AEX-Notify: Thwarting Precise Single-Stepping Attacks through Interrupt Awareness for Intel® SGX Enclaves

Scott Constable¹, Jo Van Bulck², Xiang Cheng³, Yuan Xiao¹, Cedric Xing¹, Ilya Alexandrovich¹, Taesoo Kim³, Frank Piessens², Mona Vij¹, Mark Silberstein⁴

32nd USENIX Security Symposium, August 10, 2023

¹Intel Corporation ²imec-DistriNet, KU Leuven ³Georgia Institute of Technology ⁴Technion
Part I: Problem statement
Enclaved execution: Reducing attack surface

Intel® SGX: Hardware-level isolation and attestation
Enclaved execution: Privileged side channels

Game changer: Untrusted OS $\rightarrow$ new class of powerful side channels!
Challenge: Side-channel sampling rate

- Fast shutter speed
- Medium shutter speed
- Slow shutter speed
SGX-Step: Executing enclaves one instruction at a time
SGX-Step: Executing enclaves one instruction at a time

[GitHub Repository Link]

https://github.com/jovanbulck/sgx-step
SGX-Step: Executing enclaves one instruction at a time

```c
if secret do
    inst1
else
    inst2
endif
```

Interrupt handler

Attacker

Enclave

ERELEASE

user space

OS kernel

/dev/sgx-step
SGX-Step: Enabling a line of high-resolution attacks

1. Interrupt latency
   [CCS'18, USENIX'21]

2. Interrupt counting
   [CCS'19, CHES'20-21, USENIX'20]

3. Zero-step replaying
   [USENIX'18, CCS'19, ISCA'19, S&P'21]

4. Amplification
   [ATC'17, CCS'19/21, CHES'17-19, S&P'20-21, USENIX'17/18/22]
SGX-Step demo: Building a `memcmp` password oracle

```
jo@breuer:~/sgx-step-demo$ sudo ./app
```
Root-causing SGX-Step: Aiming the timer interrupt

APIC timer oneshot TSC distribution

Arm timer ERESUME NOP1 NOP2 NOP3 RIP... RIP

Zero-step Single-step Multi-step
Root-causing SGX-Step: Microcode assists to the rescue!

1. Clear PTE A-bit
2. TLB flush
3. Assisted PT walk

<table>
<thead>
<tr>
<th>PTE A-bit</th>
<th>Mean (cycles)</th>
<th>Stddev (cycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A=1</td>
<td>27</td>
<td>30</td>
</tr>
<tr>
<td>A=0</td>
<td>666</td>
<td>55</td>
</tr>
</tbody>
</table>

```mermaid
graph TD
    page walk ($RIP) --> exec[exec]
```
Root-causing SGX-Step: Microcode assists to the rescue!

1. Clear PTE A-bit
2. TLB flush
3. Assisted PT walk

Arm timer  ERESUME  NOP₁
Root-causing SGX-Step: Microcode assists to the rescue!

1. Clear PTE A-bit
2. TLB flush
3. Assisted PT walk
4. Filter zero-step (PTE A-bit)
Part II: Solution overview
Ideas that were rejected (1)

Enclave

Enter

Exit

Disable A/D-bit assists

Enable A/D-bit assists

Breaks the OS/User contract
Ideas that were rejected (2)

Arm timer

ERESUME

What if...?

NOP_1
NOP_2
NOP_3
NOP_4
NOP_5
...

Highly complex
Ideas that were rejected (3)

Enclave

Enclave’s Page
Tables

What if...?

Enclave’s Page
Tables

Highly complex
Ideas that were rejected (3)

Enclave's Page Tables

Virtual Machine Monitor (VMM)

Extended Page Tables

Guest physical address

Host physical address

Virtual Machine

Memory
AEX-Notify solution overview

Enclave

Enclave App

Legacy enclave behavior

Interrupt or Exception

Attacker

ERESUME
AEX-Notify solution overview

Legend:
- AEX-Notify ISA Extension

AEX-Notify behavior

Interrupt or Exception

1. Call a C3 byte on .page1
2. Load all cache lines in .page1
3. JMP [&NOP₁]

ERESUME

AEX Handler

Enclave App

page walk (.page1)

exec

AEX

Enclave

Enclave App

EDECCSSA

AEX Handler

ERESUME

Attacker

NOP₁

RET # (C3 byte)
We implemented a fast, constant-time decoder (CTD)

AEX Handler
1. Decode the saved [RIP]
2. Read and write back to [RAX]
3. ...

Enclave App
...INC [RAX]
...RET # (C3 byte)

ERESUME

CTD Instruction Coverage for popular SGX runtimes

<table>
<thead>
<tr>
<th>SGX Runtime (# of binaries analyzed)</th>
<th>Covered w/ CTD</th>
<th>Covered w/o CTD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel SGX SDK (18)</td>
<td>98.6%</td>
<td>37.4%</td>
</tr>
<tr>
<td>Gramine (53)</td>
<td>97.5%</td>
<td>26.5%</td>
</tr>
<tr>
<td>Occlum (35)</td>
<td>98.1%</td>
<td>35.1%</td>
</tr>
<tr>
<td>Total (106)</td>
<td>98.0%</td>
<td>32.0%</td>
</tr>
</tbody>
</table>

Covered w/ CTD
Covered w/o CTD

We implemented a fast, constant-time decoder (CTD)
Experiments were conducted on an Intel Coffee Lake processor-based Xeon E3 platform
Real-world performance impact depends on **AEX frequency**...

<table>
<thead>
<tr>
<th></th>
<th>With Mitigation</th>
<th>Without Mitigation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resuming an enclave thread</td>
<td>58% slowdown (6,500 → 10,300 cycles)</td>
<td>Performance unaffected</td>
</tr>
<tr>
<td>Handling an exception within an enclave</td>
<td>76% speedup</td>
<td>88% speedup</td>
</tr>
</tbody>
</table>

If the enclave is interrupted every 1 million cycles, the overhead is:

\[
\frac{10,300 - 6,500}{1,000,000} = 0.38\% 
\]
Conclusions

Extensible AEX-Notify hardware-software co-design

Eliminate root cause of perfect single/zero stepping

Minimal performance overhead and fast CTD

Thank you! Questions?