SGX-Step: An Open-Source Framework for Precise Dissection and Practical Exploitation of Intel SGX Enclaves

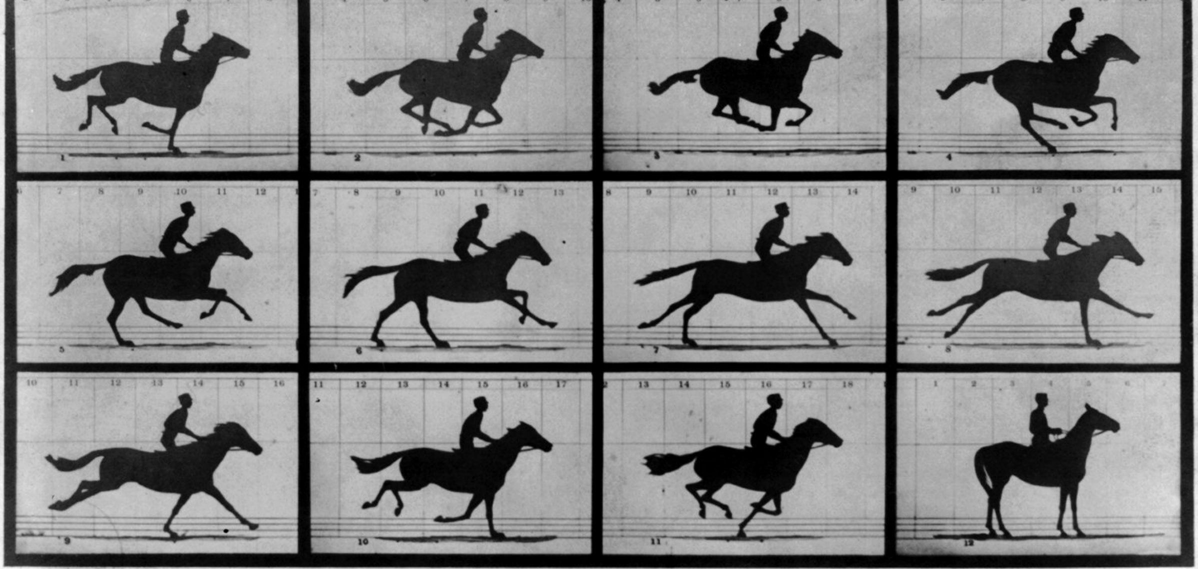
Jo Van Bulck, Frank Piessens

Cybersecurity Artifacts Competition and Impact Award Finalist
ACSAC 2023, December 7, 2023

DistriNet, KU Leuven, Belgium jo.vanbulck@cs.kuleuven.be jovanbulck

DistriNet
KU LEUVEN

DALL·E 3



MORSE'S Gallery, 417 Montgomery St., San Francisco.

THE HORSE IN MOTION.

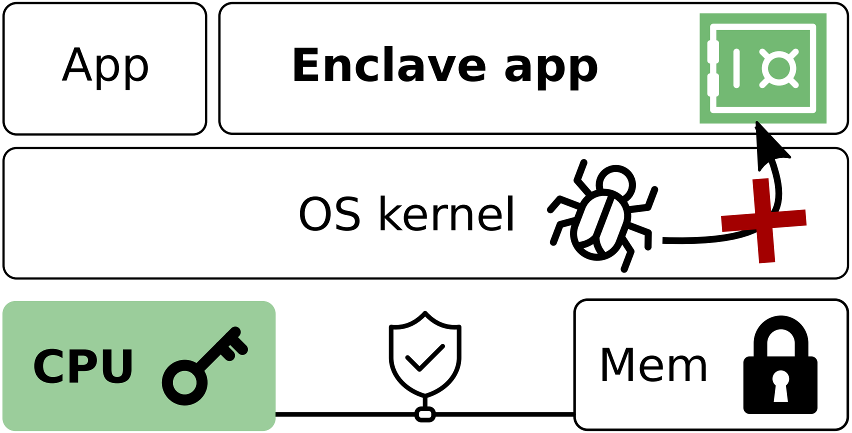
Illustrated by

MUYBRIDGE.

AUTOMATIC ELECTRO-PHOTOGRAPH.

"SALLIE GARDNER," owned by LELAND STANFORD; running at a 1.40 gait over the Palo Alto track, 19th June, 1878.

From Horses to Enclaves: Reducing Attack Surface



Intel SGX promise: Hardware-level **isolation and attestation**

From Horses to Enclaves: Reducing Attack Surface

Game changer: Untrusted OS → new class of powerful **side channels!**

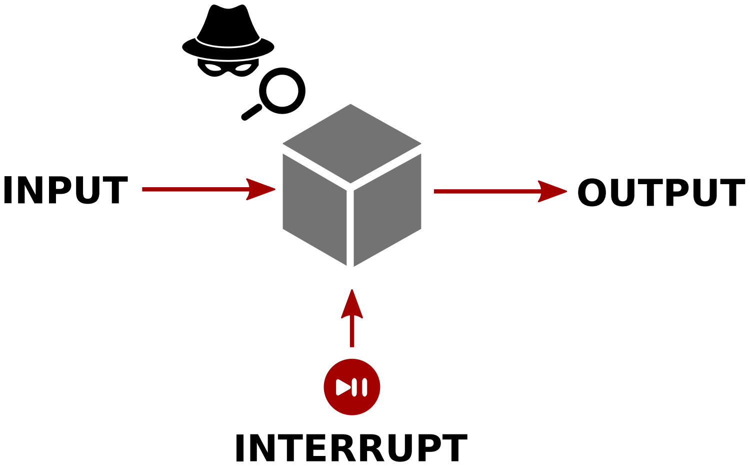
Challenge: Side-channel Sampling Rate

Slow
shutter speed

Medium
shutter speed

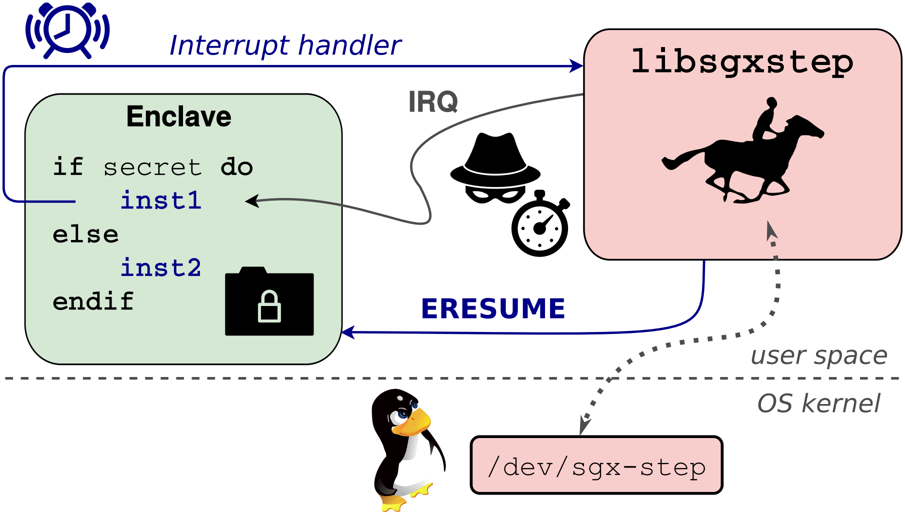
Fast
shutter speed

SGX-Step: Executing Enclaves one Instruction at a Time



🗋 Van Bulck et al., "SGX-Step: A Practical Attack Framework for Precise Enclave Execution Control", SysTEX 2017.

