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

Jo Van Bulck
imec-DistriNet, KU Leuven, Belgium
jo.vanbulck@cs.kuleuven.be

Frank Piessens
imec-DistriNet, KU Leuven, Belgium
frank.piessens@cs.kuleuven.be

ABSTRACT

This short paper presents insights from our experience in maintaining SGX-Step, an open-source attack framework designed to facilitate vulnerability research on Intel Software Guard Extensions (SGX) enclave technology. At a high level, SGX-Step enables precise "single-stepping" of hardware-isolated victim enclaves without native debugging capabilities. By strategically triggering page faults and timer interrupts, SGX-Step can seamlessly interleave arbitrary attacker code at a *maximal* temporal resolution, i.e., after every individual victim instruction. Since its original release in 2017, SGX-Step has been widely utilized in over 38 academic papers, driving an ongoing line of high-resolution, interrupt-driven enclave side-channel attacks. Its impact also extends beyond Intel SGX, directly inspiring similar tools for alternative hardware enclave technologies and even finding applications in non-enclave settings. From a defensive perspective, SGX-Step has permanently refined the trusted-execution threat model and guided recent defensive works that now properly account for single-stepping adversaries. Highlighting its continued relevance, SGX-Step has prompted recent architectural extensions to Intel SGX's hardware specification, introducing interrupt-awareness for enclaves to enable the development of principled mitigations against interrupt-driven attacks in software. A first such mitigation that thwarts deterministic single-stepping has already been included in the Intel SGX-SDK.

1 INTRODUCTION

Recent efforts from industry and academia have developed hardware-based trusted execution environments (TEEs) to safeguard critical application code and data in the presence of a possibly hostile operating system. The introduction of the popular Software Guard Extensions (SGX) [17, 35], included in selected Intel processors from 2015 onwards, has first made support for such "enclaved execution" widely available on mainstream computing platforms. SGX has since enabled a wide variety of enclave applications and is presently even used as a critical building block for Intel's upcoming Trust Domain Extensions (TDX) [36] server technology. SGX thus holds the promise of securely outsourcing private computations to untrusted remote cloud platforms.

However, the popularity of SGX and its strengthened adversary model has also enabled novel offensive research. Specifically, researchers have shown that privileged SGX adversaries can exercise their increased control over the untrusted operating system to mount an innovative series of highly capable and low-noise side-channel attacks, also referred to as controlled-channel attacks [84]. One commonality underlying many of these attacks is that they benefit from frequently preempting the victim enclave through privileged page faults or timer interrupts to repeatedly gather side-channel samples. Thus, the temporal resolution of these attacks is dictated by the accuracy with which an adversary can interrupt a victim enclave. This early insight, resulting from our own pioneering interrupt-driven attack efforts [75, 76], is what first motivated us to release the open-source SGX-Step [74] framework for fine-grained enclave execution control. Looking back, SGX-Step brought two main novelties to the early SGX attack scene.

First, SGX-Step contributed an innovative manipulation technique of the processor's integrated advanced programmable interrupt controller (APIC) timer device that for the first time allowed to reliably interrupt enclaves at a *maximal* temporal resolution, i.e., exactly one instruction at a time. This was in notable contrast to state-of-the-art attacks [26, 45, 50] at the time, which only succeeded in preempting enclaves rather unreliably at best several instructions at once. SGX-Step has since been leveraged in a long line of high-resolution, interrupt-driven side-channel attacks [2, 3, 8, 14, 18, 25, 30, 31, 31, 32, 47, 52, 56–58, 63, 65, 71–73, 75], some of which critically rely on the ability to deterministically count the number of interrupted enclave instructions. Furthermore, from a defensive perspective, SGX-Step's precise single-stepping capability fundamentally defeated early mitigations [22, 33, 45] that relied on partial atomic behavior of the enclave instruction stream. SGX-Step subsequently informed more recent defensive works [5, 9, 15, 28, 34], which now properly take single-stepping adversary capabilities into account. Most notably, SGX-Step's versatility and precision in single-stepping production enclaves was considered so impactful that it explicitly prompted Intel to develop extensions to the SGX instruction set architecture (ISA). Specifically, we collaborated with Intel on AEX-Notify [15], a minimal hardware-software co-design, which is now integrated into recent Intel processors and SDKs, to facilitate the development of principled mitigations against interrupt-driven attacks in software.

As a second novelty, SGX-Step was the first framework of its kind to offer open-source reusable building blocks for rapid enclave attack prototyping. That is, state-of-the-art SGX side-channel attacks at the time required developing a custom and fragile kernel driver [26, 50, 76, 84] or even patching and recompiling the entire operating system kernel [45]. In contrast, SGX-Step was purposefully engineered for compatibility with unaltered stock Linux kernels, requiring only a minimal driver to effectively delegate privileged adversary capabilities to user space. After loading the SGX-Step driver, all required attack primitives, including manipulating x86 APIC timer device registers or page-table entries, can be directly implemented with the help of a convenient attack library within the untrusted, user-space enclave host process. SGX-Step thus considerably lowered the bar for enclave attack development, as evidenced from the 38 academic research papers, many

Cybersecurity Artifacts Competition, ACSAC, 2023

of which were published at top-tier security venues, that were directly built on its reusable code base to date. Notably, the impact of our framework also extends beyond only Intel SGX. On the one hand, SGX-Step's convenient user-space page-table remapping and manipulation interfaces have found adoption for generic x86 attack prototyping [10, 21, 83, 87] and even inspired dedicated tools such as PTEditor [21, 62]. On the other hand, in the context of alternative TEEs, the widespread adoption and visibility of SGX-Step has directly sparked the development of analogous single-stepping frameworks with similar names, such as Sancus-Step [19], Load-Step [41], SEV-Step [81], and TDX-Step [37].

Open-Source Artifact. Over the past six years, we have actively maintained SGX-Step as an open-source project available at https://github.com/jovanbulck/sgx-step, continuously enhancing the framework with new features and ensuring compatibility with the latest Linux kernel and SGX-SDK versions. Beyond its academic impact, which is evident from the 196 citations for the original paper [74] and the 38 subsequent publications directly based on our code, SGX-Step has also garnered considerable recognition in the open-source community, marked by 393 stars, 81 forks, and 27 watchers. While SGX-Step has matured substantially since its inception as a research prototype aimed at addressing an intricate technical challenge, it remains a living project and there are certainly opportunities for further refinement. Nonetheless, our commitment to enhancing SGX-Step is reflected in the 33 support issues and the 15 pull requests accepted from external contributors so far, all contributing to its ongoing development and strength.

