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

PhD Defense

Michael Schwarz
IAIK, Graz University of Technology
November 5, 2019
Motivation

Remote Attack!

Research Question
How far can we reduce requirements?
Numbers

24 (13 tier 1)

7 (4 tier 1)

30

7

8

10 (3 co-authors)
Side-channel Attacks

\[ M \equiv C^d \pmod{n} \]
### Side-channel Attacks

**Description**

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

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

**Software**

<table>
<thead>
<tr>
<th>Execution time</th>
<th>Power consumption</th>
<th>CPU caches</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

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

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

<table>
<thead>
<tr>
<th>Execution time</th>
<th>Power consumption</th>
<th>CPU caches</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘1’</td>
<td>‘0’</td>
<td></td>
</tr>
</tbody>
</table>
Modern CPUs contain multiple microarchitectural elements.
Modern CPUs contain multiple microarchitectural elements.
• Modern CPUs contain multiple microarchitectural elements
Modern CPUs contain multiple microarchitectural elements.
Modern CPUs contain multiple microarchitectural elements.
Modern CPUs contain multiple microarchitectural elements.
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);

Cache miss

Request
Response
CPU Cache

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

Cache miss

Cache hit

Request

Response
CPU Cache

- **printf(”%d”, i);**
  - **Cache miss**
  - **DRAM access,** slow

- **printf(”%d”, i);**
  - **Cache hit**
  - Request
  - Response
CPU Cache

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

DRAM access,
slow

Cache miss

Cache hit

No DRAM access,
much faster

Request

Response

6

Michael Schwarz — IAIK – Graz University of Technology
Flush+Reload

ATTACKER

flush
access

Shared Memory

cached

Shared Memory

cached

VICTIM

access
Flush+Reload

ATTACKER
flush
access

Shared Memory

VICTIM
access
Flush+Reload

ATTACKER

flush
access

Shared Memory

VICTIM

access
Flush+Reload

ATTACKER

flush
access

Shared Memory

VICTIM

access
Flush+Reload

ATTACKER

flush

access

Shared Memory

VICTIM

access
Flush+Reload

ATTACKER

flush

access

Shared Memory

VICTIM

access

Victim accessed (fast) vs Victim did not access (slow)
Prime+Probe

ATTACKER
prime
access

VICTIM
access
Prime+Probe

ATTACKER
prime
access

VICTIM
access

Michael Schwarz — IAIK – Graz University of Technology
Prime+Probe

ATTACKER

prime
access

VICTIM

access
Victim did not access (fast) vs Victim accessed (slow)
Contribution
FANTASTIC TIMERS
AND WHERE TO FIND THEM
HIGH-RESOLUTION MICROARCHITECTURAL
ATTACKS IN JAVASCRIPT
• **High-resolution timer** to measure microarchitectural effects
Timers in JavaScript

- **High-resolution timer** to measure microarchitectural effects

- `window.performance.now()` represents times as floating-point numbers with up to microsecond precision.

---

Michael Schwarz — IAIK – Graz University of Technology
High-resolution timer to measure microarchitectural effects

- `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
...up to microsecond precision?

Firefox $\leq 36$ | $1 \cdot 10^{-3}$
...up to microsecond precision?

Edge 38

Firefox ≤ 36

$1 \cdot 10^{-3}$
...up to microsecond precision?

- W3C standard: 5
- Edge 38: 1
- Firefox ≤ 36: $1 \cdot 10^{-3}$
...up to microsecond precision?

<table>
<thead>
<tr>
<th>Browser Configuration</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firefox ≥ 37/Chrome/Safari</td>
<td>5</td>
</tr>
<tr>
<td>W3C standard</td>
<td>5</td>
</tr>
<tr>
<td>Edge 38</td>
<td>1</td>
</tr>
<tr>
<td>Firefox ≤ 36</td>
<td>$1 \cdot 10^{-3}$</td>
</tr>
</tbody>
</table>
...up to microsecond precision?

