ReminISCence: Trusted Monitoring Against Privileged Preemption Side-Channel Attacks

Weijie Chen[1], Yu Zhao[1], Yinqian Zhang[4], Weizhong Qiang[1,3](✉), Deqing Zou[1,3], and Hai Jin[2]

[1] National Engineering Research Center for Big Data Technology and System, Hubei Key Laboratory of Distributed System Security, Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China
{weijie_chen,z_y,deqingzou}@hust.edu.cn, wzqiang@hust.edu.cn

[2] National Engineering Research Center for Big Data Technology and System, Services Computing Technology and System Lab, Cluster and Grid Computing Lab School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China
hjin@hust.edu.cn

[3] Jinyinhu Laboratory, Wuhan 430040, China

[4] Department of Computer Science and Engineering, Research Institute of Trustworthy Autonomous Systems, Southern University of Science and Technology, Shenzhen 518055, China
yinqianz@acm.org

Abstract. Trusted Execution Environments (TEEs) have long served as a prominent security measure for ensuring isolation and data privacy in cloud environments. However, their security foundations face challenges from numerous side-channel threats, particularly those involving privileged capabilities that enable potent preemption attacks. Various solutions exist to mitigate these attacks, including monitoring-based ones featured with higher efficiency. Unfortunately, existing monitoring-based solutions do not consider privileged preemption attacks and, therefore, are not qualified for trusted monitoring within TEE enclaves. In this paper, we propose ReminISCence, a novel trusted monitoring framework designed to mitigate privileged preemption side-channel attacks on TEE architectures. We present a trusted scheduling design that enforces control over the timer interrupts, which ensures the monitoring relies on untampered trusted time slices with valid lengths and cannot be bypassed via arbitrary timer preemption. Consequently, the privileged adversary is constrained to performing preemption attacks within trusted time slices solely via non-timer interrupts, which are confidentially monitored with full coverage by ReminISCence. We implement the ReminISCence prototype on off-the-shelf RISC-V hardware by extending the OpenSBI and leveraging the RISC-V HPM facility. Our evaluations demonstrate the prototype's effectiveness and resilience to evasion in monitoring and analyzing preemption attacks of various RISC-V microarchitectural

https://doi.org/10.1007/978-3-031-70903-6_2

side-channels while maintaining high temporal resolution with negligible performance overhead (approximately 1% overhead at a resolution of 125 *us*).

Keywords: Side-channel · Preemption attacks · Hardware performance monitor · Trusted execution environment · RISC-V

1 Introduction

With the prevalence of cloud computing comes the demand for data privacy and trustworthiness, across architectures and hardware. In response, a plethora of TEEs for various architectures have been extensively explored in both academia and industry, including the most far-reaching ones, such as Intel SGX [23], ARM TrustZone [3], and RISC-V Keystone [31], upon which in-depth research and ecosystem are established.

Unfortunately, the resource-sharing nature of multi-tenant clouds breeds numerous microarchitectural side-channel attacks, including cache-based [36,50, 52], TLB-based [20], and transient attacks [27,34], etc. Initially, side-channel adversaries are considered unprivileged with quite limited capabilities, and most TEE designs explicitly exclude side-channel issues from their threat models. Nevertheless, these attacks have also been proven feasible on modern TEEs, effectively leaking enclave information and challenging their security foundations [8,9,22,46]. Even more concerning is that within the TEE context, where the attacker could compromise the OS, the original unprivileged side-channel attacks can evolve into more potent forms. Specifically, the untrusted OS is still responsible for resource allocation and scheduling. Thus, a privileged attacker can control the interrupt handling and frequency of the enclave, enabling enclave program preemption at arbitrary intervals and much finer-grained side-channel observations (e.g., instruction-by-instruction) [11,12,28,48,49].

In this regard, numerous protection techniques have been presented to mitigate side-channel threats. Application-based methods aim to eliminate side-channel vulnerabilities at the source code or binary level, such as constant-time programming [6,7] and obfuscated execution [35,42]. However, their security is no longer guaranteed under privileged attacks, which can bypass these approaches with much higher attack resolution and distinguish target information from obfuscation. Moreover, hardware-based solutions also exist, such as partitioning [15,43] and usage randomization [45,47], but according to [19], these hardware modifications do not appear to be favored by up-to-date CPUs.

Apart from these, several runtime monitoring solutions have also been proposed, which can act as an oracle for runtime mitigations featured with higher efficiency and flexibility. Nevertheless, in the context of privileged side-channel attacks within TEEs, these solutions fail to exhibit the necessary trustworthiness and security. On the one hand, unprivileged monitoring techniques [30,40,51], which constitute the majority of them, cannot guarantee the trustworthiness of their monitoring process under the TEE threat model. As their code and data are not protected from external tampering, and they rely on untrusted data sources (e.g., OS-provided interfaces such as perf and PAPI) for HPM sampling, the privileged adversary can easily manipulate and defeat them. On the other hand, as HPM hardware resources are usually core-specific, privileged monitoring techniques [29,44] can be bypassed by the privileged adversary using the preemption capability to schedule the enclave to other cores at will. Furthermore, existing approaches overlook the hazards of exposing the HPM service to lower privileges, potentially opening new side-channel surfaces while dealing with existing ones. Notably, even if the untrusted OS cannot directly access and tamper with the HPM, a malicious enclave instance can still exploit the monitoring mechanism to obtain side-channel information of other enclaves sharing the same hardware.

To address these challenges, we propose ReminISCence, a novel trusted monitoring framework against privileged preemption side-channel attacks. Specifically, ReminISCence contains three key security designs: confidential sampling, interface abstraction, and trusted scheduling, which cover the security implications when considering privileged adversaries within TEE environments. Confidential sampling lays the security foundation that ReminISCence's components and data sources cannot be learned or tampered with externally. Interface abstraction prevents malicious enclaves from abusing ReminISCence to obtain side-channel knowledge from others. Trusted scheduling is the key design that ensures valid sampling based on trusted time slices and achieves full-coverage monitoring combined with enclave-core awareness. We have implemented ReminISCence on an off-the-shelf RISC-V platform without hardware modification by moderately extending the OpenSBI firmware. Our implementation demonstrates effective monitoring with high temporal resolution and negligible performance overhead.

In summary, we make the following contributions in this paper:

– We propose ReminISCence, a novel trusted enclave monitoring framework designed to counter privileged preemption side-channel attacks in typical TEE architectures. The framework consists of three security designs: confidential sampling, interface abstraction, and trusted scheduling, which ensures the monitoring is tamper-proof against the privileged adversary and based on trusted time slices that cannot be bypassed via preemption.
– We implement the ReminISCence prototype on off-the-shelf RISC-V hardware, which enforces trusted monitoring by extending the OpenSBI and utilizing the RISC-V HPM facility.
– Using the prototype, we monitor preemption attacks across a wide range of RISC-V microarchitectural side-channels and demonstrate the effectiveness and resilience to evasion. We demonstrate that the prototype can achieve very high temporal resolution with negligible performance overhead (approximately 1% overhead at a resolution of 125 us).