2 FRAMEWORK OVERVIEW

SGX-Step primarily serves as a universal execution control framework that enables the precise interleaving of victim enclave instructions with *arbitrary* attacker code. Because native x86 hardware debug events are suppressed for SGX production enclaves [35], SGX-Step offers alternative methods to interrupt enclaves, utilizing either timer interrupt requests (IRQs) or page faults. The former enables precise single-stepping of one instruction at a time, whereas the latter can be employed in coarse-grained controlled-channel attacks [84] to intercept specific code or data page accesses at a 4 KiB granularity. Practical attacks often combine both techniques by first steadily advancing the enclaved execution using a page-fault state machine, before finally enabling single-stepping mode to reach a chosen gadget of interest within a selected victim page. SGX-Step can, furthermore, be used to "zero-step" a victim enclave by repeatedly faulting without making architectural forward progress, which can serve as a capable amplification primitive to forcibly replay transient instructions or data accesses [47, 63, 66, 71].

Figure 1 overviews the main components of our framework. SGX-Step bootstraps by loading a minimal Linux kernel driver, named `/dev/sgx-step`, which effectively extends traditionally privileged operating system capabilities into user space. Our design philosophy, from the project's inception, has been to keep this loadable kernel module as minimal as possible for the sake of usability and compatibility across Linux versions. Thus, any needed configurations are maximally achieved through stock Linux kernel command line boot parameters, while avoiding custom kernel compilation altogether. The primary function of `/dev/sgx-step` is to override

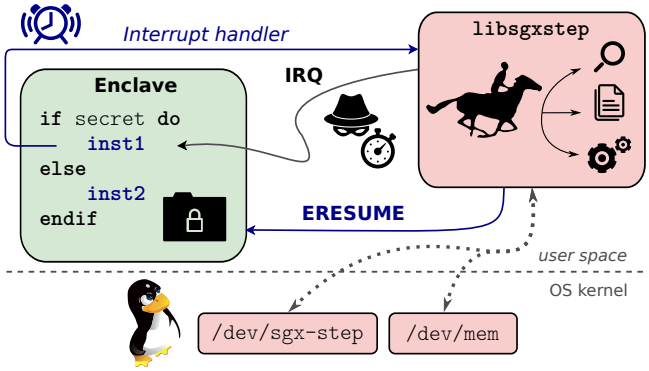

Figure 1: Overview of main SGX-Step components.

permissions on Linux's default `/dev/mem` device, granting unrestricted access to arbitrary physical memory locations. Additionally, it provides a convenient `ioctl` system-call interface for tasks such as walking page tables and establishing shadow mappings in kernel space. The core functionality of SGX-Step is thus realized in a compact `libsgxstep` library, which is linked into the untrusted enclave host application together with any specific attack logic. This library essentially acts as a mini user-space operating system, enabling direct access to privileged x86 system structures, including page-table entries and the memory-mapped I/O registers of the local APIC. Notably, some of SGX-Step's functionality has undergone substantial enhancements since its initial release. This is particularly noticeable in the interrupt handling logic, which was initially hooked statically in the kernel, but has since been relocated entirely to `libsgxstep`. This transition involved remapping the x86 global and interrupt descriptor tables into user space, allowing to install arbitrary interrupt handlers and even privileged ring-0 call gates in the untrusted enclave host application (while taking care to maintain a globally visible shadow mapping in the kernel for any interrupt handlers). Modern SGX-Step distributions, furthermore, include support for inter-processor interrupts, CPU scheduling and pinning, segmentation, cache and transient-execution attack primitives, and accessing debug enclave memory and register contents.

The main technical hurdle left for users of SGX-Step is to establish a suitable, platform-specific value for the APIC timer interrupt interval, depending on the processor's internal bus frequency and microcode version. SGX-Step includes reference timer interval values and a benchmarking tool to assist further gradual fine-tuning of this parameter using a long instruction slide in an attacker-controlled debug enclave. An overestimated timer interval will lead to "multi-step" observations, where the enclave advanced more than one instruction on the slide. Conversely, if the interval is set too low, excessive zero-step events without progress will be observed. Conveniently, the latter can be deterministically filtered out in recent SGX-Step versions by inspecting the "accessed" bit in the enclave's code page-table entry, which is only ever set when the instruction actually retired and a single-step event occurred [15, 75]. Thus, even though the onboard APIC timer is known to be inherently noisy [15], SGX-Step can achieve highly reliable, fully *deterministic* execution control even when single-stepping hundreds of thousands of instructions in practice [2, 3, 52, 56, 74].

3 IMPACT ON ATTACK RESEARCH

The ability to forcibly single-step a production enclave has proven to be be a particularly powerful and versatile attack primitive in practice. Table 1 presents an overview of published works using SGX-Step and the broader landscape of early interrupt-driven SGX attacks. The table clearly shows SGX-Step's pioneering role in introducing true single-stepping capabilities and its increasing adoption and prominence in subsequent high-resolution attack research. Additionally, even without using single-stepping, a considerable number of works [4, 10, 29, 43, 68, 79, 80, 83, 87, 89] have harnessed SGX-Step's powerful page-table manipulation interface. Lastly, the table highlights that our user-space library methodology significantly simplified the process of prototyping attacks, eliminating the need for custom kernel modules or compilation.

Novel IRQ Leakage Sources. Notably, SGX-Step has driven a distinctive category of entirely *new* attacks that would not be possible without its deterministic single-stepping capabilities.

A first influential class of attacks is based on interrupt latency [27, 56, 57, 75]. These attacks exploit that the time it takes to process an interrupt depends on the specific instruction being executed within the enclave at the moment the interrupt request arrived. SGX-Step supports transparently collecting a precise interrupt latency trace during single-stepping. This trace essentially acts as a microarchitectural "x-ray" of the enclaved execution, breaking down overall execution time into a revealing chronological sequence of *individual* instruction timings, which expose fine-grained details like instruction opcodes, operand values, and branch decisions within a cache line. Following SGX-Step, interrupt latency attacks have since also been ported to alternative TEEs, including MSP430 microcontrollers [6, 75] and AMD Secure Encrypted Virtualization (SEV) [81].

