Software-based Side-Channel Attacks and Defenses in Restricted Environments

Heinz Zemanek Preis Hearing 2020

Michael Schwarz
IAIK, Graz University of Technology
May 28, 2020
Motivation

Michael Schwarz — IAIK – Graz University of Technology
Research Question
How far can we reduce requirements?
24 (13 tier 1)

9 (4 tier 1)

34

9

2975 (513 first author)

10 (3 co-authors)

Michael Schwarz — IAIK – Graz University of Technology
\[ M \equiv C^d \pmod{n} \]
Side-channel Attacks

\[ M \equiv C^d \pmod{n} \]

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

Description

Software

Execution time

Michael Schwarz — IAIK – Graz University of Technology
Side-channel Attacks

\[ M \equiv C^d \pmod{n} \]

<table>
<thead>
<tr>
<th>Description</th>
<th>Software</th>
</tr>
</thead>
</table>
| \( C^n = \begin{cases} 
(C^2)^{\frac{n}{2}} & \text{if } n \equiv 0 \pmod{2} \\
C \cdot (C^2)^{\frac{n-1}{2}} & \text{if } n \equiv 1 \pmod{2} 
\end{cases} \) |          |

<table>
<thead>
<tr>
<th>Hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>'1'</td>
</tr>
<tr>
<td>'0'</td>
</tr>
</tbody>
</table>

Execution time | Power consumption
Side-channel Attacks

\[ M \equiv C^d \pmod{n} \]

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

- **Description**
- **Software**
- **Hardware**

Execution time

Power consumption

CPU caches

Michael Schwarz — IAIK – Graz University of Technology
• Specific implementation of a CPU
• Specific implementation of a CPU

Decoder
• Specific implementation of a CPU

Decoder

Predictor
Microarchitectural Elements

- Specific implementation of a CPU

Decoder
Predictor
Scheduler
Microarchitectural Elements

- Specific implementation of a CPU

- Decoder
- Predictor
- Scheduler
- Buffer
• Specific implementation of a CPU

Decoder
Predictor
Scheduler
Buffer
Caches
printf("%d", i);
printf("%d", i);
printf("%d", i);
printf("%d", i);
```c
printf("%d", i);
printf("%d", i);
```
printf("%d", i);
printf("%d", i);
printf("%d", i);
printf("%d", i);
CPU Cache

printf("%d", i);
printf("%d", i);

Cache miss
Cache hit

Request
Response
CPU Cache

printf("%d", i);  
Cache hit

printf("%d", i);  
Cache miss

DRAM access, slow

Request
Response
CPU Cache

printf("%d", i);

printf("%d", i);

DRAM access, slow

No DRAM access, much faster

Cache miss

Cache hit

Request

Response

5

Michael Schwarz — IAIK – Graz University of Technology
FANTASTIC TIMERS
AND WHERE TO FIND THEM
HIGH-RESOLUTION MICROARCHITECTURAL ATTACKS IN JAVASCRIPT
FC'17
• High-resolution timer to measure microarchitectural effects
• High-resolution timer to measure microarchitectural effects

`rdtsc`
• High-resolution timer to measure microarchitectural effects

![Clock]

- rdtsc
- performance.now()
Timers in JavaScript

- High-resolution timer to measure microarchitectural effects

```javascript
performance.now()
```

[...] represent times as floating-point numbers with up to microsecond precision.

— Mozilla Developer Network
Attacker

attack.js
Building a timer - Web worker

Attacker

attack.js

WebWorker

inc.js
Building a timer - Web worker

Attacker: attack.js

SharedArrayBuffer:
- timestamp

WebWorker: inc.js

Michael Schwarz — IAIK – Graz University of Technology
Building a timer - Web worker

Attacker
attack.js

SharedArrayBuffer
timestamp

WebWorker
inc.js
Building a timer - Web worker

attacker.js

SharedArrayBuffer

timestamp

WebWorker

inc.js

Attacker

$ts()$

++

Michael Schwarz — IAIK – Graz University of Technology
Building a timer - Web worker

Attacker
attack.js

timestamp

SharedArrayBuffer

WebWorker
inc.js

++

ts()
Building a timer - Web worker

Attacker
attack.js

SharedArrayBuffer
tsn()

tsn()

WebWorker
inc.js

2 ns

15 ns

Michael Schwarz — IAIK – Graz University of Technology
• Rounding timers is not a solution
Summary

- Rounding timers is not a solution
- State of the art for timers in JavaScript

---

**FC’17**

Michael Schwarz, Clémentine Maurice, Daniel Gruss and Stefan Mangard.