For the remainder of this paper, Sect. 2 introduces necessary background knowledge. Section 3 explains the threat model and system overview of ReminISCence. Section 4 presents the detailed implementation of the ReminISCence prototype on RISC-V. Section 5 evaluates ReminISCence and analyzes the experimental findings. Section 6 discusses related works on HPM-based monitoring. Section 7 draws the conclusion of this paper.

2 Background

2.1 Privileged Side-Channel Attacks

Previous studies have shown that unprivileged side-channel attackers can interrupt the victim's execution for more precise observations by tricking the underlying OS or hypervisor schedulers [52]. While under the TEE context, privileged side-channel attackers, namely a compromised OS responsible for enclave scheduling and trap handling, can achieve this goal more easily. In consequence, side-channel attacks evolve to more direct and potent forms known as preemption attacks, which can be generally categorized into exception-based and interrupt-based.

Exception-Based attacks. Also known as the controlled-channel attack [49], the untrusted OS manipulates an Intel SGX enclave's page table to leak its page-level information without noise, by triggering page faults and observing the trace of the faulting addresses. However, recent TEE designs, such as RISC-V ones [5,15,31], have drawn lessons from Intel SGX and have straightforwardly addressed the exception-based attacks by depriving page tables and virtual-to-physical mapping management from the untrusted privileged software.

Interrupt-Based Attacks. A malicious OS can abuse an off-chip interrupt controller to trigger high-frequency interrupts towards the enclave. Existing works on various TEEs [11,28,48] have demonstrated the hazard posed by such preemption attacks that greatly amplify the observation resolution based on various side-channel primitives. In particular, SGX-step [11] has achieved the highest instruction-by-instruction resolution through single-stepping the SGX enclaves and has bred many other astounding attacks over the years [9,10,12].

2.2 Hardware Performance Monitor

HPM is a platform-specific, hardware-driven facility initially designed to capture and analyze detailed execution performance metrics. Generally, hardware performance counters (HPCs), a group of dedicated registers that record runtime hardware events, are the key hardware component of HPM. Major ISAs like x86, ARM, and RISC-V have all introduced their specific HPM standards and monitoring solutions [2,26,32]. Besides conventional usage, studies have also shown HPM's potential for security problems, such as malware and side-channel attack detection [39,51]. Currently, the latest RISC-V privileged specification covers a maximum of 32 HPCs, divided into two parts: 1) three mandatory fixed-function counters (CTI counters), namely `cycle`, `time`, and `instret`; 2) at most 29 event-programmable HPM counters [2,17], which can be configured with platform-specific events.

2.3 RISC-V Infrastructures

RISC-V is an ISA with a fully open standard based on the reduced instruction set computer design principles. It aims to be extensible and customizable for various scales of computing systems. In RISC-V systems, only the highest privilege level M-mode (Machine) is mandatory. As the complexity increases, `U` extension for U-mode (User) and `S` extension for S-mode (Supervisor) can be added for conventional application and operating system usage, respectively.

RISC-V Traps. In RISC-V terminology, traps denote abnormal control flows from a lower to a higher privilege level. Traps consist of asynchronous interrupts, which are independent of execution of the current hart (hardware thread, a RISC-V term for logical core), and synchronous exceptions directly attributed to the execution. Specifically, there are three types of interrupts in RISC-V: timer, software, and external interrupts, where timer interrupts are responsible for scheduling, and software interrupts are used to generate known interprocessor interrupts (IPIs).

RISC-V SBI. The RISC-V supervisor binary interface (SBI) is a standard that defines the calling convention and execution environment under a Unix-like OS, aiming to provide the S-mode software with an interface to M-mode-only hardware resources as well as portability across different platforms. Currently, there are several well-established and widely-used SBI implementations such as OpenSBI [4] and RustSBI [37].

3 System Design

3.1 Threat Model

We align our threat model and basic design principles with TEEs, assuming that the OS might be compromised or malicious and that all untrusted components in the system cannot extract or tamper with the enclave's private memory. This paper focuses on privileged microarchitectural side-channel attacks, where the adversary can conduct preemption attacks with various primitives. As is explained in Sect. 2.1, we do not cover exception-based attacks in this paper, as they are eradicated in recent TEE designs. When considering evasion techniques, we align with the assumptions proposed by [25], pointing out that side-channel attacks are time-sensitive and the attacker cannot advance the attack arbitrarily slowly. This closely aligns with the goal of preemption attacks to obtain very fine-grained and accurate observations.

Given our focus on microarchitectural side-channels, we exclude consideration of untrusted entities exploiting general software vulnerabilities in the enclave program, such as logical and memory corruption bugs. Also, we do not consider transient execution attacks and cross-core attacks as their mitigations can be extended orthogonally to our approach, as well as SMT-driven attacks that can be mitigated by reasonably disabling SMT according to security requirements.

3.2 ReminISCence Overview

ReminISCence is a trusted enclave monitoring framework designed to mitigate privileged side-channel threats on typical TEE systems. Figure 1 illustrates the overview of ReminISCence. The usage of ReminISCence can be integrated as a general, default enforcement to all enclaves or as an optional feature based on the enclave application's security requirements. The running enclaves with ReminISCence enabled are referred to as monitored enclaves, whose execution should be entirely under surveillance. When considering privileged adversaries within TEE environments, the methodologies and security implications must be reconsidered as they differ significantly from those proposed in existing monitoring solutions. Therefore, we explain the three security designs of ReminISCence that are essential to monitoring against the privileged preemption adversary model.

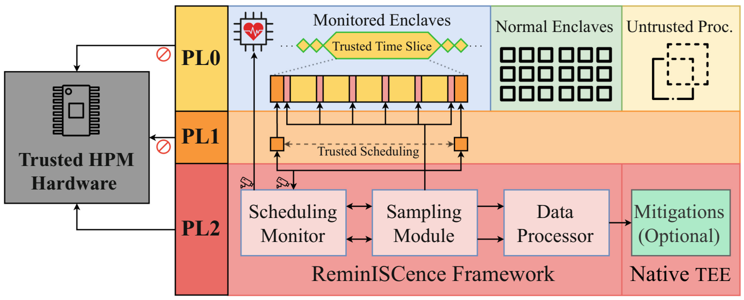


Fig. 1. Overview of ReminISCence (PL2, HPM, and enclaves are trusted)

Confidential Sampling. Confidential sampling is the fundamental foothold of ReminISCence's monitoring trustworthiness. Specifically, all the framework's internal code and data, as well as its sampling and communication processes with the target enclave, cannot be intercepted or tampered with by unauthorized entities, including the untrusted OS and potentially malicious enclave instances. Therefore, instead of relying on an external source (i.e., OS-provided facilities), the framework must sample from a trusted data source, typically through direct access to the trusted HPM hardware. In practice, the framework should be implemented as a TEE extension in the monitor privileged level (PL2), typically the highest privilege level for conventional TEEs and the hypervisor privilege level for VM-based TEEs (containing the secure VM manager, not shown in the figure). On the one hand, being in PL2 makes the framework inherently confidential against the lower privileged untrusted OS (PL1) and enables it to acquire runtime information about the enclaves from the native TEE's security monitor, including their identifier and running core. On the other hand, as HPM facilities usually incorporate a hierarchical accessibility mechanism for security concerns, the highest privilege mode has the ultimate control over the HPM, which can restrict HPM access from lower privilege levels to ensure data source trustworthiness.

