0

back of DIMM: rank 1

channel 1

front of DIMM: rank 0

chip
DRAM organization

- Chip
  - Bank 0
  - Row 0
  - Row 1
  - Row 2
  - ...
  - Row 32767
  - Row buffer
DRAM organization

- Chip
- Bank 0
  - Row 0
  - Row 1
  - Row 2
  - ...
  - Row 32767

- Row buffer

- 64k cells
- 1 capacitor, 1 transistor each
“It’s like breaking into an apartment by repeatedly slamming a neighbor’s door until the vibrations open the door you were after” – Motherboard Vice
“It’s like breaking into an apartment by repeatedly slamming a neighbor’s door until the vibrations open the door you were after” – Motherboard Vice
“It’s like breaking into an apartment by repeatedly slamming a neighbor’s door until the vibrations open the door you were after” – Motherboard Vice
“It’s like breaking into an apartment by repeatedly slamming a neighbor’s door until the vibrations open the door you were after” – Motherboard Vice
“It’s like breaking into an apartment by repeatedly slamming a neighbor’s door until the vibrations open the door you were after” – Motherboard Vice
“It’s like breaking into an apartment by repeatedly slamming a neighbor’s door until the vibrations open the door you were after” – Motherboard Vice
How to exploit random bit flips?

- Rowhammer was deemed non-exploitable and only a **reliability** issue
How to exploit random bit flips?

- Rowhammer was deemed non-exploitable and only a reliability issue
- bit flips are not random → highly reproducible flip pattern!
- ideas for exploitation
  1. bit flip in data structure, e.g., page table → privilege escalation
  2. bit flip in instruction opcode → privilege escalation
  3. bit flip in signature → fault-based cryptanalysis
Overview of side-channel attacks
From small optimizations...

- new microarchitectures yearly

2011: Sandy Bridge
2012: Ivy Bridge
2013: Haswell
2014: Broadwell
2015: Skylake
2016: Kaby Lake
2017: Coffee Lake
2018: Whiskey Lake
2019: Comet Lake/Ice Lake
2020: Tiger Lake
From small optimizations...

- new microarchitectures yearly
- performance improvement ≈ 5%

2011: Sandy Bridge
2012: Ivy Bridge
2013: Haswell
2014: Broadwell
2015: Skylake
2016: Kaby Lake
2017: Coffee Lake
2018: Whiskey Lake
2019: Comet Lake/Ice Lake
2020: Tiger Lake
From small optimizations...

- new microarchitectures yearly
- performance improvement ≈ 5%
- very small optimizations: caches, branch prediction...
... To microarchitectural side-channel attacks

- microarchitectural side channels come from these optimizations
... To microarchitectural side-channel attacks

- microarchitectural side channels come from these optimizations
- several processes are sharing microarchitectural components
... To microarchitectural side-channel attacks

- microarchitectural side channels come from these optimizations
- several processes are sharing microarchitectural components
- attacker infers information from a (vulnerable) victim process via hardware usage
... To microarchitectural side-channel attacks

- microarchitectural side channels come from these optimizations
- several processes are sharing microarchitectural components
- attacker infers information from a (vulnerable) victim process via hardware usage
- pure-software attacks by unprivileged processes
To microarchitectural side-channel attacks

- microarchitectural side channels come from these optimizations
- several processes are sharing microarchitectural components
- attacker infers information from a (vulnerable) victim process via hardware usage
- pure-software attacks by unprivileged processes
- sequences of benign-looking actions → hard to detect
Side-channel attacks
Side-channel attacks: Two faces of the same coin

Implementation

Algorithm 1: Square-and-multiply exponentiation

Input: base $b$, exponent $e$, modulus $n$

Output: $b^e \mod n$

$X \leftarrow 1$

for $i \leftarrow \text{bitlen}(e)$ downto 0 do

$X \leftarrow \text{multiply}(X, X)$

if $e_i = 1$ then

$X \leftarrow \text{multiply}(X, b)$

end

end

return $X$

Hardware
Research questions

1. Which software implementation is vulnerable?

2. Which hardware component is vulnerable?
1. Which software implementation is vulnerable?

State of the art (more or less)

1. Spend too much time reading OpenSSL code
2. Find vulnerability
3. Exploit it manually using known side channel → e.g. CPU cache
4. Publish
5. goto step 1

