A Systematic Look at Quality-of-Service Feature Effects on Side Channels in AMD SEV-SNP

Sudheendra Raghav Neela[1](✉), Carina Fiedler[1], Ruiyi Zhang[2], Robin Leander Schröder[3], Michael Schwarz[2], and Daniel Gruss[1]

[1] Graz University of Technology, Graz, Austria
{sudheendra.neela, carina.fiedler, daniel.gruss}@tugraz.at
[2] CISPA Helmholtz Center for Information Security
{ruiyi.zhang, michael.schwarz}@cispa.de
[3] Fraunhofer SIT, Darmstadt, Germany & Fraunhofer Austria, Vienna
leander.schroeder@sit.fraunhofer.de

Abstract. Modern systems execute numerous workloads in parallel, with tasks contending for the same shared resources, e.g., cache and memory bandwidth. To minimize the impact on workloads by "noisy neighbors", Intel and AMD released quality-of-service (QoS) extensions to enforce limits on such resources along with monitoring metrics.
In this paper, we show that these well-intentioned quality-of-service extensions introduce new system-level cross-core side channels and amplify known attacks in confidential computing environments on modern AMD server CPUs, namely SEV-SNP. We demonstrate our attacks in the malicious or compromised hypervisor threat model, attacking an AMD SEV-SNP Confidential VM. In these settings we mount an MBM-based Bleichenbacher attack on RSA, inter-keystroke timing attacks (F_1 score up to 96 %) exploiting MBM, MBA, CAT, and CDP, a new MBA rate-limit-based side channel (up to 205 B/s), and a website fingerprinting attack (F_1 score up to 72.10 %) exploiting MBM, MBA, CAT, and CDP. We also evaluate a CAT-amplified Prime+Probe L3 covert channel with a maximum speedup of 2.27×. We conclude that these well-intention set of features require mitigation, most likely requiring hardware changes.

Keywords: Side Channels · AMD SEV-SNP · Quality of Service

1 Introduction

Quality-of-service (QoS) features of modern CPUs are the foundation to detecting and mitigating hardware-level interference in shared environments, e.g., across processes, containers, or virtual machines (VMs) [1, 54]. AMD offers QoS features as a part of their Platform Quality of Service features [3] and Intel provides these QoS features as a part of their Resource Director Technology [19, 54]. Both feature sets constitute similar features for allocation and resource monitoring. Cache Allocation Technology (CAT) partitions last-level cache (LLC) ways, Code and Data Prioritization (CDP) refines CAT with disjoint cache way masks

for code and data, and Memory Bandwidth Allocation (MBA) sets upper bounds for the traffic between the LLC and memory traffic, all per *class of service*. Cache Monitoring Technology (CMT) allows to monitor LLC occupancy and Memory Bandwidth Monitoring (MBM) reports how much data is transferred between memory and LLC, both also per class of service. With QoS features, privileged system software can ensure that tasks are allocated system resources [1], further assisting resource provisioning [36] for VM migration across systems. However, sharing not only influences quality of service but also has security implications: Over the past 30 years, researchers studied how sharing hardware resources introduces numerous side channels in modern CPUs, with cache side channels most widely studied [41, 68], both in same-core and cross-core attacks [37, 68].

Even well-intentioned features sometimes can be abused to enable or amplify attacks. For instance, CAT has been demonstrated as a useful mechanism in Rowhammer attacks [33] since the eviction of fewer (or even a single) cache ways is significantly faster. Kurth et al. [28] used CAT to mount remote cache attacks in a data center scenario using Intel's Direct Cache Access (DCA) network feature. Minkin and Kasikci [38] used CAT to restrict an SGX enclave's cache ways to reduce noise in a Prime+Probe attack from other processes, e.g., in attacks on compression algorithms. In local scenarios, CAT has also been explored as a potential cache side-channel mitigation [34, 56]. Other QoS features, namely CMT and MBA, have been similarly explored as potential defenses or detection techniques [5, 10, 67]. However, the security implications of other QoS features or combinations of QoS features have has not been studied yet.