A second, particularly powerful line of attacks [2, 3, 40, 52, 73, 74] recognizes that the number of single-stepping interrupts between page accesses directly reveals the number of instructions executed in the victim enclave. Thus, interrupt counting allows for the fully deterministic detection of extremely subtle control-flow imbalances, ultimately differing only in a single instruction, which may otherwise not be exploitable in practice. Several studies [2, 3, 52] have used SGX-Step's instruction counts to extract full keys from vetted real-world cryptographic libraries, even in cases involving one-time key generation functions that require precisely probing successive branches in a single run of the victim enclave [52]. Instruction counting with SGX-Step has also been applied to build deterministic side-channel oracles in challenging scenarios involving tight loops, e.g., to exploit slightly non-constant-time `strlen()` pointer validation [73] or `memcmp()` password comparison [69] logic.

Lastly, SGX-Step timer interrupts or page faults have been used for zero-stepping. This was first applied to forcibly reload interrupted register contents in the context of transient-execution attacks [51, 63, 71] and later generalized to replay microarchitectural resource utilization [66] and power consumption [47].

High-Resolution Probing. The most general and widely used application of SGX-Step is to amplify the temporal resolution of side-channel attacks that rely on frequent probing. Such interrupt amplification with SGX-Step has been repeatedly applied to the CPU cache [14, 30, 31, 65], allowing to accurately probe remarkably nuanced secret-dependent data accesses in tight loops or bypass software prefetching mitigations. Other works have similarly used SGX-Step to probe the branch predictor [32, 86, 88] or segmentation [25] units at a precise, instruction-level granularity. Finally, SGX-Step has also been utilized to accurately progress a victim enclave's execution until it reaches a selected gadget of interest, e.g., in the context of transient execution [51, 58, 63, 71, 72] or interface [8, 18, 73] attacks.

Table 1: Overview of academic papers building on SGX-Step (highlighted rows with 🐎), alongside early interrupt-driven attacks, in terms of the temporal resolution, primary use case, and whether custom kernel compilation is avoided.

Yr	Venue	Paper	Step	Use Case	Drv
'15	S&P	Ctrl channel [84]	~ Page	*Probe* (page fault)	✓
'16	ESORICS	AsyncShock [78]	~ Page	*Exploit* (mem safety)	–
'17	CHES	CacheZoom [50]	✗ >1	*Probe* (L1 cache)	✓
'17	ATC	Hahnel et al. [26]	✗ 0 - >1	*Probe* (L1 cache)	✓
'17	USENIX	BranchShadow [45]	✗ 5 - 50	*Probe* (BPU)	✗
'17	USENIX	Stealthy PTE [76]	~ Page	*Probe* (page table)	✓
'17	USENIX	DarkROP [44]	~ Page	*Exploit* (mem safety)	✓
'17	SysTEX	SGX-Step [74]	✓ 0 - 1	*Framework*	✓ 🐎
'18	ESSoS	Off-limits [25]	✓ 0 - 1	*Probe* (segmentation)	✓ 🐎
'18	AsiaCCS	Single-trace RSA [80]	~ Page	*Probe* (page fault)	✓ 🐎
'18	USENIX	Foreshadow [71]	✓ 0 - 1	*Probe* (transient exec)	✓ 🐎
'18	EuroS&P	SgxPectre [11]	~ Page	*Exploit* (transient)	✓
'18	CHES	CacheQuote [20]	✗ >1	*Probe* (L1 cache)	✓
'18	ICCD	SGXlinger [27]	✗ >1	*Probe* (IRQ latency)	✗
'18	CCS	Nemesis [75]	✓ 1	*Probe* (IRQ latency)	✓ 🐎
'19	USENIX	Spoiler [38]	✓ 1	*Probe* (IRQ latency)	✓ 🐎
'19	CCS	ZombieLoad [63]	✓ 0 - 1	*Probe* (transient exec)	✓ 🐎
'19	CCS	Fallout [10]	–	*Probe* (transient exec)	✓ 🐎
'19	CCS	Tale of 2 worlds [73]	✓ 1	*Exploit* (mem safety)	✓ 🐎
'19	ISCA	MicroScope [66]	~ 0 - Page	*Framework*	✗
'20	CHES	Bluethunder [32]	✓ 1	*Probe* (BPU)	✓ 🐎
'20	USENIX	Big troubles [79]	~ Page	*Probe* (page fault)	✓ 🐎
'20	S&P	Plundervolt [53]	–	*Exploit* (undervolt)	✓ 🐎
'20	CHES	Viral primitive [2]	✓ 1	*Probe* (IRQ count)	✓ 🐎
'20	USENIX	CopyCat [52]	✓ 1	*Probe* (IRQ count)	✓ 🐎
'20	S&P	LVI [72]	✓ 1	*Exploit* (transient)	✓ 🐎
'20	CHES	A to Z [4]	~ Page	*Probe* (page fault)	✓ 🐎
'20	CCS	Déjà Vu NSS [68]	~ Page	*Probe* (page fault)	✓ 🐎
'20	MICRO	PTHammer [87]	–	*Probe* (page walk)	✓ 🐎
'21	USENIX	Frontal [56]	✓ 1	*Probe* (IRQ latency)	✓ 🐎
'21	S&P	CrossTalk [58]	✓ 1	*Probe* (transient exec)	✓ 🐎
'21	CHES	Online template [3]	✓ 1	*Probe* (IRQ count)	✓ 🐎
'21	NDSS	SpeechMiner [83]	–	*Framework*	✓ 🐎
'21	S&P	Platypus [47]	✓ 0 - 1	*Probe* (voltage)	✓ 🐎
'21	DIMVA	Aion [31]	✓ 1	*Probe* (cache)	✓ 🐎
'21	CCS	SmashEx [18]	✓ 1	*Exploit* (mem safety)	✓ 🐎
'21	CCS	Util::Lookup [65]	✓ 1	*Probe* (L3 cache)	✓ 🐎
'22	USENIX	Rapid prototyping [21]	✓ 1	*Framework*	✓ 🐎
'22	CT-RSA	Kalyna expansion [14]	✓ 1	*Probe* (L3 cache)	✓ 🐎
'22	SEED	Enclyzer [89]	–	*Framework*	✓ 🐎
'22	NordSec	Self-monitoring [43]	~ Page	*Defense* (detect)	✓ 🐎
'22	AutoSec	Robotic vehicles [48]	✓ 1 - >1	*Exploit* (timestamp)	✓ 🐎
'22	ACSAC	MoLE [42]	✓ 1	*Defense* (randomize)	✓ 🐎
'22	USENIX	AEPIC [8]	✓ 1	*Probe* (I/O device)	✓ 🐎
'22	arXiv	Confidential code [57]	✓ 1	*Probe* (IRQ latency)	✓ 🐎
'23	ComSec	FaultMorse [29]	~ Page	*Probe* (page fault)	✓ 🐎
'23	CHES	HQC timing [30]	✓ 1	*Probe* (L3 cache)	✓ 🐎
'23	ISCA	Belong to us [86]	✓ 1	*Probe* (BPU)	✓ 🐎
'23	USENIX	BunnyHop [88]	✓ 1	*Probe* (BPU)	✓ 🐎
'23	USENIX	DownFall [51]	✓ 0 - 1	*Probe* (transient exec)	✓ 🐎
'23	USENIX	AEX-Notify [15]	✓ 1	*Defense* (prefetch)	✓ 🐎