Fantastic Timers and Where to Find Them: High-Resolution Microarchitectural Attacks in JavaScript.
Summary

- Rounding timers is not a solution
- State of the art for timers in JavaScript
- Microarchitectural attacks in the browser are possible again

FC’17

Michael Schwarz, Clémentine Maurice, Daniel Gruss and Stefan Mangard.
Fantastic Timers and Where to Find Them: High-Resolution Microarchitectural Attacks in JavaScript.
Application

Untrusted part

Create Enclave

Operating System
Application

Untrusted part

Create Enclave

Trusted part

Call Gate

Trusted Fnc.

Operating System

Michael Schwarz — IAIK – Graz University of Technology
SGX

Application

Untrusted part

Create Enclave

Call Trusted Fnc.

Trusted part

Call Gate

Trusted Fnc.

Operating System

Michael Schwarz — IAIK – Graz University of Technology
Application

Untrusted part → Trusted part

- Create Enclave
- Call Trusted Fnc.
- Call Gate
- Trusted Fnc.

Operating System
Application

Untrusted part

Create Enclave

Call Trusted Fnc.

Trusted part

Call Gate

Trusted Fnc.

Operating System
Application

Untrusted part

- Create Enclave
- Call Trusted Fnc.

Trusted part

- Call Gate
- Trusted Fnc.
- Return

Operating System
Application

Untrusted part

Create Enclave

Call Trusted Fnc.

Trusted part

Call Gate

Trusted Fnc.

Return

Operating System
Application

Untrusted part

Create Enclave

Call Trusted Fnc.

... 

Trusted part

Call Gate

Trusted Fnc.

Return

Operating System
Application

Untrusted part

Create Enclave

Call Trusted Fnc.

... 

Trusted part

Call Gate

Trusted Fnc.

Return

Operating System

Michael Schwarz — IAIK – Graz University of Technology
Victim
Attack Settings

SGX

Victim
Victim

SGX

RSA Signature
+ private key

Public API
Attack Settings

Attacker

Victim

SGX

RSA Signature + private key

Public API
Attack Settings

Attacker

SGX

Victim

RSA Signature
+ private key

Public API
Attack Settings

Attacker

SGX

Malware

Loader

Victim

SGX

RSA Signature
+ private key

Public API
Attack Settings

Attacker

SGX

Malware

Loader

Victim

SGX

RSA Signature + private key

Public API
Attack Settings

- SGX Malware
  - Loader
  - L1/L2 Cache

- SGX RSA Signature + private key
  - Public API
  - L1/L2 Cache

Michael Schwarz — IAIK – Graz University of Technology
Attack Settings

Attacker
- SGX
- Malware (Prime+Probe)
- Loader
- L1/L2 Cache

Victim
- SGX
- RSA Signature + private key
- Public API
- L1/L2 Cache

Shared LLC
SGX Limitations

No privileges

No syscall
No shared memory
No rdtsc
No physical addresses
No 2 MB pages
SGX Limitations

- No privileges
- No syscall
- No shared memory
- No rdtsc
- No physical addresses
- No 2 MB pages

Michael Schwarz — IAIK – Graz University of Technology
SGX Limitations

No privileges

No syscall

No shared memory
SGX Limitations

- No privileges
- No syscall
- No shared memory
- No rdtsc
- No physical addresses
- No 2 MB pages
Raw Prime+Probe trace...
...processed with a simple moving average...
...allows to clearly see the bits of the exponent
• Attacks from within SGX possible
• Attacks from within SGX possible
• New state-of-the-art high-resolution timer
Summary