For example: CVE-2016-0702, CVE-2016-2178, CVE-2016-7440, CVE-2016-7439, CVE-2016-7438, CVE-2018-0495,
CVE-2020-16150
Canonical example: GnuPG 1.4.13 RSA square-and-multiply exponentiation

GnuPG version 1.4.13 (2013)

Algorithm 1: GnuPG 1.4.13 Square-and-multiply exponentiation

Input: base \( b \), exponent \( e \), modulus \( n \)

Output: \( b^e \mod n \)

\( X \leftarrow 1 \)

for \( i \leftarrow \text{bitlen}(e) \) downto 0 do

\( X \leftarrow \text{square}(X) \)

\( X \leftarrow X \mod n \)

if \( e_i = 1 \) then

\( X \leftarrow \text{multiply}(X, b) \)

\( X \leftarrow X \mod n \)

end

end

return \( X \)
Attacking GnuPG 1.4.13 RSA exponentiation

- monitor the square and multiply functions with a cache side channel to recover the bits of the secret exponent
Attacking GnuPG 1.4.13 RSA exponentiation

- monitor the square and multiply functions with a cache side channel to recover the bits of the secret exponent
2. Which hardware component leaks information?

State of the art (more or less)

1. Spend too much time reading Intel manuals
2. Find weird behavior in corner cases
3. Exploit it using a known vulnerability
4. Publish
5. goto step 1
Each component shared by two processes is a potential micro-architectural side-channel vector.
Hyper-threading: Same-core attacks

- threads sharing one core share resources: L1, L2 cache, branch predictor, TLB...

Translation leak-aside buffer
USENIX Sec'18
PortSmash
S&P'19
L1d, L1i, L2
BSDCon'05, CT-RSA'06
Branch Prediction
CT-RSA'07
• instructions are decomposed in uops to optimize Out-of-Order execution
• uops are dispatched to specialized execution units through CPU ports
• deterministic decomposition of instructions into uops
Port contention

No contention

All attacker instructions are executed in a row
→ fast execution time

Port contention

**No contention**

All attacker instructions are executed in a row  
→ fast execution time

**Contention**

Victim instructions delay the attacker instructions  
→ slow execution time

---

Port contention side-channel attack

Victim

\[
\begin{align*}
\text{secret} &= 0 \\
\text{secret} &= 1
\end{align*}
\]

Contention on Port 1

\[
\begin{align*}
\text{POPCNT} &\text{ %r8,%r8} \\
\text{VPBROADCASTD} &\text{ %xmm0, %ymm0}
\end{align*}
\]

Contention on Port 5

\[
\begin{align*}
\text{POPCNT} &\text{ %r8,%r8} \\
\text{VPBROADCASTD} &\text{ %xmm0, %ymm0}
\end{align*}
\]

Monitors port usage
Port contention side-channel attack

Victim

secret == 0

secret == 1

POPCNT %r8,%r8
POPCNT %r8,%r8
...
POPCNT %r8,%r8
POPCNT %r8,%r8

VPBROADCASTD %xmm0, %ymm0
VPBROADCASTD %xmm0, %ymm0
...
VPBROADCASTD %xmm0, %ymm0
VPBROADCASTD %xmm0, %ymm0

Contention on Port 1

Secret is 0!
Port contention side-channel attack

```
Victim

secret == 0
POPCNT %r8,%r8
POPCNT %r8,%r8
...
POPCNT %r8,%r8
POPCNT %r8,%r8

secret == 1
VPBROADCASTD %xmm0, %ymm0
VPBROADCASTD %xmm0, %ymm0
...
VPBROADCASTD %xmm0, %ymm0
VPBROADCASTD %xmm0, %ymm0
```

Contention on Port 5

Secret is 1!
Port contention: applications

- end-to-end attack on a TLS server (OpenSSL 1.1.0h): recovers a P-384 ECDSA private key
  - secret dependent on double-and-add operations of \texttt{ec\_wNAF\_mul} point multiplication
- SMoTherSpectre, a speculative code-reuse attack

---

A. Bhattacharyya et al. “SMoTherSpectre: Exploiting Speculative Execution through Port Contention”. In: CCS. 2019.
Possible side channels using components shared by a core?
Possible side channels using components shared by a core?