Alternative Use Cases and Platforms. SGX attack research has proven to be a fertile ground for pioneering novel side-channel attack methods. Interestingly, it has been observed [61, 70] that some of these innovations, initially targeted at a privileged TEE adversary model, have subsequently found applications in traditional isolation environments, such as unprivileged processes or virtual machines. This phenomenon underscores the importance of offensive enclave research and the cross-pollination of ideas and techniques between TEEs and conventional security models.

As a first tendency, SGX-Step's open-source framework, particularly its user-space page-table manipulation interface, has been employed to prototype generic x86 attacks that are unrelated to SGX [10, 21, 83, 87]. Furthermore, it has served as an inspiration for dedicated frameworks like PTEditor [62].

Second, the interrupt-driven single-stepping technique pioneered by SGX-Step has since been ported to similar frameworks for other TEEs. This includes Sancus-Step [19] targeting Sancus [54] enclaves on embedded MSP430 microcontrollers, Load-Step [41] and CacheGrab [59] for ARM TrustZone, and most recently SEV-Step [81] perfecting prior attempts [46, 77] to single-step encrypted AMD SEV virtual machines. Moreover, as part of the security assurance program of their upcoming TDX architecture, Intel [37] in close collaboration with Google [1] has demonstrated TDX-Step attacks. All of these projects are explicitly named after SGX-Step and acknowledge our framework as an inspiration, marking its lasting impact on the wider TEE attack research landscape.

4 IMPACT ON MITIGATIONS

Software Hardening. A significant outcome of confronting privileged adversary capabilities through SGX-Step is that it has conclusively guided applications towards strict compliance with constant-time coding practices. This is most evident in the long line of attacks that leverage SGX-Step [2–4, 14, 29, 30, 32, 52, 56, 65, 68, 79, 80, 88] or derived TEE single-stepping frameworks [41, 59, 81, 90] to exploit ever more nuanced side-channel leaks in vetted cryptographic libraries. Without SGX-Step, many of these vulnerabilities may not have been deemed exploitable in practice, thus effectively raising the bar for the stricter verification of cryptographic implementations and even informing improved, single-stepping-aware side-channel detection tools [85].

The enhanced precision of single-stepping adversaries has also informed specialized compiler mitigations. For instance, our single-stepping attacks [25, 74, 75] against the Zigzagger [45] branch-shadowing mitigation have directly inspired an improved compile-time hardening technique based on randomization [28]. Once more highlighting the importance of the continued attack-defense cycle, however, even this improved compiler mitigation, explicitly designed to withstand SGX-Step, was subsequently bypassed via interrupt counting [52] or architectural interfaces that approximate SGX-Step on RISC-V [23]. Likewise, Intel explicitly mentions SGX-Step in the security analysis of their software-based Load Value Injection (LVI) mitigations [34]. Finally, in the context of low-end embedded TEEs, interrupt latency attacks have sparked a line of compile-time balancing mitigations [7, 55, 60, 82].

Reactive Interrupt Detection. In the original SGX specification [49], enclaves are explicitly *interrupt-unaware*. In response to the interrupt-driven techniques supported by SGX-Step, various researchers [5, 12, 13, 24, 64, 67] have explored leveraging Intel's (deprecated) Transactional Synchronization Extensions (TSX) to make enclaves interrupt-aware. However, heuristic detection approaches are inherently fragile, suffering from both false positives and negatives [31, 39]. Moreover, TSX is scarcely available and comes with substantial performance overheads [64, 67].

Proactive Interrupt Safeguards. One of the most notable consequences of SGX-Step is the development of more principled, *architectural* approaches to address interrupt-driven attacks. For instance, the embedded Sancus TEE prototype includes a modified, hardware-level interrupt mechanism that was formally proven to be free from side-channel leakage, including interrupt latency [9].

Moreover, in explicit response to SGX-Step [15, 16], Intel recently announced ISA extensions integrated in contemporary SGX processors. Particularly, we engaged in a collaborative effort with Intel to develop a flexible hardware-software co-design known as AEX-Notify [15]. The hardware component empowers SGX enclaves with interrupt awareness by allowing them to register a trusted software handler to be executed following an interrupt or exception. Following a thorough root-cause analysis of SGX-Step's underlying behavior, we developed a robust AEX-Notify software handler to thwart deterministic single-stepping attacks. Our handler, included in the latest Intel SGX-SDK, includes a constant-time disassembler and a carefully crafted assembly stub to transparently determine and atomically prefetch the working set of the next enclave application instruction. This prefetching ensures that the next application instruction executes quickly and does not fault, thereby obviating the prerequisites for single- and zero-stepping attacks (while not precluding multi-step or fault-driven attacks). Notably, SGX-Step's open-source model has been instrumental during this entire process. Not only prompting Intel to address interrupt-awareness for SGX enclaves, a long-standing limitation, but also to inform its root-cause analysis and to stress-test rejected early alternatives [15] as well as the final mitigation, thus helping to ensure the solution's robustness and effectiveness in practice.