SGX-Step: Executing Enclaves one Instruction at a Time



Van Bulck et al., "SGX-Step: A Practical Attack Framework for Precise Enclave Execution Control", SysTEX 2017.

A Retrospective of 5 Years of SGX-Step Development

SGX-Step

https://github.com/jovanbulck/sgx-step

Unwatch 27 | Fork 82 | Star 402

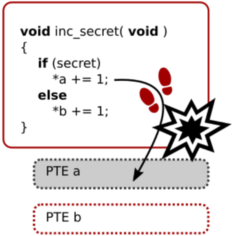
- Became **de-facto standard** for interrupt-driven attacks
- Actively maintained & supported
- Widely recognized:
  - > 400 GitHub stars
  - > 215 academic citations
- Marked influence on both **attacks & defenses** on SGX and beyond

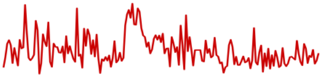
Highlight #1: Impact on Attacks

SGX-Step: Enabling a New Line of High-Resolution Attacks

Yr	Venue	Paper	Step	Use Case	Drv
'15	S&P	Ctrl channel [XCP15]	~ Page	*Probe* (page fault)	✓
'16	ESORICS	AsyncShock [WKPK16]	~ Page	*Exploit* (mem safety)	–
'17	CHES	CacheZoom [MIE17]	✗ >1	*Probe* (L1 cache)	✓
'17	ATC	Hahnel et al. [HCP17]	✗ 0 - >1	*Probe* (L1 cache)	✓
'17	USENIX	BranchShadow [LSG+17]	✗ 5 - 50	*Probe* (BPU)	✗
'17	USENIX	Stealthy PTE [VBWK+17]	~ Page	*Probe* (page table)	✓
'17	USENIX	DarkROP [LJJ+17]	~ Page	*Exploit* (mem safety)	✓
'17	SysTEX	SGX-Step [VBPS17]	✓ 0 - 1	*Framework*	✓
'18	ESSoS	Off-limits [GVBPS18]	✓ 0 - 1	*Probe* (segmentation)	✓
'18	AsiaCCS	Single-trace RSA [WSB18]	~ Page	*Probe* (page fault)	✓
'18	USENIX	Foreshadow [VBMW+18]	✓ 0 - 1	*Probe* (transient exec)	✓
'18	EuroS&P	SgxPectre [CCX+19]	~ Page	*Exploit* (transient)	✓
'18	CHES	CacheQuote [DDME+18]	✗ >1	*Probe* (L1 cache)	✓
'18	ICCD	SGXlinger [HZDL18]	✗ >1	*Probe* (IRQ latency)	✗
'18	CCS	Nemesis [VBPS18]	✓ 1	*Probe* (IRQ latency)	✓
'19	USENIX	Spoiler [IMB+19]	✓ 1	*Probe* (IRQ latency)	✓
'19	CCS	ZombieLoad [SLM+19]	✓ 0 - 1	*Probe* (transient exec)	✓
'19	CCS	Fallout [CGG+19]	–	*Probe* (transient exec)	✓
'19	CCS	Tale of 2 worlds [VBOM+19]	✓ 1	*Exploit* (mem safety)	✓
'19	ISCA	MicroScope [SYG+19]	~ 0 - Page	*Framework*	✗
'20	CHES	Bluethunder [HMW+20]	✓ 1	*Probe* (BPU)	✓
'20	USENIX	Big troubles [WSBS19]	~ Page	*Probe* (page fault)	✓
'20	S&P	Plundervolt [MOG+20]	–	*Exploit* (undervolt)	✓
'20	CHES	Viral primitive [AB20]	✓ 1	*Probe* (IRQ count)	✓
'20	USENIX	CopyCat [MVBH+20]	✓ 1	*Probe* (IRQ count)	✓
'20	S&P	LVI [VBMS+20]	✓ 1	*Exploit* (transient)	✓
'20	CHES	A to Z [AGB20]	~ Page	*Probe* (page fault)	✓
'20	CCS	Déjà Vu NSS [uHGDL+20]	~ Page	*Probe* (page fault)	✓
'20	MICRO	PTHammer [ZCL+20]	–	*Probe* (page walk)	✓
'21	USENIX	Frontal [PSHC21]	✓ 1	*Probe* (IRQ latency)	✓
'21	S&P	CrossTalk [RMR+21]	✓ 1	*Probe* (transient exec)	✓
'21	CHES	Online template [AB21]	✓ 1	*Probe* (IRQ count)	✓
'21	NDSS	SpeechMiner [XZT20]	–	*Framework*	✓
'21	S&P	Platypus [LKO+21]	✓ 0 - 1	*Probe* (voltage)	✓
'21	DIMVA	Aion [HXCL21]	✓ 1	*Probe* (cache)	✓
'21	CCS	SmashEx [CYS+21]	✓ 1	*Exploit* (mem safety)	✓
'21	CCS	Util::Lookup [SBWE21]	✓ 1	*Probe* (L3 cache)	✓
'22	USENIX	Rapid prototyping [ESSG22]	✓ 1	*Framework*	✓
'22	CT-RSA	Kalyna expansion [CGYZ22]	✓ 1	*Probe* (L3 cache)	✓
'22	SEED	Enclyzer [ZXTZ22]	–	*Framework*	✓
'22	NordSec	Self-monitoring [LBA22]	~ Page	*Defense* (detect)	✓
'22	AutoSec	Robotic vehicles [LS22]	✓ 1 - >1	*Exploit* (timestamp)	✓
'22	ACSAC	MoLE [LWM+22]	✓ 1	*Defense* (randomize)	✓
'22	USENIX	AEPIC [BKS+22]	✓ 1	*Probe* (I/O device)	✓
'22	arXiv	Confidential code [PSL+22]	✓ 1	*Probe* (IRQ latency)	✓
'23	ComSec	FaultMorse [HZL+23]	~ Page	*Probe* (page fault)	✓
'23	CHES	HQC timing [HSC+23]	✓ 1	*Probe* (L3 cache)	✓
'23	ISCA	Belong to us [YJF23]	✓ 1	*Probe* (BPU)	✓
'23	USENIX	BunnyHop [ZTO+23]	✓ 1	*Probe* (BPU)	✓
'23	USENIX	DownFall [Mog23]	✓ 0 - 1	*Probe* (transient exec)	✓
'23	USENIX	AEX-Notify [CVBC+23]	✓ 1	*Defense* (prefetch)	✓