<table>
<thead>
<tr>
<th>Browser</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tor</td>
<td>$1 \cdot 10^5$</td>
</tr>
<tr>
<td>Firefox $\geq$ 37/Chrome/Safari</td>
<td>5</td>
</tr>
<tr>
<td>W3C standard</td>
<td>5</td>
</tr>
<tr>
<td>Edge 38</td>
<td>1</td>
</tr>
<tr>
<td>Firefox $\leq$ 36</td>
<td>$1 \cdot 10^{-3}$</td>
</tr>
</tbody>
</table>
...up to microsecond precision?

<table>
<thead>
<tr>
<th>Browser</th>
<th>Fuzzyfox</th>
<th>Tor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firefox ≥ 37/Chrome/Safari</td>
<td>$1 \cdot 10^5$</td>
<td>$1 \cdot 10^5$</td>
</tr>
<tr>
<td>W3C standard</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Edge 38</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Firefox ≤ 36</td>
<td>$1 \cdot 10^{-3}$</td>
<td></td>
</tr>
</tbody>
</table>
Building a timer - Web worker

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
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

timestamp

WebWorker
inc.js

SharedArrayBuffer

++

ts()
Rounding timers is not a solution
• Rounding timers is not a solution
• Multithreading + shared data → new timers
• Rounding timers is not a solution
• Multithreading + shared data → new timers
• 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.
SGX malware DIMVA'17
Application

Untrusted part

Operating System
Application

Untrusted part

Create Enclave

Operating System
SGX

Application

Untrusted part

Create Enclave

Call Trusted Fnc.

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**
  - Create Enclave
  - Call Trusted Fnc.

- **Trusted part**
  - Call Gate
  - Trusted Fnc.

**Operating System**
Application

Untrusted part

Create Enclave

Call Trusted Fnc.

Call Gate

Trusted part

Trusted Fnc.

Return

Operating System

Michael Schwarz — IAIK – Graz University of Technology
SGX

Application

Untrusted part
- Create Enclave
- Call Trusted Fnc.

Trusted part
- Call Gate
- Trusted Fnc.
- Return

Operating System

Michael Schwarz — IAIK – Graz University of Technology
SGX

Application

Untrusted part

Create Enclave

Call Trusted Fnc.

... 

Trusted part

Call Gate

Trusted Fnc.

Return

Operating System
Victim
Attack Settings

Victim

SGX
Attack Settings

Victim

SGX

RSA
Signature
+
private key

Public API
**Attack Settings**

- **Attacker**
- **Victim**
  - **SGX**
  - **RSA Signature**
  - + private key
  - **Public API**
Attack Settings

Attacker

SGX

Victim

SGX

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

Attacker

SGX

Malware

Loader

L1/L2 Cache

Victim

SGX

RSA Signature

+ private key

Public API

L1/L2 Cache
Attack Settings

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

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

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
SGX Limitations

- No privileges
- No syscall
- No shared memory
- No rdtsc
- No physical addresses
SGX Limitations

- No privileges
- No syscall
- No shared memory
- No rdtsc
- No physical addresses
- No 2 MB pages
CPU cycles per increment

```c
rdtsc 1
```

```c
timestamp = rdtsc();
```
CPU cycles per increment

```
while (1) {
    timestamp ++;
}
```
CPU cycles per increment

<table>
<thead>
<tr>
<th></th>
<th>rdtsc</th>
<th>C</th>
<th>Assembly</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>4.7</td>
<td>4.67</td>
</tr>
</tbody>
</table>

```c

mov &timestamp, %rcx
1: incl (%rcx)
jmp 1b
```

Michael Schwarz — IAIK – Graz University of Technology
CPU cycles per increment

<table>
<thead>
<tr>
<th>Method</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdtsc</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>4.7</td>
</tr>
<tr>
<td>Assembly</td>
<td>4.67</td>
</tr>
<tr>
<td>Optimized</td>
<td>0.87</td>
</tr>
</tbody>
</table>

C code:

```
mov &timestamp, %rcx
1:  inc %rax
mov %rax, (%rcx)
jmp 1b
```
Raw Prime+Probe trace...
...processed with a simple moving average...
...allows to clearly see the bits of the exponent
• Attacks from within SGX possible