Interface Abstraction. While confidential sampling is primarily against privilege capability issues of the OS (PL1), interface abstraction focuses on preventing malicious enclaves (PL0) from abusing ReminISCence. Specifically, the framework should only provide necessary interfaces to the monitored enclave. For instance, HPM resources should never be exposed to or configurable in user space through ReminISCence functionalities. Also, the internal states managed by ReminISCence should be agnostic to the target enclave. Regarding monitored results, if any, the framework can only return the monitored enclave with highly abstract results that reveal no low-level execution information. The abstraction aspect is often overlooked, resulting in the unintended creation of new side channels while attempting to address the existing ones. In our implementation, we deploy a machine-learning-based data processor for interface abstraction.

Trusted Scheduling. Many HPM resources are typically core-specific, making them sensitive to core scheduling. Consequently, privileged preemption capabilities can be exploited to disrupt and bypass the monitoring by arbitrarily scheduling the enclave to another core on the sneak. To address this, we propose trusted scheduling as a pivotal component of ReminISCence, tackling privileged preemption attacks and ensuring trusted monitoring. First, the framework should capture and have ultimate control over all interrupts during the monitored enclave's execution. Here, we categorize interrupts into timers and non-timers, with timers responsible for periodically interrupting enclave processes and performing scheduling decisions. The privileged adversary chooses either timers or non-timers to achieve high-frequency preemption. For **timers**, ReminISCence restricts their duration with a lower bound, ensuring that monitoring is built on trusted time slices with valid lengths. Otherwise, the timer-based preemption attacks can still be performed with high frequency, as well as severely compromising monitoring trustworthiness. For instance, scheduling the enclave to another core before a sampling interval finishes creates unmonitored windows. In the worst case, when the scheduling is always triggered faster than the first sampling of the time slice, the monitoring would be fully bypassed as no valid sampling can be made. In other words, trusted scheduling not only assures monitoring trustworthiness but also mitigates timer-based preemption attacks as the frequency of timers is censored and controlled. For **non-timers**, altering their arrival time or restricting their frequency may impact the usability and functionality of the enclave program. However, with timers controlled, non-timer-based preemptions targeting the trusted time slices are fully monitored by ReminISCence. Additionally, the framework is aware of the core on which the monitored enclave is executing by tracking this information through the native TEE. This ensures that ReminISCence monitors the entire execution of the enclave and the monitoring is performed on the proper core's HPM, and also prevents the malicious OS from illegally scheduling the enclave to another core through non-timers.

According to Fig. 1, ReminISCence contains three main functional parts: the scheduling monitor, the sampling module, and the data processor. The sampling module directly retrieves raw data from the trusted HPM facility through interrupt-based sampling approaches that provide dynamically adjustable sampling intervals and require no modification on the enclave program. The alternative polling-based sampling approaches are unsuitable because the sampling point depends on the enclave execution and may not fit into the trusted time slice. The scheduling monitor mainly enforces our trusted scheduling design, which checks interrupts and tracks the core residence (enclave-core awareness) during the monitored enclave's execution. The data processor is responsible for transforming the raw data from the sampling module and enforcing the interface abstraction on the final results that may be delivered to the monitored enclave. The final results or internal states can be further used as an indicator for optional runtime mitigations extended in the native TEE.

As stated above, the framework typically resides in the monitor privilege level, which implies the highest privilege level containing the security monitor of conventional TEEs. Although this paper does not delve into virtualization settings as RISC-V H-mode (Hypervisor) is still on its way to hardware integration, here we briefly discuss how ReminISCence can be applied to cutting-edge VM-based TEE designs of various architectures. As for ARM CCA, the trusted monitoring mechanism can be extended to the trusted Realm Manager Monitor (RMM) that resides in the exception level 2 of the Realm world and manages all secure VMs (Realms) inside the world. The ARM Coresight architecture [32] also explicitly recommends HPM implementations to enforce observability and access control over different worlds, which complies with the confidential sampling design. However, it might be challenging to extend ReminISCence to x86 counterparts. The Intel TDX module, which runs in a special SEAM processor mode and provides interfaces for the hypervisor to schedule and manage secure VMs, is proprietary to Intel. Fortunately, Intel TDX module specification has enforced countermeasures against interrupt-based attacks that a secure VM continues executing a random amount of instructions if interrupted too soon after resuming [24]. As for AMD SEV, it features encryption-based memory isolation and does not incorporate a trusted counterpart like the RMM or the TDX module, which makes it hard to underpin trusted scheduling. Also, the x86 PMU is normally accessible from ring 0, at which the untrusted OS runs, so confidential sampling is unable to be achieved unless further PMU access controls are applied.

4 Implementation

This section outlines the ReminISCence implementation details. We implement the ReminISCence prototype on an off-the-shelf RISC-V hardware platform, StarFive VisionFive V2. The entire implementation is software-only, extending

Table 1. Platform parameters

Description	Value
Hardware name	StarFive VisionFive V2 (DT)
CPU	RISC-V U74 Quad-Core V64GC ISA SoC@1.5 GHz
Priv. Spec. Ver.	v1.11
OS	Debian GNU/Linux bookworm/sid
Kernel	Linux 5.15.0-starfive
HPM	`cycle`, `time`, `instret`, `hpmcounter3/4`

the M-mode OpenSBI from the standard v1.0 version. Detailed parameters of our experiment platform are illustrated in Table. 1.

4.1 ReminISCing over Side-Channel Vectors on RISC-V

We first look into various microarchitectural side-channel vectors on RISC-V and their implications on preemption attacks. Recently, Gerlach et al. [19] have systematically analyzed microarchitectural side-channel vectors on currently representative RISC-V CPUs. As listed in Table 2, besides the conventional ones, RISC-V platforms suffer from new attack surfaces due to their high-resolution counters, unprivileged cache maintenance instructions, and exclusive hardware optimizations, which we respectively summarize as follows.

Table 2. Primary microarchitectural side-channel attack vectors on RISC-V

Hardware	Attack Primitives	Descriptions
Counters	CycleDrift*	User space `cycle` & `instret` counters
Cache	Evict+Reload Prime+Probe	Eviction-based, targeting L1 data (L1-D) cache
	Flush(Fence)+Reload Flush+Fault/Ret*	Flush-based, flushing the L1 instruction (L1-I) cache with `fence.i`
TLB	TLB-eviction	Eviction-based
Branch Predictor	Cache+Time*	Combining branch prefetching with flushing the L1-I cache

* New side-channel primitives proposed by Gerlach et al.

1) The RISC-V specification explicitly requires all implementations to offer the CTI counters, which by default count events in all privileges and are universally accessible. These counters enable unprivileged programs to extract exe-

cution information from higher-privileged domains, creating the CycleDrift attack.