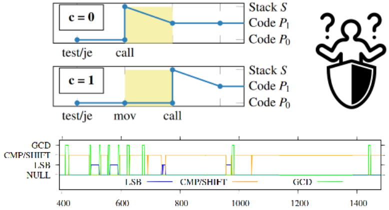
A Versatile Open-Source Attack Toolkit





Interrupt latency

[CCS'18, USENIX'21]



Page-table manipulation

[AsiaCCS'18, USENIX'18-23, CCS20, CHES'20, NDSS'21]

SGX-Step

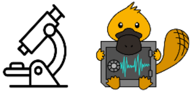
Interrupt counting

[CCS'19, CHES'20-21, USENIX'20]

High-resolution probing

[CCS'19/21, CHES'20, S&P'20-21, USENIX'17/18/22]

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



SGX-Step demo: Building a `memcmp()` Password Oracle

```
[idt.c] DTR.base=0xfffffe0000000000/size=4095 (256 entries)
[idt.c] established user space IDT mapping at 0x7f7ff8e9a000
[idt.c] installed asm IRQ handler at 10:0x56312d19b000
[idt.c] IDT[ 45] @0x7f7ff8e9a2d0 = 0x56312d19b000 (seg sel 0x10); p=1; dpl=3; type=14; ist=0
[file.c] reading buffer from '/dev/cpu/1/msr' (size=8)
[apic.c] established local memory mapping for APIC_BASE=0xfee00000 at 0x7f7ff8e99000
[apic.c] APIC_ID=2000000; LVTT=400ec; TDCR=0
[apic.c] APIC timer one-shot mode with division 2 (lvtt=2d/tdcr=0)

--------------------------------------------------------------------------------
[main.c] recovering password length
--------------------------------------------------------------------------------

[attacker] steps=15; guess='******'
[attacker] found pwd len = 6

--------------------------------------------------------------------------------
[main.c] recovering password bytes
--------------------------------------------------------------------------------

[attacker] steps=35; guess='SECRET' --> SUCCESS

[apic.c] Restored APIC_LVTT=400ec/TDCR=0)
[file.c] writing buffer to '/dev/cpu/1/msr' (size=8)
[main.c] all done; counted 2260/2183 IRQs (AEP/IDT)
jo@breuer:~/sgx-step-demo$
```

Highlight #2: Impact on Defenses

Hardening Enclaves against Single-Stepping

SGX-Step sets the **bar for adequate side-channel defenses!**

→ (e.g., LVI, compiler, static analysis, constant-time, etc.)

"ineffective if the attacker can single-step through the enclave using the recent SGX-Step framework. Taking into account these stronger attacker capabilities, we propose a new defense. . . " [HLLP18]

SGX-Step inspired several dedicated **hardware-software mitigations**

→ Collaboration with Intel on AEX-Notify: Innovative hardware-software co-design included in recent processors

→ Probabilistic: SGX-Step remains relevant!

CHAPTER 8
ASYNCHRONOUS ENCLAVE EXIT NOTIFY AND THE EDECCSSA USER LEAF FUNCTION

8.1 INTRODUCTION

Asynchronous Enclave Exit Notify (AEX-Notify) is an extension to Intel® SGX that allows Intel SGX enclaves to be notified after an asynchronous enclave exit (AEX) has occurred. EDECCSSA is a new Intel SGX user leaf function (ENCLU[EDECCSSA]) that can facilitate AEX notification handling, as well as software exception handling. This chapter provides information about changes to the Intel SGX architecture that support AEX-Notify and ENCLU[EDECCSSA].

The following list summarizes the add
details are provided in Section 8.3):

- SECS.ATTRIBUTES.AEXNOTIFY: Th
- TCS.FLAGS.AEXNOTIFY: This enclave thread may receive AEX notifications.
- SSA.GPRSGX.AEXNOTIFY: Enclave-writable byte that allows enclave software to dynamically enable/disable AEX notifications.

An AEX notification is delivered by ENCLU[ERESUME] when the following conditions are met:

SGX-Step led to **new x86 processor instructions!**

→ *shipped in millions of devices ≥ 4th Gen Xeon [CVBC+23]* phoronix

ARTICLES & REVIEWS | NEWS ARCHIVE | FORUMS | PREMIUM | CONTACT | CATEGORIES

Intel AEX Notify Support Prepped For Linux To Help Enhance SGX Enclave Security

Written by Michael Larabel in Intel on 6 November 2022 at 06:01 AM EST. 5 Comments

Future Intel CPUs and some existing processors via a microcode update will support a new feature called the Asynchronous EXit (AEX) notification mechanism to help with Software Guard Extensions (SGX) enclave security. Patches for the Linux kernel are pending for implementing this Intel AEX Notify support with capable processors.

Intel's Asynchronous EXit (AEX) notification mechanism lets SGX enclaves run a handler after an AEX event. Those handlers can be used for things like mitigating SGX-Step as an attack framework for precise enclave execution control.

Code 1 in intel/linux-sgx Filter

sdk/trts/linux/trts_mitigation.S

```
 * Description:
 *     The file provides mitigations for SGX-Step
 */
 *  Function:
constant_time_apply_sgxstep_mitigation_and_continue_execution
 *      Mitigate SGX-Step and return to the point at which the
most recent
 *      interrupt/exception occurred.
```

SGX-Step led to **changes in major OSs and enclave SDKs**

Beyond SGX-Step: Derived Frameworks for Emerging TEEs

SGX-Step has inspired similar single-stepping frameworks for alternative TEEs

→ e.g., AMD SEV, intel TDX, arm TrustZone

Independent testimonies on SGX-Step's impact

- *"In the hope that the framework inspires a similar community as SGX-Step, we dubbed it SEV-Step." [WWRE23]*
- *"Leveraging SGX-Step type attack to compromise Intel TDX, which is coined as TDX-Step [. . .] Working exploit well within the timeline but also collaborated closely with the Intel TDX architecture team to review and refine the mitigation for the vulnerability." [Int23]*

Conclusions and Outlook

Paradigm shift: Extremely high-resolution enclave attacks

Hardware-software **mitigations** for new and emerging TEEs

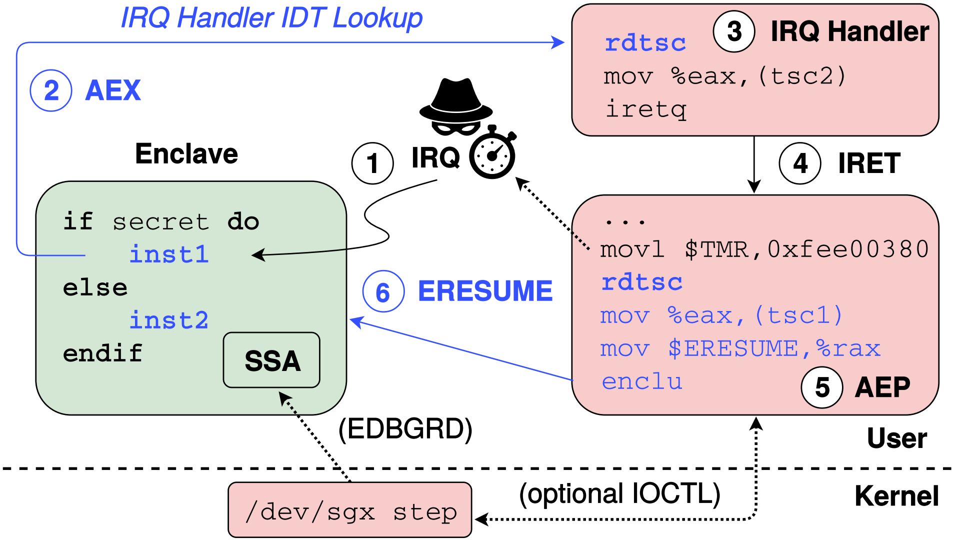
Open-source attack framework sets the bar for defenses!