In this paper, we present the first systematic evaluation of the security implications of quality-of-service features in modern AMD server CPUs. We show that this well-intentioned set of five QoS features introduces new system-level cross-core side channels, and in other cases amplifies known attacks in SEV-SNP confidential VMs (CVMs). We show that these features can be misused in the malicious or compromised hypervisor threat model, leaking information from a CVM across cores. The foundation of three of our four attacks on CVMs is MBM, not used in any prior attack so far. While CMT reports cache occupancy data to privileged software in multiples of 16 KiB, MBM accurately reports data transfer between cache and memory with cache-line-size granularity, *i.e.*, 64 B. We show that these techniques can be combined with MBA, which provides a configurable and collective rate-limit, inadvertently introducing a new rate-limit-based side channel. CDP enables code and data to be placed on separate L3 cache ways, making code-targeted or data-targeted cache attacks less noisy. Finally, we also confirm that restricting the cache with CAT can be used to assist microarchitectural attacks such as Prime+Probe [28, 33, 38, 69]. However, we show that combining CAT with our MBM-based side channel yields a practical alternative to Prime+Probe on CVMs which may be difficult to mount on modern CPUs due to non-inclusive caches and SMT, which is avoided for security reasons.

Unlike prior work [11, 48, 49, 70], we neither use nor are aided by single stepping [66]. Instead, the attacks run *live* while the CVM runs. We show that MBM tracks data flow between memory and cache at a cache-line-size granu-

larity, paving way for a malicious hypervisor to mount a Bleichenbacher attack on RSA on a QoS-unrestricted CVM, finding further that CAT amplifies this leakage. With MBM, MBA, CAT, and CDP, and their combinations, we present inter-keystroke timing attacks mounted by a malicious hypervisor on a remote user, e.g., using a thin client, across four QoS-restricted settings, whose F_1 score rises from 89.1 % (unrestricted) to 96 % (QoS-restricted) with a temporal resolution of 0.095 s. Our attack can be mounted on a remote user connected to the CVM even if they are more than 7 000 km and at least 16 network hops away.

We show that MBA rate limits on memory-to-cache bandwidth introduce a new side channel within a class of service with a covert channel capacity of 205 B/s. At unrestricted MBA (default CPU state), this side channel does not exist. We evaluate the Prime+Probe across all possible CAT restrictions, counter-intuitively finding that 5 cache ways enable a covert channel faster than 1 cache way, which was the scenario explored in prior work [38], with a maximum speed up of 2.27×. Across 18 QoS settings of CAT and MBA, we lastly evaluate a website fingerprinting attack mounted on a CVM in a top-20 open-world scenario, e.g., again using a thin client, recording traces for all 18 settings, overall taking more than 60 hours. The website-fingerprinting attack's F_1 score increases from 44.72 % (unrestricted) to 72.10 % (QoS-restricted). We conclude that these well-intentioned quality-of-service features all facilitate malicious use, with mitigation strategies most likely requiring hardware changes in future CPUs.

In summary, our main contributions are as follows:

- We investigate all five quality-of-service (QoS) features on AMD Zen 4 server CPUs (CAT, CMT, MBA, MBM, and CDP). All introduce new cross-core side channels or amplify existing ones by up to 2.27×.
- We evaluate our attacks across physical cores in a malicious or compromised hypervisor scenario, attacking a SEV-SNP CVM without single-stepping or similar fine-grained active control of the CVM.
- We mount a Bleichenbacher attack on RSA with and without QoS amplification from a malicious hypervisor, an inter-keystroke timing attack with a temporal resolution of 0.095 s and a 96 % F_1 score, and a website fingerprinting attack evaluated over 18 QoS settings in a top-20 open-world scenario attaining 72.10 % F_1 score.

Outline. Section 2 provides background. Section 3 discusses QoS features. Section 4 discusses threat models. Section 5 presents our covert channels. Section 6 presents a Bleichenbacher attack on RSA, Section 7 presents a keystroke-timing, and Section 8 presents a website-fingerprinting attack. Section 9 discusses our findings, related work, and possible mitigations. Section 10 concludes.

Responsible Disclosure. We responsibly disclosed our findings to AMD in August 2025; they deemed it out of scope, saying SEV-SNP's threat model excludes side-channel attackers. AMD announced our findings as AMD-SB-3035.

2 Background

In this section, we briefly overview hardware properties we exploit, related cache side channels, quality-of-service (QoS) mechanisms, and possible attacks.

Shared Resources and Side Channels. Modern CPUs multiplex shared microarchitectural resources across hardware threads. The last-level cache (LLC, typically L3) is set-associative and organized in 64 B cache lines and cache sets. It is physically indexed and tagged and can be inclusive or non-inclusive to private caches (L1 and L2). Server CPUs often use non-inclusive LLCs, limiting cross-core eviction primitives unless the victim's activity also affect the LLC. On AMD EPYC, cores are grouped into core complexes with an LLC slice per complex, and several controls apply at that granularity [30].