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 high-resolution timer
• Attacks from within SGX possible
• New high-resolution timer
• Leaked 4096-bit RSA key (11 traces)
• Attacks from within SGX possible
• New high-resolution timer
• Leaked 4096-bit RSA key (11 traces)
• SGX allows hiding an attack

DIMVA’17

Michael Schwarz, Samuel Weiser, Daniel Gruss, Clémentine Maurice, Stefan Mangard.
Malware Guard Extension: Using SGX to Conceal Cache Attacks.
Kernel

/dev/input/event*

Real key

Fake key
KeyDrown

Widget

Library
libgtk / libinput

Kernel
/dev/input/event*

Application Window

Real key
Fake key
In KeyDrown, the system architecture involves several components:

1. **Kernel**: This is the foundation where real and fake keys are generated.
   - Real key
   - Fake key

2. **Library**: This includes the libraries `libgtk` and `libinput`.
   - `libgtk` and `libinput` are responsible for processing input events.

3. **Widget**: This represents the graphical interface and includes:
   - Hidden Window
   - Application Window

4. **/dev/input/event***: This is a file system entry where input events are generated.

The diagram illustrates the flow of events from the kernel to the graphical interface, highlighting the distinction between real and fake keys.
Flush+Reload on libgdk-3.so

Without Keydrown, F-Score 0.99

With Keydrown, F-Score 0.09
Prime+Probe on i8042_interrupt

Without Keydrown, F-Score 0.81

With Keydrown, F-Score 0.11
Interrupt (rdtsc)

Without Keydrown, F-Score 0.94
With Keydrown, F-Score 0.14
Summary

- 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

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
• 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.
• 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

5 identified categories

Memory addresses
Landscape of Microarchitectural Attacks

11 known attacks

Memory addresses

5 identified categories

Accurate timing
Landscape of Microarchitectural Attacks

11 known attacks

5 identified categories

Memory addresses

Accurate timing

Multithreading
Landscape of Microarchitectural Attacks

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

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

11 known attacks

5 identified categories

Memory addresses

Accurate timing

Multithreading

Shared data

Sensor API

Michael Schwarz — IAIK – Graz University of Technology
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>Keystore Timing</th>
<th>Browser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffer ASLR</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>Array preloading</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Non-deterministic array</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>Array index randomization</td>
<td>○</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Low-resolution timestamp</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Fuzzy time</td>
<td>○</td>
<td>●*</td>
<td>○</td>
<td>○*</td>
<td>○</td>
<td>●*</td>
<td>●*</td>
<td>●*</td>
</tr>
<tr>
<td>WebWorker polyfill</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>Message delay</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Slow SharedArrayBuffer</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>No SharedArrayBuffer</td>
<td>○</td>
<td>●*</td>
<td>●</td>
<td>●*</td>
<td>●</td>
<td>○*</td>
<td>○*</td>
<td>○*</td>
</tr>
<tr>
<td>Summary</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>

Fully prevented (●), partly prevented (○), not prevented (□). Policies must be combined (*).
• Just rounding timers is not sufficient

NDSS’18
Michael Schwarz, Moritz Lipp, Daniel Gruss.
JavaScript Zero: Real JavaScript and Zero Side-Channel Attacks.
• Just rounding timers is not sufficient
• 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.
• Just rounding timers is not sufficient
• Microarchitectural attacks in the browser are still possible
• Efficient countermeasures can be implemented in browsers
• Just rounding timers is not sufficient
• Microarchitectural attacks in the browser are still possible
• Efficient countermeasures can be implemented in browsers
• More microarchitectural attacks in JavaScript will appear
A Double Fetch

string

\[ /path/file \quad \emptyset \quad \text{payload} \quad \emptyset \]

length

Kernel \quad User
A Double Fetch

string

```
/path/file
```

length

Kernel

User

Ø

payload Ø

30
A Double Fetch

```
string

/ path / file X payload Ø
```

Kernel

User

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

string

/path/file\texttt{X}payload\emptyset

length

Kernel

\texttt{\emptyset}

\downarrow \texttt{copy string}

User

string[10] = 'X';

overflow
Double-fetch Detection and Exploitation

Runtime [cycles]