SGX-Step

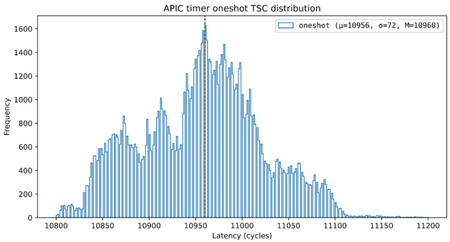
Thank you! Questions?

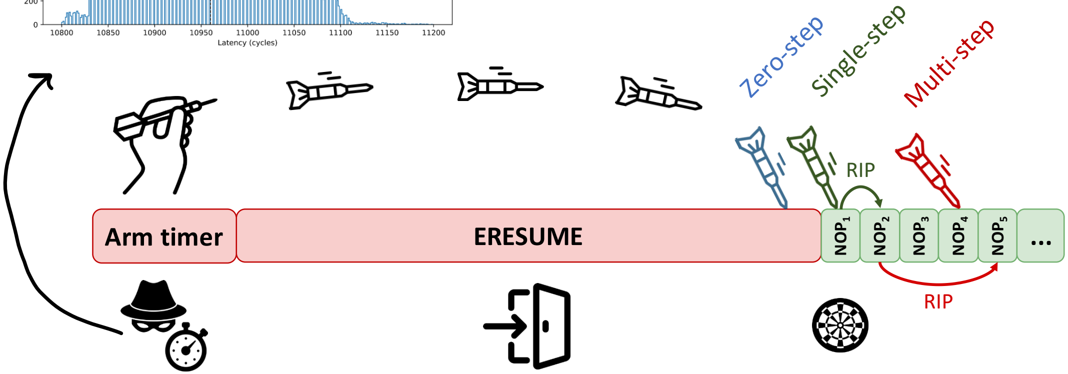
Appendix

Interrupting and Resuming SGX Enclaves



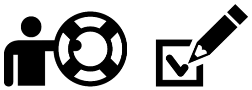
Root-causing SGX-Step: Aiming the timer interrupt

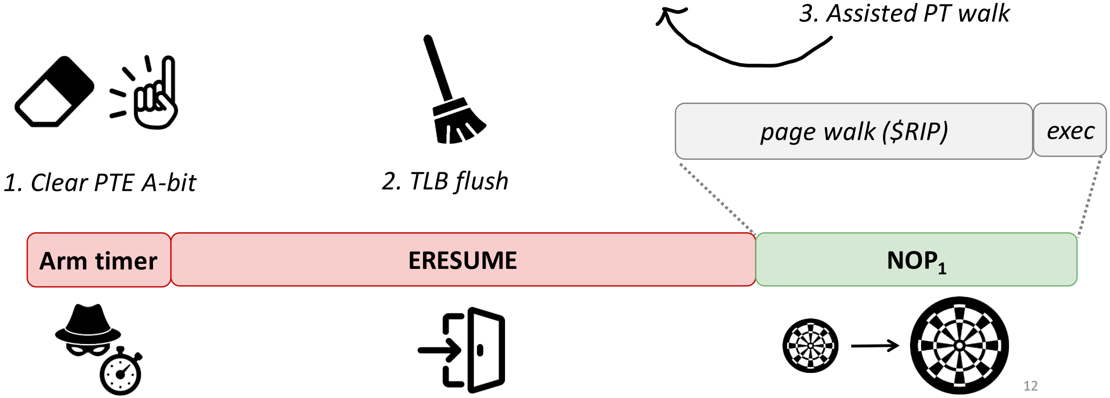




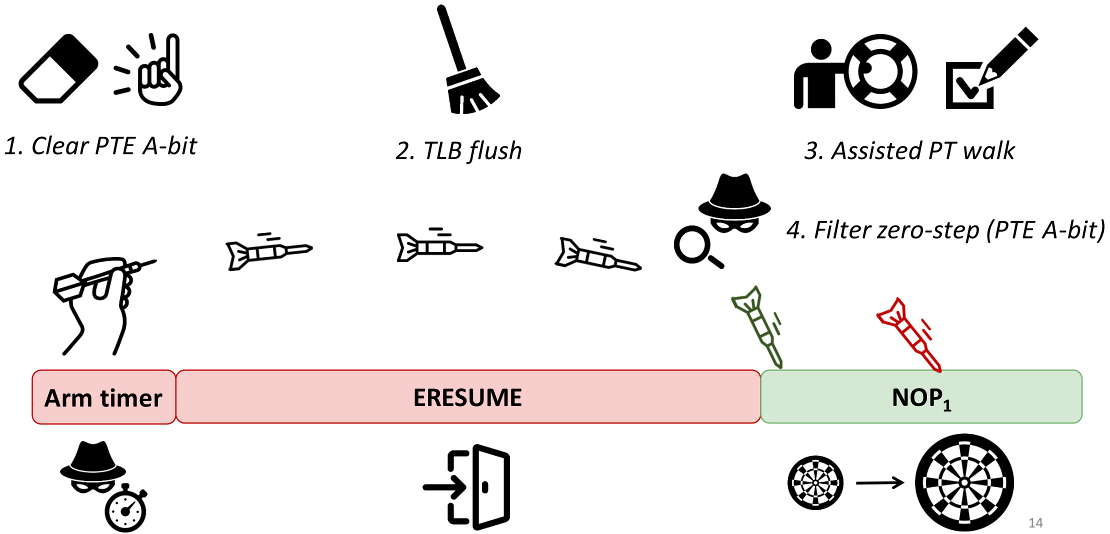
Root-causing SGX-Step: Microcode assists to the rescue!

PTE A-bit	Mean (cycles)	Stddev (cycles)
A=1	27	30
A=0	666	55

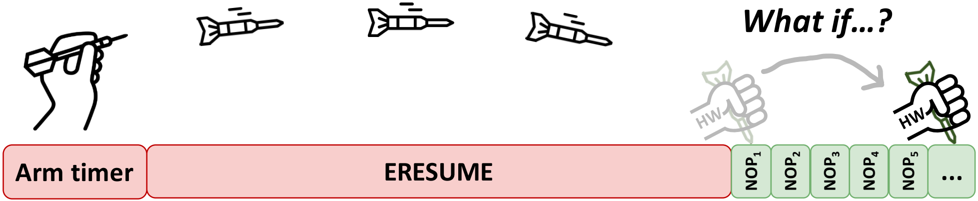


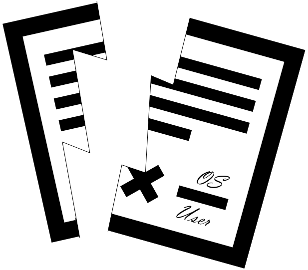


Root-causing SGX-Step: Microcode assists to the rescue!