Cache side channels infer a victim's activity from attacker-observed latency differences, with different scope, preconditions, and granularity. Flush+Reload [68] and Flush+Flush [15] operate at cache-line granularity and typically require shared memory. Evict+Reload [16] and Prime+Probe [41] contend for lines that map to the same set, trading precision for broader applicability. Beyond these, new primitives explore instructions that architecturally interact with the cache state [9, 72], or manipulate cache residency without DRAM traffic [46].

Cache attacks are often used for recovering cryptographic secrets. Early work showed key extraction from table-based ciphers via cache access patterns [41], with follow-up attacks demonstrating practical AES key recovery from userspace processes [17] and across VMs [21, 35]. Outside cryptography, fine-grained activity has been inferred as well: inter-keystroke timing and keystroke detection from user applications [16], and kernel address-space randomization breaks by correlating cache effects during privileged operations [14, 18].

Quality of Service for Shared Resources. Commodity platforms ship QoS mechanisms intended for multi-tenant consolidation and fairness, on Linux via the `resctrl` interface. They can be split into two categories: Allocation and Monitoring. All QoS mechanisms are configured via model-specific registers (MSRs) by privileged software and, on AMD, are enforced per core complex.

Cache Allocation Technology. CAT partitions LLC ways among Classes of Service (CoS). With multiple VMs running, some may utilize more cache than others, e.g., due to data streaming, transcoding, data processing [1, 40, 54], resulting in degraded performance for co-located tenants. CAT can be employed to restrict logical cores of a CoS domain to specifiable L3 cache ways [3]. By enforcing cache way separation for CoS domains via CAT, privileged software can ensure that workloads or VMs of these CoS domains do not face interference by workloads in other CoS domains. Prior work [43,54,58] has shown that employing CAT can be beneficial and speed up certain workloads due to fewer LLC misses. Furthermore, prior work [34, 56] showed that CAT can serve as a mitigation against cache side-channel attacks. However, Wieczorkiewicz et al. [64] note that Intel did not intend CAT to be used as a security mechanism.

Memory Bandwidth Allocation. MBA offers a way for privileged software to limit bandwidth between L3 cache and memory for a specified Class of Service (CoS). Prior work [54] showed the benefit of enforcing MBA limits when multiple pro-

cesses simultaneously perform memory-intensive operations. On AMD systems, MBA limits are applied in multiples of 128 MiB/s [3, 10].

Code and Data Prioritization. CDP is a refinement to CAT that introduces disjoint way masks for code and data. CDP can be useful for optimizing certain types of workloads, e.g., data streaming or large codebases. Unlike CAT and MBA, each CoS is associated with two MSRs: one for code and the other for data. Enabling CDP halves the amount of CoSes, from 16 to 8.

Cache QoS Monitoring. CMT can be used by privileged software to be informed of L3 cache occupancy for a specified Resource Management Identifier (RMID). With the value reported by CMT, system software can monitor the L3 cache usage of workloads and possibly evaluate their impact on the performance of other workloads, further serving as a criterion to apply CAT allocations. AMD states that CMT values may be an approximation [3]. CMT, along with performance counters, has been explored for detection of side-channel attacks [5, 67] on Intel.

Memory Bandwidth Monitoring. MBM reports the amount of data transfer between L3 cache and memory, *i.e.*, bandwidth, of a specified Resource Management Identifier (RMID) to privileged software. There are two types of memory that MBM can track: local and total [3]. Local MBM reports a monitored thread's memory bandwidth through the memory controller of the CPU upon which the monitored thread runs. Total MBM includes the previous number, plus the bandwidth through memory controllers of other NUMA CPUs. On single CPU systems, both local and total report the same number.