Likewise, Intel ensured that their upcoming TDX 1.0 software module contains a mitigation against TDX-Step attacks [1, 37]. The TDX module is described to recognize high interrupt frequencies and, upon detecting a potential single-stepping attack, randomly delays the interrupt delivery to the adversarial virtual-machine monitor. This marks tangible progress: thanks to the open model provided by SGX-Step, Intel and Google could swiftly prototype TDX-Step (even well ahead of their internal timeline [37]) and develop and test a mitigation prior to TDX's public release.

5 CONCLUSIONS AND OUTLOOK

SGX-Step constitutes a paradigm shift in TEE attack research, effectively redefining the boundaries of enclave security analysis, much like the advent of high-speed photography cameras revolutionized the exploration of fast-paced and fleeting phenomena in the physical world. We firmly believe that the continued development of SGX-Step's open-source prototype, accessible to all for inspection and enhancement, is vital to confront the ultimate consequences of a privileged TEE adversary model and, thus, set the bar for adequate enclave mitigations in the confidential computing era.

ACKNOWLEDGEMENTS

This research is partially funded by grants of the Research Foundation – Flanders (FWO), under grants number 1261222N and G081322N, and by the Flemish Research Programme Cybersecurity.

REFERENCES

[1] Erdem Aktas, Cfir Cohen, Josh Eads, James Forshaw, and Felix Wilhelm. 2023. Intel Trust Domain Extensions (TDX) Security Review. https://services.google.com/fh/files/misc/intel_tdx_-_full_report_041423.pdf

[2] Alejandro Cabrera Aldaya and Billy Bob Brumley. 2020. When One Vulnerable Primitive Turns Viral: Novel Single-Trace Attacks on ECDSA and RSA. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2020), 196–221.

[3] Alejandro Cabrera Aldaya and Billy Bob Brumley. 2021. Online template attacks: Revisited. *CHES* (2021), 28–59.

[4] Alejandro Cabrera Aldaya, Cesar Pereida García, and Billy Bob Brumley. 2020. From A to Z: Projective Coordinates Leakage in the Wild. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2020).

[5] Fritz Alder, N Asokan, Arseny Kurnikov, Andrew Paverd, and Michael Steiner. 2019. S-Faas: Trustworthy and Accountable Function-as-a-Service Using Intel SGX. In *ACM Conference on Cloud Computing Security Workshop*. 185–199.

[6] Marton Bognar, Jo Van Bulck, and Frank Piessens. 2022. Mind the Gap: Studying the Insecurity of Provably Secure Embedded Trusted Execution Architectures. In *43rd IEEE Symposium on Security and Privacy (S&P)*.

[7] Marton Bognar, Hans Winderix, Jo Van Bulck, and Frank Piessens. 2023. Micro-Profiler: Principled Side-Channel Mitigation through Microarchitectural Profiling. In *8th IEEE European Symposium on Security and Privacy (EuroS&P)*.

[8] Pietro Borrello, Andreas Kogler, Martin Schwarzl, Moritz Lipp, Daniel Gruss, and Michael Schwarz. 2022. ÆPIC Leak: Architecturally Leaking Uninitialized Data from the Microarchitecture. In *USENIX Security*.

[9] Matteo Busi, Job Noorman, Jo Van Bulck, Letterio Galletta, Pierpaolo Degano, Jan Tobias Mühlberg, and Frank Piessens. 2020. Provably Secure Isolation for Interruptible Enclaved Execution on Small Microprocessors. In *33rd IEEE Computer Security Foundations Symposium (CSF)*. 262–276.

[10] Claudio Canella, Daniel Genkin, Lukas Giner, Daniel Gruss, Moritz Lipp, Marina Minkin, Daniel Moghimi, Frank Piessens, Michael Schwarz, Berk Sunar, Jo Van Bulck, and Yuval Yarom. 2019. Fallout: Leaking Data on Meltdown-Resistant CPUs. In *26th ACM Conference on Computer and Communications Security (CCS)*. 769–784.

[11] Guoxing Chen, Sanchuan Chen, Yuan Xiao, Yinqian Zhang, Zhiqiang Lin, and Ten H Lai. 2019. SgxPectre Attacks: Stealing Intel Secrets from SGX Enclaves via Speculative Execution. In *4th IEEE European Symposium on Security and Privacy (Euro S&P)*.

[12] Guoxing Chen, Wenhao Wang, Tianyu Chen, Sanchuan Chen, Yinqian Zhang, XiaoFeng Wang, Ten-Hwang Lai, and Dongdai Lin. 2018. Racing in Hyperspace: Closing Hyper-Threading Side Channels on SGX with Contrived Data Races. In *39th IEEE Symposium on Security and Privacy (S&P)*.

[13] Sanchuan Chen, Xiaokuan Zhang, Michael K Reiter, and Yinqian Zhang. 2017. Detecting Privileged Side-Channel Attacks in Shielded Execution with Déjà Vu. In *12th ACM Asia Conference on Computer and Communications Security (AsiaCCS)*. 7–18.

[14] Chitchanok Chuengsatiansup, Daniel Genkin, Yuval Yarom, and Zhiyuan Zhang. 2022. Side-Channeling the Kalyna Key Expansion. In *CT-RSA*.

[15] Scott Constable, Jo Van Bulck, Xiang Cheng, Yuan Xiao, Cedric Xing, Ilya Alexandrovich, Taesoo Kim, Frank Piessens, Mona Vij, and Mark Silberstein. 2023. AEX-Notify: Thwarting Precise Single-Stepping Attacks through Interrupt Awareness for Intel SGX Enclaves. In *32nd USENIX Security Symposium*. 4051–4068.

[16] Scott Constable, Yuan Xiao, Bin Xing, Mona Vij, and Mark Shanahan. 2022. Techniques and technologies to address malicious single-stepping and zero-stepping of trusted execution environments. US Patent App. 17/485,077.