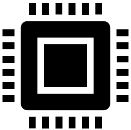


Ideas that were rejected (2)

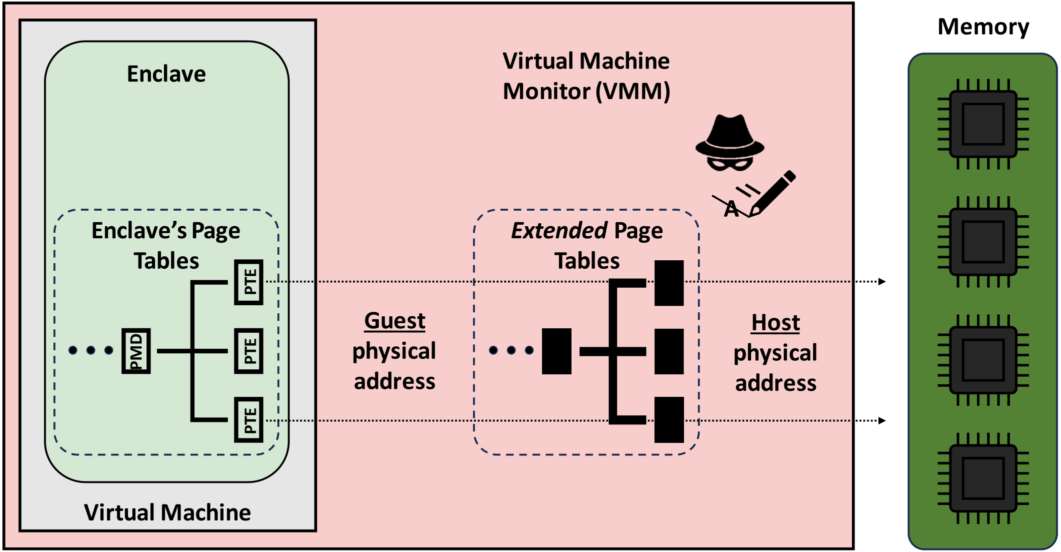




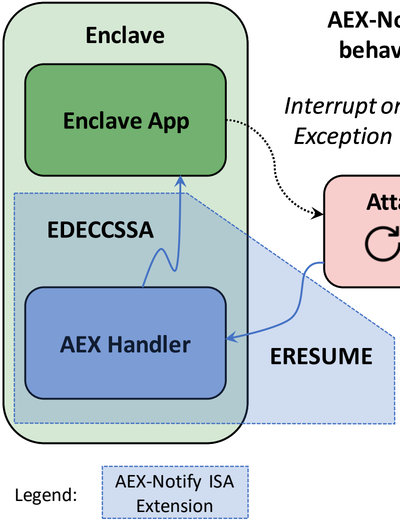
Highly complex

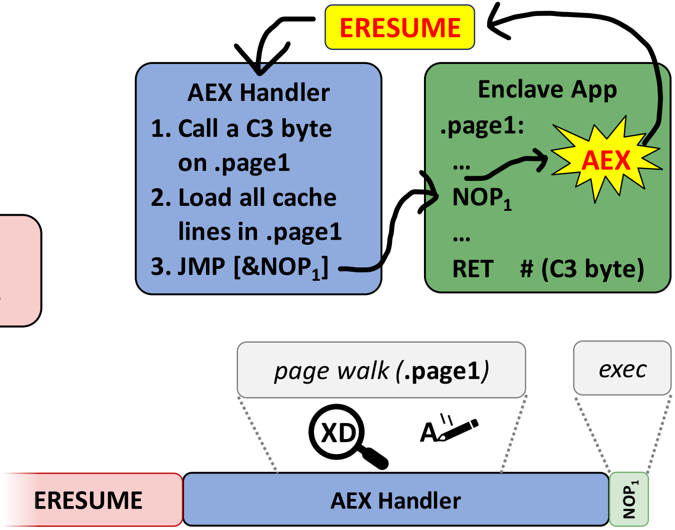


Ideas that were rejected (3)



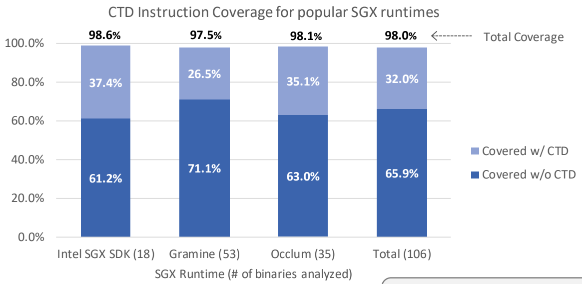
AEX-Notify solution overview

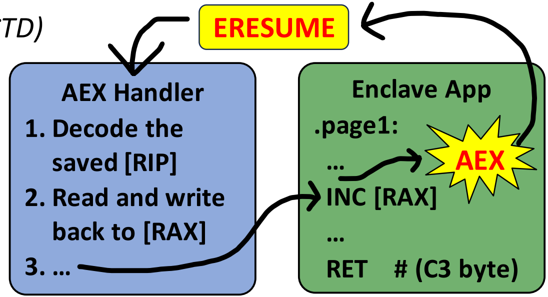


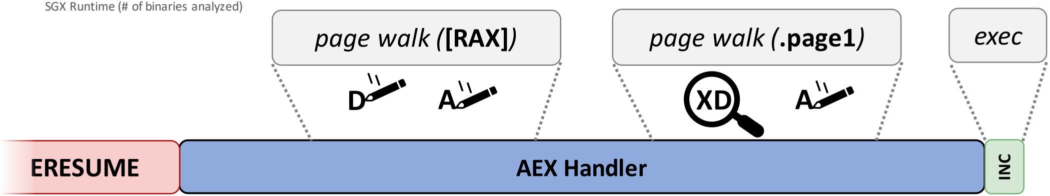


AEX-Notify solution overview

We implemented a fast, constant-time decoder (CTD)







TECHNIQUES AND TECHNOLOGIES TO ADDRESS MALICIOUS SINGLE-STEPPING AND ZERO-STEPPING OF TRUSTED EXECUTION ENVIRONMENTS

TECHNICAL FIELD

[0001] The disclosure relates generally to electronics, and, more specifically, an embodiment of the disclosure relates to techniques and technologies to address malicious single-stepping and zero-stepping of trusted execution environments (TEEs).