• Attacks from within SGX possible
• New state-of-the-art high-resolution timer
• Leaked 4096-bit RSA key (11 traces)

DIMVA’17

Michael Schwarz, Samuel Weiser, Daniel Gruss, Clémentine Maurice, Stefan Mangard.
Malware Guard Extension: Using SGX to Conceal Cache Attacks.
• Attacks from within SGX possible
• New state-of-the-art high-resolution timer
• Leaked 4096-bit RSA key (11 traces)
• SGX allows hiding an attack
KeyDrown

Kernel

/dev/input/event*

Real key

Fake key
KeyDrown

Widget

Library
libgtk / libinput

Kernel

Hidden Window
Application Window

/dev/input/event*

Real key
Fake key
Flush+Reload on libgdk-3.so

Without Keydrown, F-Score 0.99

With Keydrown, F-Score 0.09
Without Keydrown, F-Score 0.94

With Keydrown, F-Score 0.14
• Keystroke-timing attack only with high-resolution timer

**NDSS’18**

**Michael Schwarz**, Moritz Lipp, Daniel Gruss, Samuel Weiser, Clémentine Maurice, Raphael Spreitzer, Stefan Mangard.

KeyDrown: Eliminating Software-Based Keystroke Timing Side-Channel Attacks.
• Keystroke-timing attack only with high-resolution timer
• First accurate Prime+Probe attack on kernel
Summary

- Keystroke-timing attack only with high-resolution timer
- First accurate Prime+Probe attack on kernel
- First generic countermeasure against keystroke-timing attack

NDSS’18

Michael Schwarz, Moritz Lipp, Daniel Gruss, Samuel Weiser, Clémentine Maurice, Raphael Spreitzer, Stefan Mangard.
KeyDrown: Eliminating Software-Based Keystroke Timing Side-Channel Attacks.
Summary

- Keystroke-timing attack only with high-resolution timer
- First accurate Prime+Probe attack on kernel
- First generic countermeasure against keystroke-timing attack
- PoC for Linux and Android

NDSS’18

Michael Schwarz, Moritz Lipp, Daniel Gruss, Samuel Weiser, Clémentine Maurice, Raphael Spreitzer, Stefan Mangard.
KeyDrown: Eliminating Software-Based Keystroke Timing Side-Channel Attacks.
JavaScript zero

REAL JavaScript AND ZERO SIDE-CHANNEL ATTACKS

NDSS'18
Landscape of Microarchitectural Attacks

11 known attacks

5 identified categories
Landscape of Microarchitectural Attacks

11 known attacks

Memory addresses

5 identified categories
Landscape of Microarchitectural Attacks

- 11 known attacks
- Memory addresses
- Accurate timing
- 5 identified categories
- Multithreading
- Shared data
- Sensor API
Landscape of Microarchitectural Attacks

- 11 known attacks
- 5 identified categories
- Memory addresses
- Accurate timing
- Multithreading

Michael Schwarz — IAIK – Graz University of Technology
Landscape of Microarchitectural Attacks

- 11 known attacks
- 5 identified categories
- Memory addresses
- Accurate timing
- Multithreading
- Shared data
Landscape of Microarchitectural Attacks

11 known attacks

Memory addresses

Multithreading

5 identified categories

Accurate timing

Shared data

Sensor API
- Functions and properties are replaced by wrappers
<table>
<thead>
<tr>
<th>Defense</th>
<th>Prevents</th>
<th>Rowhammer.js</th>
<th>Page Deduplication</th>
<th>DRAM Covert Channel</th>
<th>Anti-ASLR</th>
<th>Cache Eviction</th>
<th>Keystroke Timing</th>
<th>Browser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffer ASLR</td>
<td></td>
<td>O</td>
<td></td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Array preloading</td>
<td></td>
<td>O</td>
<td></td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Non-deterministic array</td>
<td></td>
<td>O</td>
<td></td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Array index randomization</td>
<td></td>
<td>O</td>
<td></td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Low-resolution timestamp</td>
<td></td>
<td>O</td>
<td></td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Fuzzy time</td>
<td></td>
<td>O</td>
<td></td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>WebWorker polyfill</td>
<td></td>
<td>O</td>
<td></td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Message delay</td>
<td></td>
<td>O</td>
<td></td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Slow SharedArrayBuffer</td>
<td></td>
<td>O</td>
<td></td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>No SharedArrayBuffer</td>
<td></td>
<td>O</td>
<td></td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Summary</td>
<td></td>
<td>O</td>
<td></td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