[17] V. Costan and S. Devadas. 2016. Intel SGX Explained. *IACR Cryptology ePrint Archive* 2016, 086 (2016), 1–118.

[18] Jinhua Cui, Jason Zhijingcheng Yu, Shweta Shinde, Prateek Saxena, and Zhiping Cai. 2021. SmashEx: Smashing SGX Enclaves Using Exceptions. In *CCS*.

[19] Sven Cuyt. 2019. *A Security Analysis of Interrupts in Embedded Enclaved Execution*. Master's thesis. KU Leuven.

[20] Fergus Dall, Gabrielle De Micheli, Thomas Eisenbarth, Daniel Genkin, Nadia Heninger, Ahmad Moghimi, and Yuval Yarom. 2018. CacheQuote: Efficiently Recovering Long-Term Secrets of SGX EPID via Cache Attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2 (2018), 171–191.

[21] Catherine Easdon, Michael Schwarz, Martin Schwarzl, and Daniel Gruss. 2022. Rapid prototyping for microarchitectural attacks. In *31st USENIX Security Symposium (USENIX Security 22)*. 3861–3877.

[22] Yangchun Fu, Erick Bauman, Raul Quinonez, and Zhiqiang Lin. 2017. SGX-LAPD: Thwarting Controlled Side Channel Attacks via Enclave Verifiable Page Faults. In *20th International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*. 357–380.

[23] Lukas Gerlach, Daniel Weber, Ruiyi Zhang, and Michael Schwarz. 2023. A Security RISC: Microarchitectural Attacks on Hardware RISC-V CPUs. In *44th IEEE Symposium on Security and Privacy*.

[24] Daniel Gruss, Julian Lettner, Felix Schuster, Olya Ohrimenko, Istvan Haller, and Manuel Costa. 2017. Strong and Efficient Cache Side-Channel Protection Using Hardware Transactional Memory. In *26th USENIX Security Symposium*.

[25] Jago Gyselinck, Jo Van Bulck, Frank Piessens, and Raoul Strackx. 2018. Off-limits: Abusing Legacy x86 Memory Segmentation to Spy on Enclaved Execution. In *International Symposium on Engineering Secure Software and Systems (ESSoS)*. 44–60.

[26] Marcus Hähnel, Weidong Cui, and Marcus Peinado. 2017. High-Resolution Side Channels for Untrusted Operating Systems. In *USENIX Annual Technical Conference (ATC)*.

[27] Wenjian He, Wei Zhang, Sanjeev Das, and Yang Liu. 2018. Sgxlinger: A New Side-Channel Attack Vector Based on Interrupt Latency Against Enclave Execution. In *36th IEEE International Conference on Computer Design (ICCD)*. 108–114.

[28] Shohreh Hosseinzadeh, Hans Liljestrand, Ville Leppänen, and Andrew Paverd. 2018. Mitigating Branch-Shadowing Attacks on Intel SGX Using Control Flow Randomization. In *3rd Workshop on System Software for Trusted Execution (SysTEX)*. 42–47.

[29] Lifeng Hu, Fan Zhang, Ziyuan Liang, Ruyi Ding, Xingyu Cai, Zonghui Wang, and Wenguang Jin. 2023. FaultMorse: An automated controlled-channel attack via longest recurring sequence. *Computers & Security* 124 (2023), 103003.

[30] Senyang Huang, Rui Qi Sim, Chitchanok Chuengsatiansup, Qian Guo, and Thomas Johansson. 2023. Cache-timing attack against HQC. *IACR ePrint Archive* (2023).

[31] Wei Huang, Shengjie Xu, Yueqiang Cheng, and David Lie. 2021. Aion attacks: Manipulating software timers in trusted execution environment. In *DIMVA*.

[32] Tianlin Huo, Xiaoni Meng, Wenhao Wang, Chunliang Hao, Pei Zhao, Jian Zhai, and Mingshu Li. 2020. Bluethunder: A 2-level Directional Predictor Based Side-Channel Attack Against SGX. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2020), 321–347.

[33] Intel. 2017. *Intel Software Guard Extensions Developer Guide: Protection from Side-Channel Attacks*.

[34] Intel. 2020. Deep Dive: Load Value Injection. https://intel.ly/3gnAYGj

[35] Intel. 2020. *Intel 64 and IA-32 Architectures Software Developer's Manual – Combined volumes*. Reference no. 325462-062US.

[36] Intel. 2022. Intel Trust Domain Extensions. https://cdrdv2.intel.com/v1/dl/getContent/690419

[37] Intel. 2023. Intel Trust Domain Extension Research and Assurance. https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/technical-documentation/tdx-security-research-and-assurance.html

[38] Saad Islam, Ahmad Moghimi, Ida Bruhns, Moritz Krebbel, Berk Gulmezoglu, Thomas Eisenbarth, and Berk Sunar. 2019. SPOILER: Speculative Load Hazards Boost Rowhammer and Cache Attacks. In *28th USENIX Security Symposium*.

[39] Jianyu Jiang, Claudio Soriente, and Ghassan Karame. 2022. On the Challenges of Detecting Side-Channel Attacks in SGX. In *RAID*.

[40] Deokjin Kim, Daehee Jang, Minjoon Park, Yunjong Jeong, Jonghwan Kim, Seokjin Choi, and Brent Byunghoon Kang. 2019. SGX-LEGO: Fine-grained SGX Controlled-Channel Attack and its Countermeasure. *Computers & Security* 82 (2019), 118–139.

[41] Zili Kou, Wenjian He, Sharad Sinha, and Wei Zhang. 2021. 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)*. 979–984.

[42] Fan Lang, Wei Wang, Lingjia Meng, Jingqiang Lin, Qiongxiao Wang, and Linli Lu. 2022. MoLE: Mitigation of Side-channel Attacks against SGX via Dynamic Data Location Escape. In *Proceedings of the 38th Annual Computer Security Applications Conference*. 978–988.

[43] David Lantz, Felipe Boeira, and Mikael Asplund. 2022. Towards Self-monitoring Enclaves: Side-Channel Detection Using Performance Counters. In *Nordic Conference on Secure IT Systems*. Springer, 120–138.