Confidential Computing and Hypervisors. Side-channel attacks have always been a concern for the security of TEEs, especially for Intel SGX [52]. Modern VM-based TEEs (e.g., AMD SEV and Intel TDX) protect entire VMs instead of single applications from an untrusted host, the paradigm called *confidential computing*. Both Intel TDX [20] and AMD SEV [26] (and SEV-SNP [2]) protect the memory contents of confidential VMs (CVMs) by encrypting data in memory. To protect the guest's memory while still allowing for fast communication with the host, Intel splits the memory into a private encrypted part, only accessible by the guest and TDX module, and a shared part, accessible also by the host. Despite these protection mechanisms, attacks exploiting architectural bugs or physical properties can still lead to full compromises of the guest [60, 63, 70]. In particular, shared states can still be exploited via side channels, e.g., the cache [47], interrupts [48,49], CPU vulnerabilities [70], performance monitoring [11], I/O operations [31], power [39]. Prior work has shown to break the guarantees of memory encryption [32, 51, 65] and guarantees of data integrity via misconfiguration [8] or malicious improper initialization [50].

3 Quality-of-Service Features

In this section, we investigate the quality-of-service (QoS) features offered on an AMD EPYC 8024P (Zen 4c) [3]. We also verify that these QoS features work on an AMD EPYC 9124 (Zen 4). In Table 1, we list both Intel and AMD's QoS features. In this paper, we use the terms used by the Linux kernel [27].

Table 1: AMD QoS features and their equivalents on Intel RDT. Throughout this paper, we use the terms and abbreviations used by the Linux kernel.

AMD	Intel	Linux Kernel	Attacks
L3 Cache Allocation Enforcement	Cache Allocation Technology	Cache Allocation Technology (**CAT**)	§5, §6, §7, §8
Monitoring L3 Occupancy	Cache Monitoring Technology	Cache QoS Monitoring (**CMT**)	§8
L3 External Bandwidth Enforcement	Memory Bandwidth Allocation	Memory Bandwidth Allocation (**MBA**)	§5, §7, §8
Monitoring L3 Memory Bandwidth	Memory Bandwidth Monitoring	Memory Bandwidth Monitoring (**MBM**)	§6, §7, §8
Code and Data Prioritization	Code and Data Prioritization	Code and Data Prioritization (**CDP**)	§7

We first describe our experimental setup and the methods through which these QoS features are accessed. Afterwards, we investigate each QoS feature, following which we build and evaluate attacks in Sections 5 to 8. Table 1 provides forward references to the sections where we mount attacks with the QoS feature.

QoS Usage. All five QoS features are set by privileged software through the `PQR_ASSOC` model-specific register (MSR). This MSR exists for each logical core and contains the Class of Service (CoS) and Resource Management Identifier (RMID) [3]. The CoS is used by the allocation features (CAT, MBA, CDP), while the RMID is used by the monitoring features (CMT, MBM). AMD systems support a maximum of 16 CoSes and RMIDs [3].

Allocation limits for CAT and MBA are stored in their respective MSRs (`L3_MASK` and `L3QOS_BW_CONTROL`), with limits for a specific CoS defined at an offset equal to the CoS number from the base MSR. These per-CoS allocation-limit MSRs are defined for each core complex (CCX). Values reported by CMT and MBM are accessed through the `QM_CTR` MSR. Unlike the allocation-limit MSRs, this MSR exists per logical core, with the RMID and event (CMT or MBM) specified in the per-logical-core `QM_EVTSEL` MSR.

Experimental Setup. All investigations in this section were performed on an AMD EPYC 8024P (Zen 4c) with 8 cores, 16 threads, and 2 CCXes. We perform all experiments on an entirely isolated CCX. AMD servers have a "Periodic Directory Rinse (PDR) Tuning" BIOS option that periodically clears the coherence directory [29], invalidating cached data. We change this from the default memory-sensitive to cache-bound, reducing the rinse frequency. Discussed further in our investigation of MBM, this change eliminates noise from MBM readings, as directory rinses reload recently-invalidated data from memory.

Cache Allocation Technology. CAT enables privileged software to restrict the L3 cache available to a CoS. Our system has a 16-way set associative, non-inclusive L3 cache, split across two CCXes, each CCX with 16 MiB. In Figure 1 (zone [A]), we plot the CMT values of a core assigned to 1 cache way while reading a 10 MiB file. We see a maximum usage of 1 MiB as opposed to 2 MiB (from

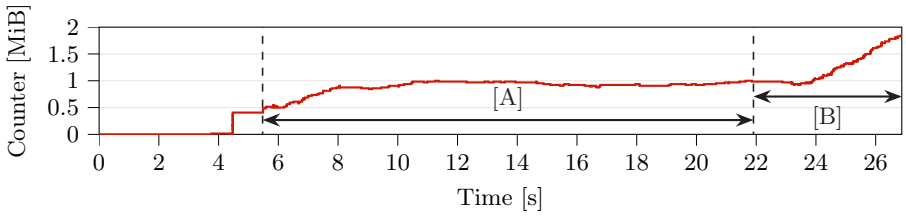


Fig. 1: A logical core restricted to a single L3 cache way with CAT can use at most 1 MiB of private cache as reported by CMT (zone [A]). However, CMT reports higher usage when accessing shared-data across different cache ways (zone [B]).