Fully prevented (●), partly prevented (○), not prevented (O). Policies must be combined (*).
• First **systematic analysis** of the attack surface

---

**NDSS’18**

**Michael Schwarz, Moritz Lipp, Daniel Gruss.**

JavaScript Zero: Real JavaScript and Zero Side-Channel Attacks.
Summary

- First systematic analysis of the attack surface
- Microarchitectural attacks in the browser are still possible

NDSS’18

Michael Schwarz, Moritz Lipp, Daniel Gruss.
JavaScript Zero: Real JavaScript and Zero Side-Channel Attacks.
Summary

• First systematic analysis of the attack surface
• Microarchitectural attacks in the browser are still possible
• First efficient countermeasures in browsers

NDSS’18

Michael Schwarz, Moritz Lipp, Daniel Gruss.
JavaScript Zero: Real JavaScript and Zero Side-Channel Attacks.
• First systematic analysis of the attack surface
• Microarchitectural attacks in the browser are still possible
• First efficient countermeasures in browsers
• More microarchitectural attacks in JavaScript will appear

NDSS’18

Michael Schwarz, Moritz Lipp, Daniel Gruss.
JavaScript Zero: Real JavaScript and Zero Side-Channel Attacks.
A Double Fetch

string

/\texttt{path/file}\ 0 \texttt{payload}\ 0

\begin{tikzpicture}
\node (kernel) at (0,0) {Kernel};
\node (user) at (5,0) {User};
\draw[<->] (kernel.east) -- (user.west) node[midway,above] {length};
\end{tikzpicture}
A Double Fetch

string

string

Kernel

User

Michael Schwarz — IAIK — Graz University of Technology
A Double Fetch

```
string

/path/fileXpayload Ø
```

```
Kernel

User

string[10] = 'X';
```
A Double Fetch

string

/ path / file X payload \emptyset

\text{length}

\text{Kernel}

\downarrow \text{copy string}

\begin{array}{c}
\text{overflow}
\end{array}

\text{User}

\text{string}[10] = 'X';
Double-fetch Detection and Exploitation

![Graph showing runtime (cycles) vs. access time (cycles)]
Double-fetch Detection and Exploitation

```
3 3.5 4 4.5 5 5.5 6 6.5 7
3 3.5 4 4.5 5 5.5 6 6.5 7
```

Runtime [cycles]

Access time [cycles]

First access

\[ \cdot 10^5 \]

Michael Schwarz — IAIK — Graz University of Technology
Double-fetch Detection and Exploitation

Access time [cycles]

Runtime [cycles]

First access

Second access

\( \cdot 10^5 \)
Double-fetch Detection and Exploitation

- Access time [cycles]
- Runtime [cycles]

- First access
- Modify value
- Second access with modified value

Michael Schwarz — IAIK – Graz University of Technology
• Fuzzing + cache attacks $\rightarrow$ double-fetch bugs
• Fuzzing + cache attacks → double-fetch bugs
• Cache as trigger outperforms state of the art
Summary

- Fuzzing + cache attacks → double-fetch bugs
- Cache as trigger outperforms state of the art
- Hardware transactional memory prevents exploitation

AsiaCCS’18

Michael Schwarz, Daniel Gruss, Moritz Lipp, Clémentine Maurice, Thomas Schuster, Anders Fogh, Stefan Mangard.
Automated Detection, Exploitation, and Elimination of Double-Fetch Bugs using Modern CPU Features.
• Template attack on JavaScript properties
• Template attack on JavaScript properties
• Template attack on JavaScript properties
• Template attack on JavaScript properties
• Template attack on JavaScript properties
• Template attack on JavaScript properties
• Template attack on JavaScript properties
• Template attack on JavaScript properties
Browser + version
Detection