2) RISC-V CPU vendors commonly introduce custom cache maintenance instructions, with `fence.i` being one of the most prevalent. The `fence.i` instruction enables an unprivileged attacker to flush the entire L1-I cache, which is usually impossible on other ISAs. This leads to Fence+Reload, Flush+Fault/Ret, and Cache+Time attacks, as shown in Table 2. Specifically, Fence+Reload is a variant of Flush+Reload that targets the L1-I cache using `fence.i`, while Flush+Fault(Ret) is another variant that replaces the data reload step by jumping into the shared victim code following a fault or return.
3) While most off-the-shelf RISC-V CPUs are in order, they incorporate prediction-based optimizations that speculatively prefetch branch instructions instead of executing them. In a Cache+Time attack, the attacker first flushes the L1-I cache to a deterministic state, then uses a speculative gadget to prefetch one of the branch paths into the cache, and finally measures the time required for branch execution to determine whether the prefetched path is selected.

   Regarding our work, as the attack primitives serve as injected payloads and the essence of preemption attacks remains the same, cross-platform differences in attack primitives and HPM events might yield diverse monitoring performance but do not impact the basic applicability of ReminISCence framework. On RISC-V, the timer-based preemption attacks can be achieved by manipulating the `mtimecmp` register, while the non-timer-based ones can be achieved by sending IPIs to the target hart.

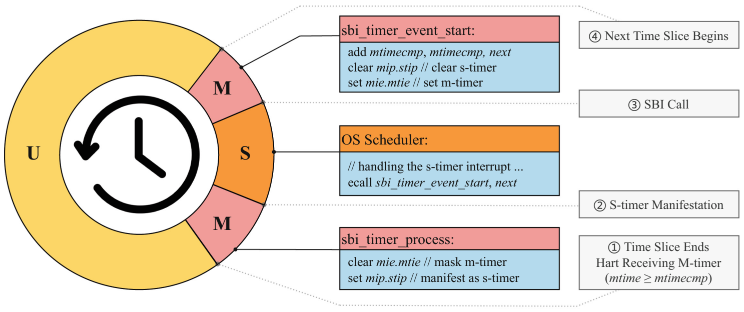
4.2 Sampling Facility

According to Sect. 3.2, the sampling method should be interrupt-based, indicating an event-based sampling method that requires performance monitoring interrupts (PMIs) and samples each time a specified number of certain events occur [16]. However, RISC-V PMI support (the Sscofpmf extension), which provides essential hardware mechanisms for generating and capturing HPC overflow interrupts, has only been ratified quite recently and yet exists in current RISC-V platforms. Therefore, to address this challenge, we adopt a m-timer-based sampling approach. The machine timer interrupts (m-timers) can be seen as a special type of PMI that necessarily exists, where a hart's `mtimecmp` register specifies the upper limit for `mtime` to trigger the m-timer. Specifically, our implementation samples trusted data directly from the HPM hardware based on the high-resolution timer (1 tick equals $\sim$250 ns), with an average sampling latency of less than 1 tick on our platform.

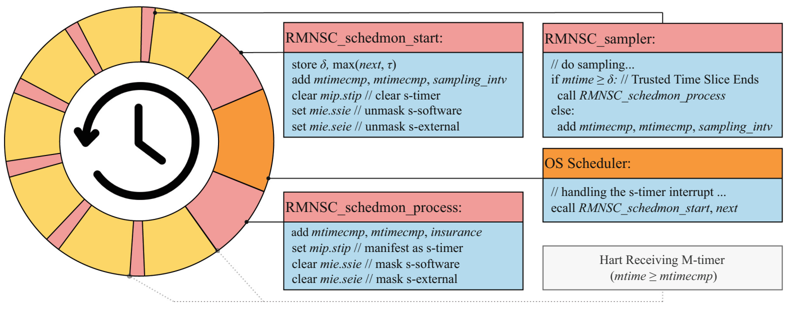
Additionally, we are aware of the non-determinism of HPCs proposed by [16]. On the one hand, we validate the determinism and high accuracy of our HPCs, aligning with the findings of [19], and our direct sampling method eliminates interference caused by the additional execution of performance tools. On the other hand, we utilize the `mcountinhibit` CSR that stops and resumes the counters' increment, to avoid data pollution due to the execution of ReminISCence. In order to straightforwardly thwart the CycleDrift attack surface, we prohibit non-M-mode accesses of `cycle` and `instret` during the monitored enclave's execution. We set the event-programmable HPM counters inaccessible to lower privileged modes during all times as exposing them creates new side-channel threats, complying with the interface abstraction design.

4.3 Trusted Scheduling

Nevertheless, if m-timers are used in event sampling, a naive solution of sampling at each m-timer's arrival and writing for the next one can corrupt the hart's regular scheduling. We delicately address this timer contention issue by integrating the m-timer-based sampling with the trusted scheduling design. Trusted scheduling ensures ReminISCence's control over the timer, enabling trusted and full-coverage monitoring under the privileged preemption threat model. To avoid interference with regular scheduling, we implement the trusted scheduling based on the normal time slice on a RISC-V hart, as illustrated in Fig. 2a. In the course of a time slice, a timer is not received until a hart's `mtimecmp` is reached by the incrementing `mtime`(①). M-timers are directly handled in M-mode, where the


U
M
S
M
sbi_timer_event_start:
add mtimecmp, mtimecmp, next
clear mip.stip // clear s-timer
set mie.mtie // set m-timer
OS Scheduler:
// handling the s-timer interrupt ...
ecall sbi_timer_event_start, next
sbi_timer_process:
clear mie.mtie // mask m-timer
set mip.stip // manifest as s-timer
④ Next Time Slice Begins
③ SBI Call
② S-timer Manifestation
① Time Slice Ends
Hart Receiving M-timer
(mtime ≥ mtimecmp)

(a) Original time slice


RMNSC_schedmon_start:
store δ, max(next, τ)
add mtimecmp, mtimecmp, sampling_intv
clear mip.stip // clear s-timer
set mie.ssie // unmask s-software
set mie.seie // unmask s-external
RMNSC_sampler:
// do sampling...
if mtime ≥ δ: // Trusted Time Slice Ends
call RMNSC_schedmon_process
else:
add mtimecmp, mtimecmp, sampling_intv
OS Scheduler:
// handling the s-timer interrupt ...
ecall RMNSC_schedmon_start, next
RMNSC_schedmon_process:
add mtimecmp, mtimecmp, insurance
set mip.stip // manifest as s-timer
clear mie.ssie // mask s-software
clear mie.seie // mask s-external
Hart Receiving M-timer
(mtime ≥ mtimecmp)

(b) Trusted time slice

Fig. 2. Detailed implementation of trusted scheduling

SBI immediately manifests it as s-timers (②). After the m-timer trap returns, the hart is promptly trapped again in the OS Scheduler that handles the s-timer and schedules the next m-timer event via the SBI call (③). Lastly, the SBI assigns the next timer event to the HPM accordingly and returns, indicating that the next time slice begins (④).