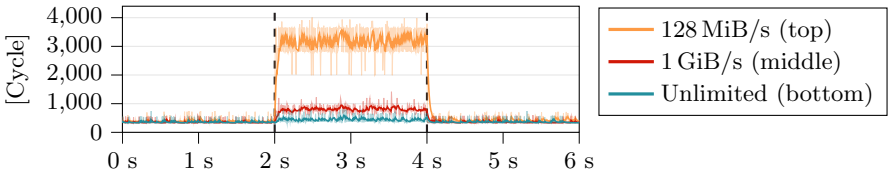


Fig. 2: One core measures single-reload latency (cycles) over time (x-axis) while another reloads cache lines for 2 s (red dashed lines). Low MBA rate limit values yield the largest latency delta: 360 cycle to 3 500 cycle at 128 MiB/s (top-most).

32 MiB ÷ 16 ways), likely due to LLC of AMD CCXes being independent [30]. In Figure 1 (zone [B]), the core assigned to 1 cache way reads data already present in different cache ways. CMT reports the core using more than 1 MiB, the allotted amount of cache ways, similar to Intel CPUs [34, 56]. Prior work [56] notes that while attacks built on shared memory [15, 68] are possible using CAT restrictions, attacks exploiting cache set and way mappings like Prime+Probe [35] in the LLC are eliminated. This observation breaks down when the attacker is privileged, and can limit and share cache ways with a victim using CAT.

Memory Bandwidth Allocation. MBA offers a way for privileged software to limit bandwidth between L3 cache and memory. Across MBA limits, we observe that hardware introduces a per-CoS rate limit. Figure 2 shows the impact of reload timings when an aggressor thread assigned to the same CoS performs memory accesses. Reloads generally require 360 cycle; however, with the aggressor, the reload time increases depending on the MBA limit. At 128 MiB/s (lowest), the average reload requires 3 500 cycle. With no limits, significantly more reloads are necessary to make a difference: the reload times increase to an average of 445 cycle with the aggressor continuously reloading 2 500 KiB.

Code and Data Prioritization. CDP allows privileged software to specify different cache ways for instructions and data. Figure 3 shows execution time of a simple, page-aligned, and unoptimized C function while streaming 128 MiB of data. The execution time ranges between 240 cycle to 264 cycle without streaming, 360 cycle to 792 cycle (average: 430 cycle) with streaming and CDP, and

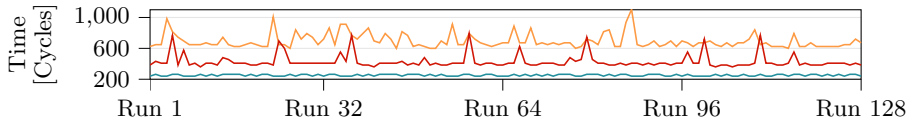


Fig. 3: Execution time of a C function while streaming 128 MiB of data: without CDP (top), with CDP (middle). The bottom-most line is the normal execution time without any data streaming, *i.e.*, only L1 cache hits.

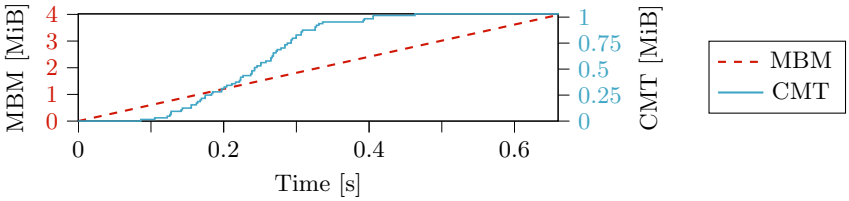


Fig. 4: CMT reports data in 16 KiB increments, while MBM returns a monotonically-increasing count of 64 B cache-line transfers.