BACKGROUND

[0002] Trusted Execution Environments (TEEs), such as Intel® Software Guard Extensions (Intel® SGX), are susceptible to methods that induce interrupts or exceptions to maliciously single-step (e.g. SGX-Step) or zero-step instruction processing in the TEE (e.g. Microscope replay attack, PLATYPUS power side-channel attack). During single-stepping or zero-stepping, a malicious hypervisor or operating system (OS) may be able to increase the granularity of side channel information which can be collected during the TEE processing. Analyzing side channel information is a method that can be used to infer information, such as instruction flows and data, about the TEE. Thus, there is value in techniques that can mitigate these attack techniques, specifically single-stepping and zero-stepping of TEEs.

side-channel attack) and then resumes execution of the code from the enclave according to embodiments of the disclosure.

[0011] FIG. **8** illustrates a method of handling an asynchronous exit of the execution of code from an enclave that utilizes an enclave enter instruction, an enclave exit instruction, and an enclave resume instruction that invokes a handler to handle an operating system signal caused by the asynchronous exit and then resumes execution of the code from the enclave according to embodiments of the disclosure.

[0012] FIG. **9** illustrates a method of handling an exception with an enclave that comprises a field to indicate a set of one or more exceptions to suppress, and when execution of the code in the enclave encounters the exception, a handler is invoked without delivering the exception to an operating system according to embodiments of the disclosure.

[0013] FIG. **10** illustrates a hardware processor coupled to storage that includes one or more enclave instructions (e.g., an enclave resume (ERESUME) instruction) according to embodiments of the disclosure.

[0014] FIG. **11** is a flow diagram illustrating operations of a method for processing an "ERESUME" instruction according to embodiments of the disclosure.

[0015] FIG. **12** is a flow diagram illustrating operations of another method for processing an "ERESUME" instruction according to embodiments of the disclosure.

[0016] FIG. **13**A is a block diagram illustrating a generic

Configuring the Timer Interrupt

🔔 **SGX-Step goal:** Executing enclaves one instruction at a time

Challenge: we need a very precise timer interrupt:

☹ x86 hardware *debug features* disabled in enclave mode

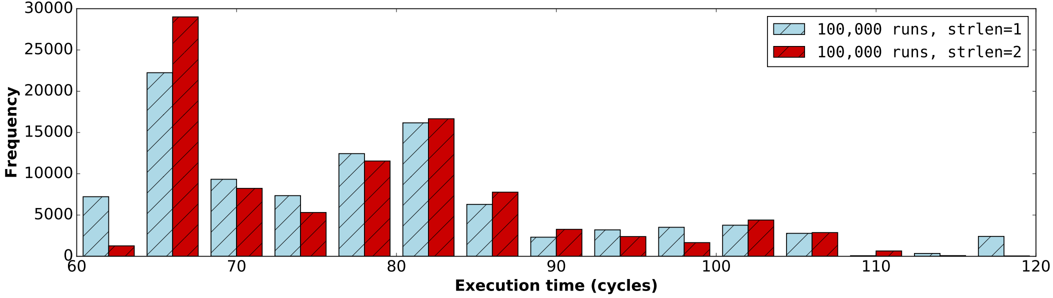
☺ . . . but we have *root access!*

⇒ Setup user-space virtual **memory mappings** for x86 APIC (+ PTEs)

```
jo@sgx-laptop:~$ cat /proc/iomem | grep "Local APIC"
fee00000-fee00fff : Local APIC
jo@sgx-laptop:~$ sudo devmem2 0xFEE00030 h
/dev/mem opened.
Memory mapped at address 0x7f37dc187000.
Value at address 0xFEE00030 (0x7f37dc187030): 0x15
jo@sgx-laptop:~$
```

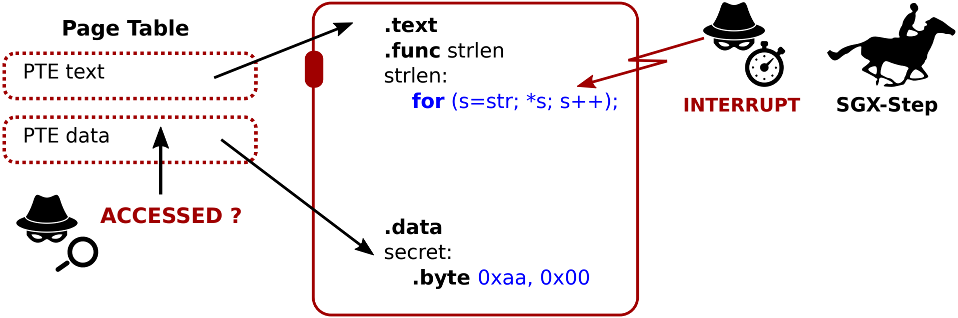
Building the `strlen()` side-channel oracle with execution timing?

Too noisy: modern x86 processors are lightning fast. . .



Counting `strlen()` loop iterations with SGX-Step

Page table accessed bit set? → **strlen++** → resume



Demo: Breaking AES-NI with the `strlen()` null byte oracle

```
Useless leakage 48 for 484
Useless leakage 48 for 485
Useless leakage 48 for 486
Useless leakage 48 for 487
Useless leakage 48 for 488
Useless leakage 48 for 489
Useless leakage 48 for 490
Useless leakage 48 for 491
Useless leakage 48 for 492
Useless leakage 48 for 493
Useless leakage 48 for 494
Useless leakage 48 for 495
Useless leakage 18 for 496
Useless leakage 48 for 497
Useless leakage 48 for 498
Useless leakage 48 for 499
Useless leakage 48 for 500
Useless leakage 48 for 501
Useless leakage 48 for 502
Useless leakage 48 for 503
Useless leakage 48 for 504
Useless leakage 48 for 505
Useless leakage 48 for 506
Useless leakage 48 for 507
Useless leakage 48 for 508
Useless leakage 48 for 509
Useless leakage 48 for 510
Useless leakage 48 for 511
Useless leakage 48 for 512
Useless leakage 48 for 513
Useless leakage 48 for 514
Useless leakage 20 for 515
Useless leakage 48 for 516
Useless leakage 48 for 517
Useless leakage 48 for 518
Useless leakage 48 for 519
Useful leak at 520 for key byte 15 = c5-> already known
Current rk10 = 13 11 1d 7f e3 94 00 17 f3 07 a7 8b 4d 2b 30 c5
Useful leak at 521 for key byte 6 = 4a-> NEW!
All round key bytes found after 522 plaintexts
Current rk10 = 13 11 1d 7f e3 94 4a 17 f3 07 a7 8b 4d 2b 30 c5
   sgx-dsn:~/0xbadc0de-poc/intel-sgx-sdk-strlen-ssa$
```

00:06 00:06

References i

A. C. Aldaya and B. B. Brumley.
When one vulnerable primitive turns viral: Novel single-trace attacks on ECDSA and RSA.
IACR Transactions on Cryptographic Hardware and Embedded Systems, pp. 196–221, 2020.

A. C. Aldaya and B. B. Brumley.
Online template attacks: Revisited.
CHES, pp. 28–59, 2021.

A. C. Aldaya, C. P. García, and B. B. Brumley.
From A to Z: Projective coordinates leakage in the wild.
IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020.