Figure 2b shows the lifecycle of a trusted time slice following the implementation of the trusted scheduling and m-timer-based sampling. At the start of each time slice, the scheduling monitor compares the *next* timer value delivered from the supervisor with a lower bound τ and keeps the greater as δ. Then, the m-timer is scheduled to the first sampling point, after *sampling_intv* ticks. For the subsequent m-timers, the sampling module compares the current `mtime` with δ and decides to either schedule the next sampling timer or call the scheduling monitor when the trusted time slice should end. Here, execution inside the trusted time slice can be viewed as an *s-timer-atomic* region because, within which, if the supervisor intends to regain control of the hart via timers, it must wait for the current time slice to end, but the length of which cannot be less than τ and is specified by the SBI rather than the supervisor.

However, it is interesting to note that, in theory, the supervisor still retains the capability to perform preemption attacks on the trusted time slice. At the OS scheduler arc of the trusted time slice, if the supervisor can hand over control to the enclave on the sneak before calling the scheduling monitor, it can still perform non-timer-based preemption attacks on the enclave's execution, the process of which would be agnostic to ReminISCence. To mend this security fissure, we enforce the OS scheduler arc to be *s-non-timer-atomic* by masking the supervisor software and external interrupts. Henceforth, the only way for the OS to preempt the enclave and regain control here is via the timer, which, falls into the trusted time slice cycle again. Additionally, to avoid losing control of the hart when the malicious OS actually performs this bypass attack without knowing the *s-non-timer-atomic* feature, the scheduling monitor sets a relatively long insurance timer to reclaim the hart in the future.

Note that we do not need to care how long the supervisor is exclusively holding the hart, as preemption attacks essentially require interleaved execution and a long piece of supervisor execution alone does not leak valid information about the enclave. However, the implementation of trusted scheduling limits the timer-based preemption interval to τ, which is orders of magnitude higher for retrieving effective side-channel results. At this point, the only alternative for the adversary to achieve a feasible resolution is non-timer-based preemption attacks targeting inside the trusted time slice. Now that the execution within the trusted time slice is fully monitored, all non-timer behavior and its microarchitectural footprint will be entirely captured and analyzed by ReminISCence. Therefore, our machine-learning-based data processor can fulfill the dual purpose of the interface abstraction design and monitor non-timer-based preemption attacks, the effectiveness of which will be demonstrated in the next section.

5 Evaluation

5.1 Monitoring Preemption Attacks

This section demonstrates the effectiveness and resilience to evasion of ReminISCence in monitoring and analyzing non-timer-based preemption attacks over side-channel attack primitives mentioned in Sect. 4.1. Specifically, we train supervised learning models using our generated datasets and classify non-timer-based attacks with different side-channel primitives and preemption intervals. The models are transformed into bare-metal native code using the emlearn library [41] to be deployed in the M-mode-only memory space and exclusively invoked by ReminISCence.

Data Collection. We create datasets comprised of HPM raw data generated from the sampling module of the prototype, with a sampling interval of 2000 ticks (∼500 us). For better representativeness, the dataset is collected from widely used benchmark suites, namely Coremark [18] and Mibench [21]. The target programs are cross-compiled for RISC-V and cover various application scenarios, including automotive, consumer, network, office, security, and telecomm (e.g., qsort, jpeg, dijkstra, ispell, sha, CRC32). We choose 16 out of 35 hardware events supported by the HPM of our platform together with the CTI counters; we organize them into 11 related features: `Instret`, `Load/Store Instret`, `Branch Instret`, `Branch Misprediction`, `D-cache Busy`, `D-cache Miss`, `D-cache Write-back`, `I-cache Busy`, `I-cache Miss`, `D-TLB Miss`, and `L2-TLB Miss`. To experiment with the system under realistic load conditions, we stress the system with memory-intensive and CPU-intensive processes by concurrently running cross-page memory copy operations and CPU-consuming programs in the `rv8` benchmark [14] like dhrystone and primes. Each program, attack primitive, and interval share equivalent data sizes to avoid bias.

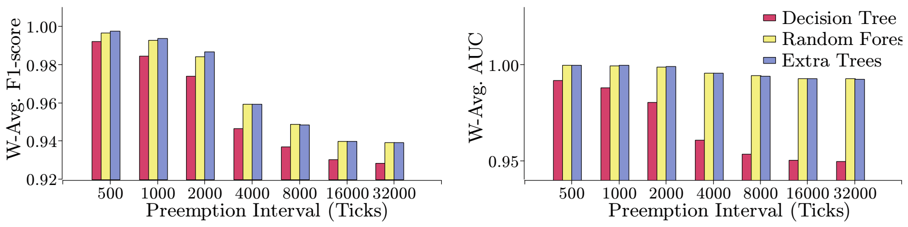
Attack Classification. While monitoring the target programs, we conduct non-timer-based attacks by sending IPI from another hart with an intuitive preemption interval of 500 ticks, to evaluate the performance of all 5 classification models supported by emlearn in distinguishing normal executions from attacked ones and identifying specific attack primitives. The classification performance, the average time per prediction and binary sizes of the models are shown in Table 3. Generally, the tree-based models, namely Decision Tree (DT), Random Forest(RF), and Extremely Randomized Trees (ERT), as well as Multilayer Perceptron (MLP) can effectively classify normal and attacked executions with F1 scores above 0.97, by contrast, Gaussian Naive Bayes (GNB) behaves poorly. Although featuring the least binary size, MLP is less accurate than the tree-based models and significantly more time-consuming, which is not advisable in frequent sampling. Hence, we hereinafter choose and discuss the tree-based models.

Despite the fact that the models cannot identify individual attack primitives with reasonable accuracy, they can effectively narrow the scope of attack

Table 3. Model evaluation results of attack primitive classfication. ⊘: Normal, CT: Cache+Time, FF: Flush+Fault, FR: Fence+Reload, ER: Evict+Reload, PP: Prime+Probe, TLB: TLB-eviction, WA: Weighted Average.

Model	Performance (F1 scores · 10^2)								Time tick	Size
	⊘	CT	FF	FR	ER	PP	TLB	WA		
DT	98.3	64.7	48.3	47.4	66.7	75.7	78.7	72.6	<1	27
	98.8	98.3			75.7		76.6	90.9		
RF	99.2	69.0	55.4	49.9	76.5	84.5	84.8	77.6	<1	243
	99.2	98.2			81.0		82.0	92.9		
ERT	99.3	68.0	52.5	49.4	75.7	83.8	84.0	76.8	<1	757
	99.1	98.3			79.7		79.7	92.2		
GNB	44.3	21.8	9.8	19.5	18.2	6.6	13.9	22.3	<1	12
	56.7	63.2			8.7		20.6	44.5		
MLP	81.4	40.0	32.4	25.3	18.9	43.1	54.0	47.6	11.6	11
	97.0	94.3			43.1		66.2	81.7		

primitives with similar microarchitectural characteristics, such as identifying the fence-based (CT, FF, FR), data-cache-based (ER, PP), and TLB-based types with weighted average F1 scores above 0.90, referring to the gray rows in the table. Due to similar extensive cross-page memory access patterns, TLB-based type introduce certain data cache anomalies as data-cache-based, resulting in their lower F1 scores comparing with normal and fence-based ones. However, if we further combine data-cache-based and TLB-based types into eviction-based type, the F1 scores of identifying which reach above 0.97 and the models' weighted averages increase to around 0.98.