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

[45] Sangho Lee, Ming-Wei Shih, Prasun Gera, Taesoo Kim, Hyesoon Kim, and Marcus Peinado. 2017. Inferring Fine-Grained Control Flow Inside SGX Enclaves with Branch Shadowing. In *26th USENIX Security Symposium*. 557–574.

[46] Mengyuan Li, Yinqian Zhang, Huibo Wang, Kang Li, and Yueqiang Cheng. 2021. CIPHERLEAKS: Breaking Constant-time Cryptography on AMDSEV via the Ciphertext Side Channel. In *30th USENIX Security Symposium (USENIX Security 21)*. 717–732.

[47] Moritz Lipp, Andreas Kogler, David Oswald, Michael Schwarz, Catherine Easdon, Claudio Canella, and Daniel Gruss. 2021. PLATYPUS: Software-based Power Side-Channel Attacks on x86. In *S&P*. 355–371.

[48] Mulong Luo and G Edward Suh. 2022. WIP: Interrupt Attack on TEE-Protected Robotic Vehicles. In *Workshop on Automotive and Autonomous Vehicle Security*

(AutoSec).

[49] Frank McKeen, Ilya Alexandrovich, Alex Berenzon, Carlos V Rozas, Hisham Shafi, Vedvyas Shanbhogue, and Uday R Savagaonkar. 2013. Innovative Instructions and Software Model for Isolated Execution. In *2nd ACM International Workshop on Hardware and Architectural Support for Security and Privacy (HASP)*.

[50] Ahmad Moghimi, Gorka Irazoqui, and Thomas Eisenbarth. 2017. CacheZoom: How SGX Amplifies the Power of Cache Attacks. In *19th International Conference on Cryptographic Hardware and Embedded Systems (CHES)*.

[51] Daniel Moghimi. 2023. Downfall: Exploiting Speculative Data Gathering. In *32nd USENIX Security Symposium (USENIX Security 23)*.

[52] Daniel Moghimi, Jo Van Bulck, Nadia Heninger, Frank Piessens, and Berk Sunar. 2020. CopyCat: Controlled Instruction-Level Attacks on Enclaves. In *29th USENIX Security Symposium*. 469–486.

[53] Kit Murdock, David Oswald, Flavio D. Garcia, Jo Van Bulck, Daniel Gruss, and Frank Piessens. 2020. Plundervolt: Software-Based Fault Injection Attacks Against Intel SGX. In *41st IEEE Symposium on Security and Privacy (S&P)*. 1466–1482.

[54] J. Noorman, J. Van Bulck, J. Tobias Mühlberg, F. Piessens, P. Maene, B. Preneel, I. Verbauwhede, J. Götzfried, T. Müller, and F. Freiling. 2017. Sancus 2.0: A Low-Cost Security Architecture for IoT Devices. *ACM Transactions on Privacy and Security* 20, 3 (2017), 1–33.

[55] Sepideh Pouyanrad, Jan Tobias Mühlberg, and Wouter Joosen. 2020. SCF$^{\text{MSP}}$: static detection of side channels in MSP430 programs. In *15th International Conference on Availability, Reliability and Security (ARES)*. ACM, 21:1–21:10.

[56] Ivan Puddu, Moritz Schneider, Miro Haller, and Srdjan Capkun. 2021. Frontal Attack: Leaking Control-Flow in SGX via the CPU Frontend. In *USENIX Security*. 663–680.

[57] Ivan Puddu, Moritz Schneider, Daniele Lain, Stefano Boschetto, and Srdjan Čapkun. 2022. On (the Lack of) Code Confidentiality in Trusted Execution Environments. *arXiv* (2022).

[58] Hany Ragab, Alyssa Milburn, Kaveh Razavi, Herbert Bos, and Cristiano Giuffrida. 2021. CrossTalk: Speculative Data Leaks Across Cores Are Real. In *42nd IEEE Symposium on Security and Privacy (S&P)*.

[59] Keegan Ryan. 2019. Hardware-Backed Heist: Extracting ECDSA Keys from Qualcomm's TrustZone. In *26th ACM Conference on Computer and Communications Security (CCS)*. 181–194.

[60] Majid Salehi, Gilles De Borger, Danny Hughes, and Bruno Crispo. 2022. NemesisGuard: Mitigating interrupt latency side channel attacks with static binary rewriting. *Computer Networks* 205 (2022), 108744.

[61] Michael Schwarz and Daniel Gruss. 2020. How Trusted Execution Environments Fuel Research on Microarchitectural Attacks. *IEEE Security & Privacy Magazine Special Issue on Hardware-Assisted Security* (2020).

[62] Michael Schwarz, Moritz Lipp, and Claudio Canella. 2018. misc0110/PTEditor: A small library to modify all page-table levels of all processes from user space for x86_64 and ARMv8.

[63] Michael Schwarz, Moritz Lipp, Daniel Moghimi, Jo Van Bulck, Julian Stecklina, Thomas Prescher, and Daniel Gruss. 2019. ZombieLoad: Cross-Privilege-Boundary Data Sampling. In *26th ACM Conference on Computer and Communications Security (CCS)*. 753–768.

[64] Ming-Wei Shih, Sangho Lee, Taesoo Kim, and Marcus Peinado. 2017. T-SGX: Eradicating Controlled-Channel Attacks Against Enclave Programs. In *24th Annual Network and Distributed System Security Symposium (NDSS)*.

[65] Florian Sieck, Sebastian Berndt, Jan Wichelmann, and Thomas Eisenbarth. 2021. Util::Lookup: Exploiting Key Decoding in Cryptographic Libraries. In *CCS*. 2456–2473.

[66] Dimitrios Skarlatos, Mengjia Yan, Bhargava Gopireddy, Read Sprabery, Josep Torrellas, and Christopher W Fletcher. 2019. MicroScope: Enabling Microarchitectural Replay Attacks. In *46th International Symposium on Computer Architecture (ISCA)*. 318–331.

[67] Raoul Strackx and Frank Piessens. 2017. The Heisenberg Defense: Proactively Defending SGX Enclaves Against Page-Table-Based Side-Channel Attacks. *arXiv preprint arXiv:1712.08519* (Dec. 2017).