P. Borrello, A. Kogler, M. Schwarzl, M. Lipp, D. Gruss, and M. Schwarz.
ÆPIC Leak: Architecturally leaking uninitialized data from the microarchitecture.
In *USENIX Security*, 2022.

M. Busi, J. Noorman, J. Van Bulck, L. Galletta, P. Degano, J. T. Mühlberg, and F. Piessens.
Provably secure isolation for interruptible enclaved execution on small microprocessors.
In *33rd IEEE Computer Security Foundations Symposium (CSF)*, pp. 262–276, June 2020.

References ii

M. Bognar, H. Winderix, J. Van Bulck, and F. Piessens.
Microprofiler: Principled side-channel mitigation through microarchitectural profiling.
In *8th IEEE European Symposium on Security and Privacy (EuroS&P)*, July 2023.

G. Chen, S. Chen, Y. Xiao, Y. Zhang, Z. Lin, and T. H. Lai.
SgxPectre attacks: Stealing Intel secrets from SGX enclaves via speculative execution.
In *4th IEEE European Symposium on Security and Privacy (Euro S&P)*, 2019.

C. Canella, D. Genkin, L. Giner, D. Gruss, M. Lipp, M. Minkin, D. Moghimi, F. Piessens, M. Schwarz, B. Sunar, J. Van Bulck, and Y. Yarom.
Fallout: Leaking data on Meltdown-resistant CPUs.
In *26th ACM Conference on Computer and Communications Security (CCS)*, pp. 769–784, November 2019.

C. Chuengsatiansup, D. Genkin, Y. Yarom, and Z. Zhang.
Side-channeling the kalyna key expansion.
In *CT-RSA*, 2022.

S. Cuyt.
A security analysis of interrupts in embedded enclaved execution.
Master's thesis, KU Leuven, 2019.

S. Constable, J. Van Bulck, X. Cheng, Y. Xiao, C. Xing, I. Alexandrovich, T. Kim, F. Piessens, M. Vij, and M. Silberstein.
Aex-notify: Thwarting precise single-stepping attacks through interrupt awareness for intel sgx enclaves.
In *32nd USENIX Security Symposium*, pp. 4051–4068, August 2023.

J. Cui, J. Z. Yu, S. Shinde, P. Saxena, and Z. Cai.
Smashex: Smashing SGX enclaves using exceptions.
In *CCS*, 2021.

F. Dall, G. De Micheli, T. Eisenbarth, D. Genkin, N. Heninger, A. Moghimi, and Y. Yarom.
CacheQuote: efficiently recovering long-term secrets of SGX EPID via cache attacks.
IACR Transactions on Cryptographic Hardware and Embedded Systems, (2):171–191, 2018.

C. Easdon, M. Schwarz, M. Schwarzl, and D. Gruss.
Rapid prototyping for microarchitectural attacks.
In *31st USENIX Security Symposium (USENIX Security 22)*, pp. 3861–3877, 2022.

J. Gyselinck, J. Van Bulck, F. Piessens, and R. Strackx.
Off-limits: Abusing legacy x86 memory segmentation to spy on enclaved execution.
In *International Symposium on Engineering Secure Software and Systems (ESSoS)*, pp. 44–60, June 2018.

M. Hähnel, W. Cui, and M. Peinado.
High-resolution side channels for untrusted operating systems.
In *USENIX Annual Technical Conference (ATC)*, 2017.

S. Hosseinzadeh, H. Liljestrand, V. Leppänen, and A. Paverd.
Mitigating branch-shadowing attacks on intel sgx using control flow randomization.
In *3rd Workshop on System Software for Trusted Execution (SysTEX)*, pp. 42–47, 2018.

T. Huo, X. Meng, W. Wang, C. Hao, P. Zhao, J. Zhai, and M. Li.
Bluethunder: A 2-level directional predictor based side-channel attack against SGX.
IACR Transactions on Cryptographic Hardware and Embedded Systems, pp. 321–347, 2020.

References v

S. Huang, R. Q. Sim, C. Chuengsatiansup, Q. Guo, and T. Johansson.
Cache-timing attack against HQC.
IACR ePrint Archive, 2023.

W. Huang, S. Xu, Y. Cheng, and D. Lie.
Aion attacks: Manipulating software timers in trusted execution environment.
In *DIMVA*, 2021.

W. He, W. Zhang, S. Das, and Y. Liu.
Sgxlinger: A new side-channel attack vector based on interrupt latency against enclave execution.
In *36th IEEE International Conference on Computer Design (ICCD)*, pp. 108–114, 2018.

L. Hu, F. Zhang, Z. Liang, R. Ding, X. Cai, Z. Wang, and W. Jin.
Faultmorse: An automated controlled-channel attack via longest recurring sequence.
Computers & Security, 124:103003, 2023.

S. Islam, A. Moghimi, I. Bruhns, M. Krebbel, B. Gulmezoglu, T. Eisenbarth, and B. Sunar.
SPOILER: Speculative load hazards boost rowhammer and cache attacks.
In *28th USENIX Security Symposium*, 2019.

References vi

Intel.
Deep Dive: Load Value Injection, March 2020.

Intel.
Intel trust domain extension research and assurance, April 2023.

Z. Kou, W. He, S. Sinha, and W. Zhang.
Load-step: A precise trustzone execution control framework for exploring new side-channel attacks like flush+ evict.
In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 979–984, 2021.

D. Lantz, F. Boeira, and M. Asplund.
Towards self-monitoring enclaves: Side-channel detection using performance counters.
In *Nordic Conference on Secure IT Systems*, pp. 120–138. Springer, 2022.

J. Lee, J. Jang, Y. Jang, N. Kwak, Y. Choi, C. Choi, T. Kim, M. Peinado, and B. B. Kang.
Hacking in Darkness: Return-Oriented Programming Against Secure Enclaves.
In *26th USENIX Security Symposium*, pp. 523–539, 2017.

M. Lipp, A. Kogler, D. Oswald, M. Schwarz, C. Easdon, C. Canella, and D. Gruss.
Platypus: Software-based power side-channel attacks on x86.
In *S&P*, pp. 355–371, 2021.

M. Luo and G. E. Suh.
Wip: Interrupt attack on tee-protected robotic vehicles.
In *Workshop on Automotive and Autonomous Vehicle Security (AutoSec)*, April 2022.

S. Lee, M.-W. Shih, P. Gera, T. Kim, H. Kim, and M. Peinado.
Inferring fine-grained control flow inside SGX enclaves with branch shadowing.
In *26th USENIX Security Symposium*, pp. 557–574, 2017.

F. Lang, W. Wang, L. Meng, J. Lin, Q. Wang, and L. Lu.
Mole: Mitigation of side-channel attacks against sgx via dynamic data location escape.
In *Proceedings of the 38th Annual Computer Security Applications Conference*, pp. 978–988, 2022.

A. Moghimi, G. Irazoqui, and T. Eisenbarth.
Cachezoom: How SGX amplifies the power of cache attacks.
In *19th International Conference on Cryptographic Hardware and Embedded Systems (CHES)*, 2017.