Browser + version

Privacy extensions
Detection

- Browser + version
- Privacy extensions
- Private mode
Detection

Browser + version

Privacy extensions

Operating system

Private mode
Detection

- **Browser + version**
- **Privacy extensions**
- **Private mode**
- **Operating system**
- **CPU vendor**
- **Virtual machine**
• JavaScript Template attacks detect environment properties
• JavaScript Template attacks detect environment properties
• Enables exploits, side-channel attacks and plausible phishing
• JavaScript Template attacks detect environment properties
• Enables exploits, side-channel attacks and plausible phishing
• Tool for browser vendors to find leakage

NDSS’19

Michael Schwarz, Florian Lackner, Daniel Gruss.
Summary

- JavaScript Template attacks detect environment properties
- Enables exploits, side-channel attacks and plausible phishing
- Tool for browser vendors to find leakage
- Advances field of fingerprinting

NDSS’19

Michael Schwarz, Florian Lackner, Daniel Gruss.
char value = faulting[0]
### User Memory

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>G</td>
</tr>
<tr>
<td></td>
<td>I</td>
<td>J</td>
</tr>
<tr>
<td></td>
<td>L</td>
<td>M</td>
</tr>
<tr>
<td></td>
<td>O</td>
<td>P</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>U</td>
<td>V</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>Y</td>
</tr>
</tbody>
</table>

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

![Fault symbol]
User Memory

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>F</td>
<td>G</td>
<td>H</td>
<td>I</td>
<td>J</td>
</tr>
<tr>
<td>L</td>
<td>M</td>
<td>N</td>
<td>O</td>
<td>P</td>
</tr>
<tr>
<td>Q</td>
<td>R</td>
<td>S</td>
<td>T</td>
<td>U</td>
</tr>
<tr>
<td>V</td>
<td>W</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
</tbody>
</table>

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

```c
mem[value]
```

Fault

Out of order
Fill-buffer Leakage

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]
```

```
mem[value]
```

Out of order

Fault

Michael Schwarz — IAIK – Graz University of Technology
Results

AES-NI key
Results

AES-NI key

SGX sealing key
Results

AES-NI key

SGX sealing key

Cross-VM covert channel
Results

- AES-NI key
- SGX sealing key
- Cross-VM covert channel
- Keyword matching

Michael Schwarz — IAIK – Graz University of Technology
Results

- AES-NI key
- SGX sealing key
- Cross-VM covert channel
- Keyword matching
- URL recovery
Results

- AES-NI key
- SGX sealing key
- Cross-VM covert channel
- Keyword matching
- URL recovery
- Targeted leakage
• Meltdown on the line-fill buffer
• Meltdown on the line-fill buffer
• Data sampling attacks are powerful

CCS’19

Michael Schwarz, Moritz Lipp, Daniel Moghimi, Jo Van Bulck, Julian Stecklina, Thomas Prescher, Daniel Gruss.
ZombieLoad: Cross-Privilege-Boundary Data Sampling.
Summary

• Meltdown on the line-fill buffer
• Data sampling attacks are powerful
• Basis for Load Value Injection (LVI)

CCS’19

Michael Schwarz, Moritz Lipp, Daniel Moghimi, Jo Van Bulck, Julian Stecklina, Thomas Prescher, Daniel Gruss.

ZombieLoad: Cross-Privilege-Boundary Data Sampling.
- Attack research is necessary for effective defenses
• Attack research is necessary for effective defenses
• Data-dependent optimizations often lead to side channel
• Attack research is necessary for effective defenses
• Data-dependent optimizations often lead to side channel
• Trade-off between performance and security
Conclusion

- Attack research is necessary for effective defenses
- Data-dependent optimizations often lead to side channel
- Trade-off between performance and security
- Created new field of transient-execution attacks
Software-based Side-Channel Attacks and Defenses in Restricted Environments

*Heinz Zemanek Preis Hearing 2020*

Michael Schwarz
IAIK, Graz University of Technology
May 28, 2020