[68] Sohaib ul Hassan, Iaroslav Gridin, Ignacio M. Delgado-Lozano, Cesar Pereida García, Jesús-Javier Chi-Domínguez, Alejandro Cabrera Aldaya, and Billy Bob Brumley. 2020. Déjà Vu: Side-Channel Analysis of Mozilla's NSS. *arXiv preprint arXiv:2008.06004* (2020).

[69] Jo Van Bulck. 2020. app/memcmp: IRQ Counting Timing Attack Example. https://github.com/jovanbulck/sgx-step/tree/master/app/memcmp

[70] Jo Van Bulck. 2020. *Microarchitectural Side-Channel Attacks for Privileged Software Adversaries*. Ph. D. Dissertation. KU Leuven.

[71] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F. Wenisch, Yuval Yarom, and Raoul Strackx. 2018. Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution. In *27th USENIX Security Symposium*. 991–1008.

[72] Jo Van Bulck, Daniel Moghimi, Michael Schwarz, Moritz Lipp, Marina Minkin, Daniel Genkin, Yarom Yuval, Berk Sunar, Daniel Gruss, and Frank Piessens. 2020. LVI: Hijacking Transient Execution through Microarchitectural Load Value Injection. In *41st IEEE Symposium on Security and Privacy (S&P)*. 54–72.

[73] Jo Van Bulck, David Oswald, Eduard Marin, Abdulla Aldoseri, Flavio D. Garcia, and Frank Piessens. 2019. A Tale of Two Worlds: Assessing the Vulnerability of Enclave Shielding Runtimes. In *26th ACM Conference on Computer and Communications Security (CCS)*. 1741–1758.

[74] Jo Van Bulck, Frank Piessens, and Raoul Strackx. 2017. SGX-Step: A Practical Attack Framework for Precise Enclave Execution Control. In *2nd Workshop on System Software for Trusted Execution (SysTEX)*. ACM, 4:1–4:6.

[75] Jo Van Bulck, Frank Piessens, and Raoul Strackx. 2018. Nemesis: Studying Microarchitectural Timing Leaks in Rudimentary CPU Interrupt Logic. In *25th ACM Conference on Computer and Communications Security (CCS)*. 178–195.

[76] Jo Van Bulck, Nico Weichbrodt, Rüdiger Kapitza, Frank Piessens, and Raoul Strackx. 2017. Telling your Secrets Without Page Faults: Stealthy Page Table-Based Attacks on Enclaved Execution. In *26th USENIX Security Symposium*. 1041–1056.

[77] Wubing Wang, Mengyuan Li, Yinqian Zhang, and Zhiqiang Lin. 2023. PwrLeak: Exploiting Power Reporting Interface for Side-Channel Attacks on AMD SEV. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 46–66.

[78] Nico Weichbrodt, Anil Kurmus, Peter Pietzuch, and Rüdiger Kapitza. 2016. AsyncShock: Exploiting Synchronisation Bugs in Intel SGX Enclaves. In *European Symposium on Research in Computer Security (ESORICS)*.

[79] Samuel Weiser, David Schrammel, Lukas Bodner, and Raphael Spreitzer. 2019. Big Numbers–Big Troubles: Systematically Analyzing Nonce Leakage in (EC) DSA Implementations. In *29th USENIX Security Symposium*.

[80] Samuel Weiser, Raphael Spreitzer, and Lukas Bodner. 2018. Single Trace Attack Against RSA Key Generation in Intel SGX SSL. In *13th ACM Asia Conference on Computer and Communications Security (AsiaCCS)*. 575–586.

[81] Luca Wilke, Jan Wichelmann, Anja Rabich, and Thomas Eisenbarth. 2023. SEV-Step: A Single-Stepping Framework for AMD-SEV. *arXiv preprint arXiv:2307.14757* (2023).

[82] Hans Winderix, Jan Tobias Mühlberg, and Frank Piessens. 2021. Compiler-Assisted Hardening of Embedded Software Against Interrupt Latency Side-Channel Attacks. In *IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 667–682.

[83] Yuan Xiao, Yinqian Zhang, and Radu Teodorescu. 2020. SPEECHMINER: A framework for investigating and measuring speculative execution vulnerabilities. In *Network and Distributed System Security Symposium (NDSS)*.

[84] Yuanzhong Xu, Weidong Cui, and Marcus Peinado. 2015. Controlled-Channel Attacks: Deterministic Side Channels for Untrusted Operating Systems. In *36th IEEE Symposium on Security and Privacy (S&P)*. 640–656.

[85] Tuba Yavuz, Farhaan Fowze, Grant Hernandez, Ken Yihang Bai, Kevin RB Butler, and Dave Jing Tian. 2022. ENCIDER: detecting timing and cache side channels in SGX enclaves and cryptographic APIs. *IEEE Transactions on Dependable and Secure Computing* 20, 2 (2022), 1577–1595.

[86] Jiyong Yu, Trent Jaeger, and Christopher Wardlaw Fletcher. 2023. 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*. 1–14.

[87] Zhi Zhang, Yueqiang Cheng, Dongxi Liu, Surya Nepal, Zhi Wang, and Yuval Yarom. 2020. PThammer: Cross-User-Kernel-Boundary Rowhammer through Implicit Accesses. *arXiv preprint arXiv:2007.08707* (2020).

[88] Zhiyuan Zhang, Mingtian Tao, Sioli O'Connell, Chitchanok Chuengsatiansup, Daniel Genkin, and Yuval Yarom. 2023. BunnyHop: Exploiting the Instruction Prefetcher. In *32nd USENIX Security Symposium (USENIX Security 23)*. 7321–7337.

[89] Jiuqin Zhou, Yuan Xiao, Radu Teodorescu, and Yinqian Zhang. 2022. ENCLYZER: Automated Analysis of Transient Data Leaks on Intel SGX. In *2022 IEEE International Symposium on Secure and Private Execution Environment Design (SEED)*. IEEE, 145–156.

[90] KOU Zili, Sharad Sinha, HE Wenjian, and Wei ZHANG. 2023. Cache Side-channel Attacks and Defenses of the Sliding Window Algorithm in TEEs. In *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 1–6.