(a) Weighted averages of F1 scores

(b) Weighted averages of AUC

Fig. 3. Monitoring performance under different preemption intervals

Resilience to Evasion. We are aware of evasive attacks that the adversary would leverage to avoid being detected, as is presented in RHMD [25]. Although the RHMD solution is complementary to ReminISCence, here we wish to demonstrate the formidable difficulty of non-timer-based attacks evading detection without significant impact on the side-channel bandwidth. Li and Gaudiot [33] have proven that the most effective way for microarchitectural attacks to evade ML-based detectors is bandwidth reduction via sleeping after each atomic execution of the attack primitive. First, we verify that the shortest non-timer intervals for all attack primitives are less than 1.13 ticks ($\sim$283 ns) on our platform, which indicates the highest non-timer-based preemption speed in theory. Then, we increase the preemption interval from the initial 500 ticks by powers of two and observe the models' performance on classifying normal, fence-based, and eviction-based types, as illustrated in Fig. 3. Despite the monotonic decrease in performance, the weighted averages of F1 scores and AUC remain above 92% and 95% even until the preemption interval inflates to 32000 ticks ($\sim$28319x bandwidth reduction), and their subsequent trend gradually stables. As claimed in Sect. 3.1 that side-channel attacks are time-sensitive, we argue that it is extremely difficult for the privileged adversary to evade our prototype without significant sacrifice of bandwidth and accuracy. Also, according to both Fig. 3 and Table 3, RF and ERT have similar effectiveness and overall outperform DT in the case of classification performance and resilience to evasion. Therefore, we ultimately select RF considering its slightly higher accuracy with a smaller binary size, whose false positive rate ranges within 0.17%-1.09% and false negative rate ranges within 0.35%-9.11%, corresponding with preemption intervals of 500–32000 ticks.

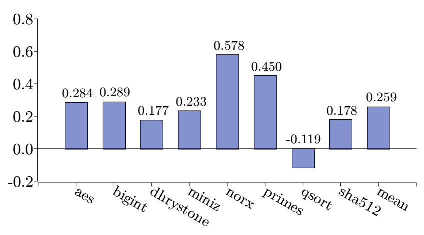
5.2 Overhead

This subsection evaluates the overhead incurred by deploying our ReminISCence prototype. According to the results in Sect. 5.1, we use the RF model with the same sampling interval trained by the same datasets ($\sim$13 MB), and the deployed model binary is about 232 KB in size.

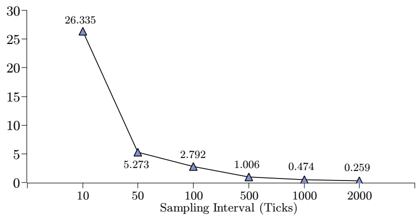
Figure 4 illustrates the runtime performance overhead of our ReminISCence implementation, according to the average of 100 different runs for each `rv8` benchmark program. Figure 4a shows that ReminISCence incurs negligible overhead below 0.6% and a mean average of 0.259% for all bench programs, under the sampling interval of 2000 ticks. One possible explanation for the negative overhead in the figure is that ReminISCence's trusted scheduling reduces the program's timer preemptions. Furthermore, we explore our performance overhead under different sampling intervals in the case of dynamically adjusted frequencies. Figure 4b indicates that ReminISCence can achieve the sampling resolution as high as 500 Ticks ($\sim$125 us) with only an average overhead of about 1%.

5.3 Security Discussion

Firstly, the privileged attacker cannot compromise the prototype as its code and data reside in the M-mode-only memory space. In addition, the privileged attacker cannot tamper with the data source as we directly sample from the



(a) Overhead under the sampling interval of 2000 ticks



(b) Average overhead under different sampling intervals

Fig. 4. ReminISCence performance overhead (in percent) on `rv8` relative to baselines

trusted HPM hardware, which is set inaccessible to lower privileges during the monitoring process. Furthermore, all HPM interfaces and internal states of the prototype are concealed from the monitored enclave, preventing malicious enclave instances from exploiting ReminISCence to gain microarchitectural information. With grounded confidentiality, we proceed to trusted monitoring against preemption attacks based on either timers or non-timers.

We implement trusted scheduling to enforce control over timers, which hinders the attacker from arbitrarily scheduling the enclave to any hart via timers. Together with the enclave-core awareness, it is ensured that the monitoring covers all time slices of the enclave using the proper hart-specific HPM. Therefore, the only remaining option for the privileged attacker by far is to preempt the trusted time slices via non-timers, which are deterministically monitored by ReminISCence. As demonstrated in Sect. 5.1, ReminISCence can effectively detect such attacks and is extremely hard to evade. As a whole, the implementation ensures trusted monitoring and mitigates preemption attacks against the privileged adversary. Last but not least, although we reasonably extend the M-mode software for around 1K LoC, this does not necessarily indicate the increase of TCB and the attack surface because the implementation never interferes with the native TEE and only provides the enclave with the interface to acquire abstract monitoring results.

6 Related Work

Several monitoring frameworks have been proposed to address performance analysis within TEEs. TEEMon [29] is a performance monitoring framework specifically designed for Intel SGX, assisting with identifying program performance bottlenecks. TS-Perf [44] is a compiler-based approach that inserts measurement code in operations like API calls and memory accesses during the compilation phase, enabling performance counting in TEEs across architectures. Microarchitectural side-channel monitoring and detection have also been extensively studied. NIGHTs-WATCH [40] utilizes machine learning models to detect Flush+Reload and Flush+Flush attacks by analyzing real-time performance counter values. Cloudradar [51] combines signature-based and anomaly-based detection methods to identify cache-based side-channel attacks in multi-tenant cloud systems. Déjà Vu [13] detects privileged side-channel attacks in Intel SGX by comparing the actual execution time with the expected one. For transient attacks, PerSpectron [1] relies on microarchitectural statistical information and a hardware-based predictor to detect Spectre and cache side-channel attacks. Le et al. [30] leveraged cache activities and neural networks to address Spectre attacks on an FPGA-based out-of-order RISC-V CPU. Speculator [38] utilizes performance counters to observe instructions during CPU speculation, providing defense against speculative execution attacks. Some studies have shown the potential evasion of monitoring methods. RHMD [25] employs random switching of models during runtime to increase the difficulty for malware adversaries in reverse-engineering the monitor. Li et al. [33] recognized that Spectre attacks can achieve evasion by reducing attack frequency but suggested that a robust attack detection rate can still be achieved through the random switching of models.