Stop sharing a core!
Cross-core attacks!

- cores also share resources: L3 cache, Ring Interconnect, GPU...

Translation leak-aside buffer
USENIX Sec'18

PortSmash
S&P'19

LLC attacks
USENIX'14, S&P'15

L1d, L1i, L2
cache attacks
BSDCon'05, CT-RSA'06

Branch Prediction
CT-RSA'07

Grand Pwning Unit
S&P'18

Lord of the Ring(s)
USENIX Sec'21
Caches on Intel CPUs

• L1 and L2 are private
• last-level cache divided in slices
• shared across cores
• inclusive

ring bus
Caches on Intel CPUs

- L1 and L2 are private
Caches on Intel CPUs

- L1 and L2 are private
- last-level cache
  - divided in slices
  - shared across cores
  - inclusive
Cache timing differences

Access time [CPU cycles]

Number of accesses

- cache hits
- cache misses
From theoretical to practical cache attacks

• first theoretical attack in 1996 by Kocher
• first practical attack on RSA in 2005 by Percival, on AES in 2006 by Osvik et al.
• renewed interest for the field in 2014 after Flush+Reload by Yarom and Falkner
• even more interest in 2018 after the disclosure of Spectre and Meltdown

Cache attacks techniques

- two (main) techniques
  1. **Flush+Reload** (Gullasch et al., Osvik et al., Yarom et al.)
  2. **Prime+Probe** (Percival, Osvik et al., Liu et al.)
- exploitable on x86 and ARM

---


Flush+Reload: Applications

- side channel attacks on cryptographic primitives:
  - RSA: 96.7% of secret key bits in a single signature
  - AES: full key recovery in 30000 dec. (a few seconds)
- attacks against pseudorandom number generators
- attacks against RSA key generation
- revival of Bleichenbacher attacks on TLS

---


Prime+Probe: Applications

- **cross-VM** side channel attacks on crypto implementations:
  - El Gamal (sliding window): full key recovery in 12 min.
- covert channels between virtual machines in the **cloud**
- tracking user behavior in the browser, in **JavaScript**

---

C. Maurice et al. “Hello from the Other Side: SSH over Robust Cache Covert Channels in the Cloud”. In: *NDSS’17*. 2017.
Possible side channels using components shared by a CPU?
Possible side channels using components shared by a CPU?

Stop sharing a CPU!?
Cross-CPU attacks!

- CPUs also share resources: DRAM

Translation leak-aside buffer
USENIX Sec'18

PortSmash
S&P'19

L1d, L1i, L2 cache attacks
BSDCon'05, CT-RSA'06

Branch Prediction
CT-RSA'07

LLC attacks
USENIX'14, S&P'15

DRAMA
USENIX Sec'16

Grand Pwning Unit
S&P'18

Lord of the Ring(s)
USENIX Sec'21
Conclusions
Conclusions

- first paper by Kocher in 1996: 25 years of research in this area
Conclusions

- first paper by Kocher in 1996: 25 years of research in this area
- domain still in expansion: increasing number of papers published since 2015
Conclusions

- first paper by Kocher in 1996: 25 years of research in this area
- domain still in expansion: increasing number of papers published since 2015
- any shared component is a potential side-channel vector
Conclusions

- first paper by Kocher in 1996: *25 years of research* in this area
- domain still in expansion: increasing number of papers published since 2015
- any *shared component* is a potential side-channel vector
- it’s **really** hard not to share a component
Conclusions

• first paper by Kocher in 1996: 25 years of research in this area
• domain still in expansion: increasing number of papers published since 2015
• any shared component is a potential side-channel vector
• it’s really hard not to share a component
• micro-architectural attacks require a low-level understanding and control over the components
Conclusions

• first paper by Kocher in 1996: 25 years of research in this area
• domain still in expansion: increasing number of papers published since 2015
• any shared component is a potential side-channel vector
• it’s really hard not to share a component
• micro-architectural attacks require a low-level understanding and control over the components
• how to prevent attacks based on performance optimizations without removing performance?
Attaques sur la micro-architecture

Clémentine Maurice, CNRS, CRIStAL
@BloodyTangerine
24 Novembre 2022—8ème édition de la journée Sécu Min2Rien