Faulty Point Unit: ABI Poisoning Attacks on Intel SGX

Fritz Alder¹, Jo Van Bulck¹, David Oswald², Frank Piessens¹
¹imec-DistriNet, KU Leuven, Belgium ²The University of Birmingham, UK
December 10, 2020
The promise of Trusted Execution Environments
The promise of Trusted Execution Environments
The promise of Trusted Execution Environments
Trusted Execution Environments: Enclave calls
Trusted Execution Environments: Enclave calls

EENTER
Trusted Execution Environments: Enclave calls

EENTER enclaves shielding runtime

Faulty Point Unit: ABI Poisoning Attacks on Intel SGX
Key insight: split sanitization responsibilities across the ABI and API tiers: machine state vs. higher-level programming language interface
x87 Floating Point Unit (FPU) and Streaming SIMD Extensions (SSE)

- Older x87 high-precision floating-point unit: FPU control word
- Newer SSE vector floating-point operations: MXCSR register
x87 Floating Point Unit (FPU) and Streaming SIMD Extensions (SSE)

- Older x87 high-precision floating-point unit: FPU control word
- Newer SSE vector floating-point operations: MXCSR register

The control bits of the MXCSR register are callee-saved (preserved across calls), while the status bits are caller-saved (not preserved). The x87 status word register is caller-saved, whereas the x87 control word is callee-saved.
FPU settings are preserved across calls

```
enclave_func:
long double weight = 2.1 * 3.4;
```
Controlling FPU precision and rounding modes  

FPU settings are preserved across calls

enclave_func:

```c
long double weight = 2.1 * 3.4;
```

weight: 7.14
Controlling FPU precision and rounding modes

CVE-2020-0561

Corrupt precision and rounding mode...

enclave_func:

\[ \text{long double } weight = 2.1 \times 3.4; \]

FPU_CW = 0x43F
Controlling FPU precision and rounding modes

CVE-2020-0561

Corrupt precision and rounding mode...

enclave_func:
long double weight = 2.1 * 3.4;

RFLAGS.DF = 0

weight: FPU_CW = 0x43F

EENTER

7.1399998664855957031250000

Faulty Point Unit: ABI Poisoning Attacks on Intel SGX
Controlling FPU precision and rounding modes  

<table>
<thead>
<tr>
<th>Exploit Patch</th>
<th>SGX-SDK*</th>
<th>OpenEnclave</th>
<th>Graphene</th>
<th>SGX-LKL</th>
<th>Rust-EDP</th>
<th>Go-TEE</th>
<th>Enarx</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>⭐</td>
<td>○</td>
<td>○</td>
<td>⭐</td>
<td>⭐</td>
<td>⭐</td>
<td>○</td>
</tr>
<tr>
<td>xorstor</td>
<td>ldmxcsw/cw</td>
<td>fxrstor</td>
<td>−</td>
<td>ldmxcsw/cw</td>
<td>xorstor</td>
<td>xorstor</td>
<td></td>
</tr>
</tbody>
</table>

* Includes derived runtimes such as Baidu’s Rust-SGX and Google’s Asylo.
Fill data registers to fault calculations

CVE-2020-15107

Mark data registers as in-use before entering the enclave

![Diagram showing EENTER function and enclave_func with its code snippet:]

```
enclave_func:
  long double weight = 2.1 * 3.4;
```
Fill data registers to fault calculations

Mark data registers as in-use before entering the enclave

enclave_func:

```c
long double weight = 2.1 * 3.4;
```

![Faulty Point Unit: ABI Poisoning Attacks on Intel SGX](image)
Summary: ABI-level FPU attack surface today

<table>
<thead>
<tr>
<th>Exploit Patch 1</th>
<th>SGX-SDK*</th>
<th>OpenEnclave</th>
<th>Graphene</th>
<th>SGX-LKL</th>
<th>Rust-EDP</th>
<th>Go-TEE</th>
<th>Enarx</th>
</tr>
</thead>
<tbody>
<tr>
<td>xrstor</td>
<td>★</td>
<td>✔</td>
<td>✔</td>
<td>★</td>
<td>★</td>
<td>★</td>
<td>○</td>
</tr>
<tr>
<td>ldmxcsr/cw</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>⊗</td>
<td>⊗</td>
<td>⊗</td>
<td>⊗</td>
</tr>
<tr>
<td>xrstor</td>
<td>★</td>
<td>★</td>
<td>★</td>
<td>★</td>
<td>★</td>
<td>★</td>
<td>⊗</td>
</tr>
</tbody>
</table>