7 Conclusion

In this paper, we propose ReminISCence, a novel trusted monitoring framework specifically designed to mitigate privileged preemption side-channel attacks within TEEs. The framework incorporates three key security designs: confidential sampling, interface abstraction, and trusted scheduling, which ensures the monitoring is tamper-proof against the privileged adversary and based on trusted time slices that cannot be bypassed via preemption. We implement the ReminISCence prototype on an off-the-shelf RISC-V platform by extending OpenSBI, which demonstrates its effectiveness and resilience to evasion in monitoring high-frequency preemption attacks across various RISC-V microarchitectural side-channels. Our evaluation also showcases ReminISCence's ability to achieve high temporal resolution with negligible performance overhead.

Acknowledgements. We sincerely thank the anonymous reviewers for their constructive comments. This work was supported in part by National Key Research and Development Program of China (Grant No. 2022YFB4501500 and 2022YFB4501502) and National Natural Science Foundation of China (Grant No. 62272181).

References

1. Ajorpaz, S.M., Pokam, G., Koruyeh, E.M., Garza, E., Abu-Ghazaleh, N.B., Jiménez, D.A.: Perspectron: detecting invariant footprints of microarchitectural attacks with perceptron. In: 53rd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2020, Athens, Greece, 17–21 October 2020, pp. 1124–1137. IEEE (2020)
2. Andrew, W., Krste, A., John, H.: The risc-v instruction set manual volume ii: Privileged architecture (2021)
3. Arm Limited: Security technology: building a secure system using trustzone technology. http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC009492C_trustzone_security_whitepaper.pdf. Accessed Nov 2023

4. avpatel: Opensbi. https://github.com/riscv-software-src/opensbi. Accessed Nov 2023
5. Bahmani, R., Brasser, F., Dessouky, G., Jauernig, P., Klimmek, M., Sadeghi, A., Stapf, E.: CURE: a security architecture with customizable and resilient enclaves. In: Bailey, M.D., Greenstadt, R. (eds.) 30th USENIX Security Symposium, USENIX Security 2021, 11–13 August 2021, pp. 1073–1090. USENIX Association (2021)
6. Barthe, G., Grégoire, B., Laporte, V.: Secure compilation of side-channel countermeasures: the case of cryptographic “constant-time”. In: 31st IEEE Computer Security Foundations Symposium, CSF 2018, Oxford, United Kingdom, 9–12 July 2018, pp. 328–343. IEEE Computer Society (2018)
7. Borrello, P., D'Elia, D.C., Querzoni, L., Giuffrida, C.: Constantine: automatic side-channel resistance using efficient control and data flow linearization. In: Kim, Y., Kim, J., Vigna, G., Shi, E. (eds.) CCS 2021: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, 15–19 November 2021, pp. 715–733. ACM (2021)
8. Brasser, F., Müller, U., Dmitrienko, A., Kostiainen, K., Capkun, S., Sadeghi, A.: Software grand exposure: SGX cache attacks are practical. In: Enck, W., Mulliner, C. (eds.) 11th USENIX Workshop on Offensive Technologies, WOOT 2017, Vancouver, BC, Canada, 14–15 August 2017. USENIX Association (2017)
9. Bulck, J.V., et al.: Foreshadow: extracting the keys to the intel SGX kingdom with transient out-of-order execution. In: Enck, W., Felt, A.P. (eds.) 27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, 15–17 August 2018, pp. 991–1008. USENIX Association (2018)
10. Bulck, J.V., et al.: LVI: hijacking transient execution through microarchitectural load value injection. In: 2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, 18–21 May 2020, pp. 54–72. IEEE (2020)
11. Bulck, J.V., Piessens, F., Strackx, R.: SGX-step: a practical attack framework for precise enclave execution control. In: Proceedings of the 2nd Workshop on System Software for Trusted Execution, SysTEX@SOSP 2017, Shanghai, China, 28 October 2017. pp. 4:1–4:6. ACM (2017)
12. Bulck, J.V., Piessens, F., Strackx, R.: Nemesis: studying microarchitectural timing leaks in rudimentary CPU interrupt logic. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, 15–19 October 2018, pp. 178–195. ACM (2018)
13. Chen, S., Zhang, X., Reiter, M.K., Zhang, Y.: Detecting privileged side-channel attacks in shielded execution with déjà vu. In: Karri, R., Sinanoglu, O., Sadeghi, A., Yi, X. (eds.) Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2017, Abu Dhabi, United Arab Emirates, 2–6 April 2017, pp. 7–18. ACM (2017)
14. Clark, M.: rv8 benchmark suite. https://github.com/michaeljclark/rv8-bench. Accessed Nov 2023
15. Costan, V., Lebedev, I.A., Devadas, S.: Sanctum: minimal hardware extensions for strong software isolation. In: Holz, T., Savage, S. (eds.) 25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, 10–12 August 2016, pp. 857–874. USENIX Association (2016)
16. Das, S., Werner, J., Antonakakis, M., Polychronakis, M., Monrose, F.: Sok: the challenges, pitfalls, and perils of using hardware performance counters for security. In: 2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, 19–23 May 2019, pp. 20–38. IEEE (2019)

17. Domingos, J.M., Tomás, P., Sousa, L.: Supporting RISC-V performance counters through performance analysis tools for linux (perf). CoRR arxiv:2112.11767 (2021)
18. Gal-On, S., Levy, M.: Exploring coremark a benchmark maximizing simplicity and efficacy. The Embedded Microprocessor Benchmark Consortium (2012)
19. Gerlach, L., Weber, D., Zhang, R., Schwarz, M.: A security RISC: microarchitectural attacks on hardware RISC-V cpus. In: 44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, 21–25 May 2023, pp. 2321–2338. IEEE (2023)
20. Gras, B., Razavi, K., Bos, H., Giuffrida, C.: Translation leak-aside buffer: Defeating cache side-channel protections with {TLB} attacks. In: 27th USENIX Security Symposium (USENIX Security 2018), pp. 955–972 (2018)
21. Guthaus, M.R., Ringenberg, J.S., Ernst, D., Austin, T.M., Mudge, T., Brown, R.B.: MiBench: a free, commercially representative embedded benchmark suite. In: Proceedings of the fourth annual IEEE international workshop on workload characterization. WWC-4 (Cat. No. 01EX538), pp. 3–14. IEEE (2001)
22. Hähnel, M., Cui, W., Peinado, M.: High-resolution side channels for untrusted operating systems. In: Silva, D.D., Ford, B. (eds.) 2017 USENIX Annual Technical Conference, USENIX ATC 2017, Santa Clara, CA, USA, 12–14 July 2017, pp. 299–312. USENIX Association (2017)
23. Intel: Intel software guard extensions programming reference. https://software.intel.com/sites/default/files/managed/48/88/329298-002.pdf. Accessed Nov 2023
24. Intel: Intel trust domain extensions module base architecture specification. https://cdrdv2.intel.com/v1/dl/getContent/733575. Accessed Nov 2023
25. Khasawneh, K.N., Abu-Ghazaleh, N.B., Ponomarev, D., Yu, L.: RHMD: evasion-resilient hardware malware detectors. In: Hunter, H.C., Moreno, J., Emer, J.S., Sánchez, D. (eds.) Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2017, Cambridge, MA, USA, 14–18 October 2017, pp. 315–327. ACM (2017)
26. Kleen, A., Strong, B.: Intel processor trace on linux. Tracing Summit (2015)
27. Kocher, P., et al.: Spectre attacks: exploiting speculative execution. In: 2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, 19–23 May 2019, pp. 1–19. IEEE (2019)
28. Kou, Z., He, W., Sinha, S., Zhang, W.: Load-step: a precise trustzone execution control framework for exploring new side-channel attacks like flush+evict. In: 58th ACM/IEEE Design Automation Conference, DAC 2021, San Francisco, CA, USA, 5–9 December 2021, pp. 979–984. IEEE (2021)
29. Krahn, R., et al.: TEEMon: a continuous performance monitoring framework for tees. In: Silva, D.D., Kapitza, R. (eds.) Middleware 2020: 21st International Middleware Conference, Delft, The Netherlands, 7–11 December 2020, pp. 178–192. ACM (2020)
30. Le, A., Hoang, T., Dao, B., Tsukamoto, A., Suzaki, K., Pham, C.: A real-time cache side-channel attack detection system on RISC-V out-of-order processor. IEEE Access **9**, 164597–164612 (2021)
31. Lee, D., Kohlbrenner, D., Shinde, S., Asanovic, K., Song, D.: Keystone: an open framework for architecting trusted execution environments. In: Bilas, A., Magoutis, K., Markatos, E.P., Kostic, D., Seltzer, M.I. (eds.) EuroSys 2020: Fifteenth EuroSys Conference 2020, Heraklion, Greece, 27–30 April 2020, pp. 38:1–38:16. ACM (2020)
32. Lee, Y., Lee, J., Heo, I., Hwang, D., Paek, Y.: Using coresight PTM to integrate CRA monitoring ips in an arm-based soc. ACM Trans. Design Autom. Electr. Syst. **22**(3), 52:1–52:25 (2017)