Access time [cycles]

First access

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

![Graph showing access time vs runtime](image)

- **First access**
- **Second access**

- Access time [cycles]
- Runtime [cycles]
Double-fetch Detection and Exploitation

![Graph showing access time and runtime with labels for first access, modify value, and second access with modified value.](image-url)
(Syscall) Fuzzer

Report general bug
DECAF

(Syscall) Fuzzer

DECAF

Report
general bug
DECAF

(Syscall) Fuzzer

DECAF

Detect double fetches

Report
general bug
DECAF

(Syscall) Fuzzer

DECAF

Detect double fetches

Double fetch candidates

Report general bug
DECAF

(Syscall) Fuzzer

Detect double fetches

Double fetch candidates

Report general bug

Exploit double fetch
DECAF

(Syscall) Fuzzer

DECAF

Exploit double fetch

Detect double fetches

Double fetch candidates

Report general bug

Report double-fetch bug
DECAF (Syscall) Fuzzer

DECAF

Exploit double fetch

Detect double fetches

Double fetch candidates

Report general bug

Fix double-fetch bug

Report double-fetch bug

32

Michael Schwarz — IAIK – Graz University of Technology
Summary

• Fuzzing + cache attacks $\rightarrow$ double-fetch bugs
• Fuzzing + cache attacks $\rightarrow$ double-fetch bugs
• Cache as trigger outperforms state of the art
• 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
JavaScript Template Attacks

- Template attack on JavaScript properties

Profiling Phase

Collect #1

Explore

Template
• Template attack on JavaScript properties
• Template attack on JavaScript properties
• Template attack on JavaScript properties
JavaScript Template Attacks

- Template attack on JavaScript properties
• Template attack on JavaScript properties
JavaScript Template Attacks

- Template attack on JavaScript properties
Detection

Browser + version
Detection

Browser + version

Privacy extensions
Detection

- Browser + version
- Privacy extensions
- Private mode
Detection

- Browser + version
- Privacy extensions
- Private mode
- Operating system
Detection

- Browser + version
- Privacy extensions
- Private mode
- Operating system
- CPU vendor

Michael Schwarz — IAIK – Graz University of Technology
Detection

- Browser + version
- Privacy extensions
- Private mode
- Operating system
- CPU vendor
- Virtual machine
• JavaScript Template attacks detect environment properties
Summary

• JavaScript Template attacks detect environment properties
• Enables exploits, side-channel attacks and plausible phishing

NDSS’19

Michael Schwarz, Florian Lackner, Daniel Gruss.
• 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.
• 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
**Fill-buffer Leakage**

**User Memory**

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

```plaintext
char value = kernel[0]
```

![Diagram of a processor or memory layout]
User Memory

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

```
char value = kernel[0]
```

```c
mem[value]
```

Out of order

Michael Schwarz — IAIK – Graz University of Technology
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 |
| 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 = kernel[0]

Out of order

mem[value]
Results

AES-NI key
Results

AES-NI key

SGX sealing key

Michael Schwarz — IAIK – Graz University of Technology
Results

- AES-NI key
- SGX sealing key
- Cross-VM covert channel
Results

- AES-NI key
- SGX sealing key
- Cross-VM covert channel
- Keyword matching
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
• Meltdown on the line-fill buffer
• Data sampling attacks are powerful
• Just the beginning of CPU vulnerabilities

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
• Often wrong assumptions on requirements
• Attack research is necessary for effective defenses
• Often wrong assumptions on requirements
• Data-dependent optimizations often lead to side channel
• Attack research is necessary for effective defenses
• Often wrong assumptions on requirements
• Data-dependent optimizations often lead to side channel
• Trade-off between performance and security
Software-based Side-Channel Attacks and Defenses in Restricted Environments

PhD Defense

Michael Schwarz
IAIK, Graz University of Technology
November 5, 2019