Patch 2

<table>
<thead>
<tr>
<th>Exploit Patch 2</th>
<th>SGX-SDK*</th>
<th>OpenEnclave</th>
<th>Graphene</th>
<th>SGX-LKL</th>
<th>Rust-EDP</th>
<th>Go-TEE</th>
<th>Enarx</th>
</tr>
</thead>
<tbody>
<tr>
<td>xrstor</td>
<td>★</td>
<td>✔</td>
<td>★</td>
<td>★</td>
<td>★</td>
<td>★</td>
<td>⊗</td>
</tr>
</tbody>
</table>

* Includes derived runtimes such as Baidu’s Rust-SGX and Google’s Asylo.
Case study 1: Floating-point exceptions as a side channel

Can we use overflows as a side channel to deduce secrets?

```
long double input

enclave_func:

a = input * secret;
```
Case study 1: Floating-point exceptions as a side channel

Can we use overflows as a side channel to deduce secrets?

enclave_func:

\[ a = \text{input} \times \text{secret}; \]

mask exceptions, register signal handler

Overflow exception

0xffffffff
Case study 1: Floating-point exceptions as a side channel

Binary search with deterministic number of steps retrieves secret

![Graph showing error distribution](image-url)
Case study 2: MNIST – ML handwriting recognition
Case study 2: MNIST – ML as an SGX Service

User 1

push model

Machine Learning Engine

User 2

push input

receive prediction
Case study 2: MNIST – ML as an SGX Service

User 1
- push model

Machine Learning Engine
- push input
- receive prediction

Enclave

Poison FPU register

User 2
- Poison FPU register
### Case study 2: MNIST – Predictions of 100 digits

<table>
<thead>
<tr>
<th>Extended precision</th>
<th>Predicted digit count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rounding mode</td>
<td>Correct</td>
</tr>
<tr>
<td>Any mode</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>0 1 2 3 4 5 6 7 8 9</td>
</tr>
<tr>
<td></td>
<td>9 14 8 10 14 8 9 14 3 11</td>
</tr>
</tbody>
</table>

**x87 Extended precision: Default predictions**
Case study 2: MNIST – Predictions of 100 digits

<table>
<thead>
<tr>
<th><strong>Extended precision</strong></th>
<th>Predicted digit count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rounding mode</td>
<td>Correct</td>
</tr>
<tr>
<td>Any mode</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>9 14 8 10 14 8 9 14 3 11</td>
</tr>
</tbody>
</table>

x87 Extended precision: Default predictions

<table>
<thead>
<tr>
<th><strong>Single precision</strong></th>
<th>Predicted digit count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rounding mode</td>
<td>Correct</td>
</tr>
<tr>
<td>Rounding down</td>
<td>8%</td>
</tr>
<tr>
<td></td>
<td>0 0 100 0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

x87 Single precision: Attacked predictions
Case study 3: SPEC 2017. Image difference in Blender
Washes away Bacteria
Frequent hand washing helps keep your family healthy.
Conclusions and outlook

Secure enclave interactions require proper sanitizations!
Conclusions and outlook

Secure enclave interactions require proper sanitizations!

- Large attack surface, including subtle side-channel oversights...
- **Defense:** Most investigated shielding runtimes now apply a full XRSTOR sanitization strategy
- Modern x86 architectures are **complex**. Need to investigate alternative processor architectures such as RISC-V

🔗 https://github.com/fritzalder/faulty-point-unit
Faulty Point Unit: ABI Poisoning Attacks on Intel SGX

Fritz Alder¹, Jo Van Bulck¹, David Oswald², Frank Piessens¹
¹imec-DistriNet, KU Leuven, Belgium ²The University of Birmingham, UK
December 10, 2020