33. Li, C., Gaudiot, J.: Detecting spectre attacks using hardware performance counters. IEEE Trans. Comput. **71**(6), 1320–1331 (2022)
34. Lipp, M., et al.: Meltdown. arXiv preprint arXiv:1801.01207 (2018)
35. Liu, C., Harris, A., Maas, M., Hicks, M.W., Tiwari, M., Shi, E.: GhostRider: a hardware-software system for memory trace oblivious computation. In: Özturk, Ö., Ebcioglu, K., Dwarkadas, S. (eds.) Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2015, Istanbul, Turkey, 14–18 March 2015, pp. 87–101. ACM (2015)
36. Liu, F., Yarom, Y., Ge, Q., Heiser, G., Lee, R.B.: Last-level cache side-channel attacks are practical. In: 2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, 17–21 May 2015, pp. 605–622. IEEE Computer Society (2015)
37. luojia65: Rustsbi. https://github.com/rustsbi/rustsbi. Accessed Nov 2023
38. Mambretti, A., Neugschwandtner, M., Sorniotti, A., Kirda, E., Robertson, W.K., Kurmus, A.: Speculator: a tool to analyze speculative execution attacks and mitigations. In: Balenson, D. (ed.) Proceedings of the 35th Annual Computer Security Applications Conference, ACSAC 2019, San Juan, PR, USA, 09–13 December 2019, pp. 747–761. ACM (2019)
39. Martin, R., Demme, J., Sethumadhavan, S.: TimeWarp: rethinking timekeeping and performance monitoring mechanisms to mitigate side-channel attacks. In: 39th International Symposium on Computer Architecture (ISCA 2012), Portland, OR, USA, 9–13 June 2012, pp. 118–129 (2012)
40. Mushtaq, M., Akram, A., Bhatti, M.K., Chaudhry, M., Lapotre, V., Gogniat, G.: NIGHTs-WATCH: a cache-based side-channel intrusion detector using hardware performance counters. In: Szefer, J., Shi, W., Lee, R.B. (eds.) Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy, HASP@ISCA 2018, Los Angeles, CA, USA, 02 June 2018, pp. 1:1–1:8. ACM (2018)
41. Nordby, J., Cooke, M., Horvath, A.: emlearn: machine learning inference engine for microcontrollers and embedded devices (2019)
42. Rane, A., Lin, C., Tiwari, M.: Raccoon: closing digital side-channels through obfuscated execution. In: Jung, J., Holz, T. (eds.) 24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA,x 12–14 June 2015, pp. 431–446. USENIX Association (2015)
43. Saileshwar, G., Qureshi, M.K.: MIRAGE: mitigating conflict-based cache attacks with a practical fully-associative design. In: Bailey, M.D., Greenstadt, R. (eds.) 30th USENIX Security Symposium, USENIX Security 2021, 11–13 August 2021, pp. 1379–1396. USENIX Association (2021)
44. Suzaki, K., Nakajima, K., Oi, T., Tsukamoto, A.: TS-Perf: general performance measurement of trusted execution environment and rich execution environment on Intel SGX, Arm TrustZone, and RISC-V Keystone. IEEE Access **9**, 133520–133530 (2021)
45. Tan, Q., Zeng, Z., Bu, K., Ren, K.: PhantomCache: obfuscating cache conflicts with localized randomization. In: 27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, 23–26 February 2020. The Internet Society (2020)
46. Wang, W., et al.: Leaky cauldron on the dark land: Understanding memory side-channel hazards in SGX. In: Thuraisingham, B., Evans, D., Malkin, T., Xu, D. (eds.) Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, 30 October–03 November 2017, pp. 2421–2434. ACM (2017)

47. Werner, M., Unterluggauer, T., Giner, L., Schwarz, M., Gruss, D., Mangard, S.: ScatterCache: thwarting cache attacks via cache set randomization. In: 28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, 14–16 August 2019. pp. 675–692 (2019)
48. Wilke, L., Wichelmann, J., Rabich, A., Eisenbarth, T.: Sev-step a single-stepping framework for AMD-SEV. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2024**(1), 180–206 (2024)
49. Xu, Y., Cui, W., Peinado, M.: Controlled-channel attacks: deterministic side channels for untrusted operating systems. In: 2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, 17–21 May 2015, pp. 640–656. IEEE Computer Society (2015)
50. Yarom, Y., Falkner, K.: FLUSH+RELOAD: a high resolution, low noise, L3 cache side-channel attack. In: Fu, K., Jung, J. (eds.) Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, 20–22 August 2014, pp. 719–732. USENIX Association (2014)
51. Zhang, T., Zhang, Y., Lee, R.B.: CloudRadar: a real-time side-channel attack detection system in clouds. In: Research in Attacks, Intrusions, and Defenses - 19th International Symposium, RAID 2016, Paris, France, 19–21 September 2016, Proceedings, pp. 118–140 (2016)
52. Zhang, Y., Juels, A., Reiter, M.K., Ristenpart, T.: Cross-vm side channels and their use to extract private keys. In: Yu, T., Danezis, G., Gligor, V.D. (eds.) the ACM Conference on Computer and Communications Security, CCS 2012, Raleigh, NC, USA, 16–18 October 2012, pp. 305–316. ACM (2012)