References viii

K. Murdock, D. Oswald, F. D. Garcia, J. Van Bulck, D. Gruss, and F. Piessens.
Plundervolt: Software-based fault injection attacks against Intel SGX.
In *41st IEEE Symposium on Security and Privacy (S&P)*, pp. 1466–1482, May 2020.

D. Moghimi.
Downfall: Exploiting speculative data gathering.
In *32nd USENIX Security Symposium (USENIX Security 23)*, 2023.

D. Moghimi, J. Van Bulck, N. Heninger, F. Piessens, and B. Sunar.
CopyCat: Controlled instruction-level attacks on enclaves.
In *29th USENIX Security Symposium*, pp. 469–486, August 2020.

I. Puddu, M. Schneider, M. Haller, and S. Capkun.
Frontal attack: Leaking control-flow in SGX via the CPU frontend.
In *USENIX Security*, pp. 663–680, 2021.

I. Puddu, M. Schneider, D. Lain, S. Boschetto, and S. Čapkun.
On (the lack of) code confidentiality in trusted execution environments.
arXiv, 2022.

H. Ragab, A. Milburn, K. Razavi, H. Bos, and C. Giuffrida.
CrossTalk: Speculative data leaks across cores are real.
In *42nd IEEE Symposium on Security and Privacy (S&P)*, May 2021.

K. Ryan.
Hardware-backed heist: Extracting ecdsa keys from qualcomm's trustzone.
In *26th ACM Conference on Computer and Communications Security (CCS)*, pp. 181–194, 2019.

F. Sieck, S. Berndt, J. Wichelmann, and T. Eisenbarth.
Util::lookup: Exploiting key decoding in cryptographic libraries.
In *CCS*, p. 2456–2473, 2021.

M. Schwarz, M. Lipp, D. Moghimi, J. Van Bulck, J. Stecklina, T. Prescher, and D. Gruss.
ZombieLoad: Cross-privilege-boundary data sampling.
In *26th ACM Conference on Computer and Communications Security (CCS)*, pp. 753–768, November 2019.

D. Skarlatos, M. Yan, B. Gopireddy, R. Sprabery, J. Torrellas, and C. W. Fletcher.
Microscope: Enabling microarchitectural replay attacks.
In *46th International Symposium on Computer Architecture (ISCA)*, pp. 318–331, 2019.

References x

- S. ul Hassan, I. Gridin, I. M. Delgado-Lozano, C. P. García, J.-J. Chi-Domínguez, A. C. Aldaya, and B. B. Brumley.
  Déjà vu: Side-channel analysis of mozilla's nss.
  arXiv preprint arXiv:2008.06004, 2020.
- J. Van Bulck, D. Moghimi, M. Schwarz, M. Lipp, M. Minkin, D. Genkin, Y. Yuval, B. Sunar, D. Gruss, and F. Piessens.
  LVI: Hijacking transient execution through microarchitectural load value injection.
  In *41st IEEE Symposium on Security and Privacy (S&P)*, pp. 54–72, May 2020.
- J. Van Bulck, M. Minkin, O. Weisse, D. Genkin, B. Kasikci, F. Piessens, M. Silberstein, T. F. Wenisch, Y. Yarom, and R. Strackx.
  Foreshadow: Extracting the keys to the Intel SGX kingdom with transient out-of-order execution.
  In *27th USENIX Security Symposium*, pp. 991–1008, August 2018.
- J. Van Bulck, D. Oswald, E. Marin, A. Aldoseri, F. D. Garcia, and F. Piessens.
  A tale of two worlds: Assessing the vulnerability of enclave shielding runtimes.
  In *26th ACM Conference on Computer and Communications Security (CCS)*, pp. 1741–1758, November 2019.

J. Van Bulck, F. Piessens, and R. Strackx.
SGX-Step: A practical attack framework for precise enclave execution control.
In *2nd Workshop on System Software for Trusted Execution (SysTEX)*, pp. 4:1–4:6. ACM, October 2017.

J. Van Bulck, F. Piessens, and R. Strackx.
Nemesis: Studying microarchitectural timing leaks in rudimentary CPU interrupt logic.
In *25th ACM Conference on Computer and Communications Security (CCS)*, pp. 178–195, October 2018.

J. Van Bulck, N. Weichbrodt, R. Kapitza, F. Piessens, and R. Strackx.
Telling your secrets without page faults: Stealthy page table-based attacks on enclaved execution.
In *26th USENIX Security Symposium*, pp. 1041–1056, August 2017.

N. Weichbrodt, A. Kurmus, P. Pietzuch, and R. Kapitza.
Asyncshock: Exploiting synchronisation bugs in Intel SGX enclaves.
In *European Symposium on Research in Computer Security (ESORICS)*, 2016.

S. Weiser, R. Spreitzer, and L. Bodner.
Single trace attack against RSA key generation in Intel SGX SSL.
In *13th ACM Asia Conference on Computer and Communications Security (AsiaCCS)*, pp. 575–586, 2018.

S. Weiser, D. Schrammel, L. Bodner, and R. Spreitzer.
Big numbers–big troubles: Systematically analyzing nonce leakage in (ec) dsa implementations.
In *29th USENIX Security Symposium*, 2019.

L. Wilke, J. Wichelmann, A. Rabich, and T. Eisenbarth.
Sev-step: A single-stepping framework for amd-sev.
arXiv preprint arXiv:2307.14757, 2023.

Y. Xu, W. Cui, and M. Peinado.
Controlled-channel attacks: Deterministic side channels for untrusted operating systems.
In *36th IEEE Symposium on Security and Privacy (S&P)*, pp. 640–656, 2015.

Y. Xiao, Y. Zhang, and R. Teodorescu.
Speechminer: A framework for investigating and measuring speculative execution vulnerabilities.
In *Network and Distributed System Security Symposium (NDSS)*, 2020.

T. Yavuz, F. Fowze, G. Hernandez, K. Y. Bai, K. R. Butler, and D. J. Tian.
Encider: detecting timing and cache side channels in sgx enclaves and cryptographic apis.
IEEE Transactions on Dependable and Secure Computing, 20(2):1577–1595, 2022.

J. Yu, T. Jaeger, and C. W. Fletcher.
All your pc are belong to us: Exploiting non-control-transfer instruction btb updates for dynamic pc extraction.
In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, pp. 1–14, 2023.

Z. Zhang, Y. Cheng, D. Liu, S. Nepal, Z. Wang, and Y. Yarom.
Pthammer: Cross-user-kernel-boundary rowhammer through implicit accesses.
arXiv preprint arXiv:2007.08707, 2020.

Z. Zhang, M. Tao, S. O'Connell, C. Chuengsatiansup, D. Genkin, and Y. Yarom.
BunnyHop: Exploiting the instruction prefetcher.
In *32nd USENIX Security Symposium (USENIX Security 23)*, pp. 7321–7337, 2023.

J. Zhou, Y. Xiao, R. Teodorescu, and Y. Zhang.
Enclyzer: Automated analysis of transient data leaks on intel sgx.
In *2022 IEEE International Symposium on Secure and Private Execution Environment Design (SEED)*, pp. 145–156. IEEE, 2022.