600 cycle to 1 104 cycle (average: 695 cycle) with streaming and without CDP. We realize that CDP can make code-targeted or data-targeted LLC attacks less noisy, as there are fewer chances for a cache line to be evicted from the LLC. There have been numerous attacks demonstrated on the LLC in recent years which would benefit from much less noise [9, 15, 16, 35, 68], as such attacks requires both the attacker and victim to access the same set of cache ways.

Cache QoS Monitoring. CMT reports the amount of L3 cache occupancy per RMID to privileged software. Depicted in Figure 4, we find that CMT reports occupancy in 16 KiB increments when a thread sequentially reads 64 B of a file. We observe that CMT does not show any interesting pattern or behavior besides an increase in cache occupancy. This, coupled with an absolute limit (size of L3 cache), makes CMT challenging to use in side-channel attacks from privileged software requiring precision. In Section 8, we note CMT-based leakage leads to the lowest F_1 scores in website fingerprinting attacks.

Memory Bandwidth Monitoring. MBM reports the data transferred between L3 cache and memory. Figure 4 shows that MBM is accurate, reporting values with 64 B resolution. We find that the "Periodic Directory Rinse (PDR) Tuning" BIOS option affects MBM readings, since PDR periodically clears the coherence directory. The default option, auto, is aliased to memory-sensitive [29], where the coherence directory is rinsed at the highest frequency: every 60 ms, we observe MBM increase by 88 KiB. Setting the option to cache-bound rinses the directory at the lowest frequency [29]; in practice, we observe that the coherence directory is never rinsed. Figure 5 shows PDR Tuning set to memory-sensitive and cache-bound for two workloads: repeated reads of the same 64 B line (left)

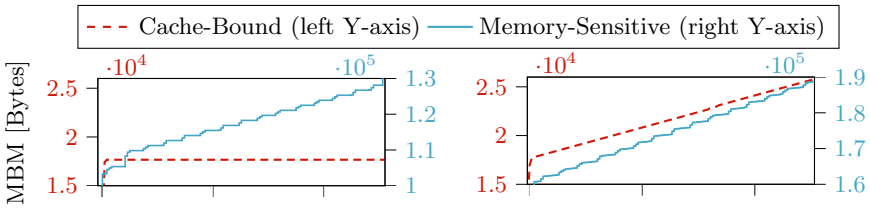


Fig. 5: MBM can detect single cache-line-size transfers between L3 cache and memory. The BIOS "Periodic Directory Rinse (PDR) Tuning" option affects MBM readings: 'memory-sensitive' PDR rinses the coherence directory every 60 ms, producing spikes of 1 408 cache lines (88 KiB).

and streaming a large file in 64 B chunks (right). To a privileged attacker, MBM is more reliable than CMT due to its precise reporting.

4 Threat Model and Evaluation Setup

We evaluate our attacks on an AMD EPYC 8024P with both the hypervisor (Linux 6.11) and SEV-SNP CVM (Linux 6.8) running Ubuntu 22.04. We assume a malicious hypervisor attacking a confidential VM (CVM), like prior work [11, 12, 48, 49, 70]. Since the hypervisor has complete control over the system, e.g., MSRs, BIOS options, scheduling, prefetcher, and QoS settings, the hypervisor is free to modify these privileged interfaces in a malicious manner. In our setup, the hypervisor sets "Periodic Directory Rinse Tuning" BIOS option to cache-bound and disables the prefetcher. Unlike prior work [11, 48, 49, 70], we neither use nor are aided by single-stepping techniques [66], and assume such possibilities are mitigated, *i.e.*, prior single-stepping-based attacks do not work.

5 Covert Channel

In this section, we evaluate our MBA- & CAT-amplified cross-core covert channel to assess the channel's capacity. Here, amplification refers to increasing the channel capacity when compared to an unrestricted case. Prior work already demonstrated CAT amplification from malicious hosts to TEEs and CVMs [38, 69]. Hence, we focus solely on the amplification effect of MBA on the covert channel, measured directly on a malicious hypervisor.

MBA. We first build a same-CoS, rate-limit MBA covert channel (see Section 3 and Figure 2). Two workloads assigned to the same CoS are spawned on two different physical cores within a core complex. We build a time-sliced, unidirectional channel, synchronized on a wallclock timer, with 1 bit transmitted per time slice: To transmit a 0, the sender remains idle, for a 1 it induces bursts of memory loads. The receiver infers the transmitted bit by measuring parallel memory load latencies. We evaluate channel capacities for various MBA limits

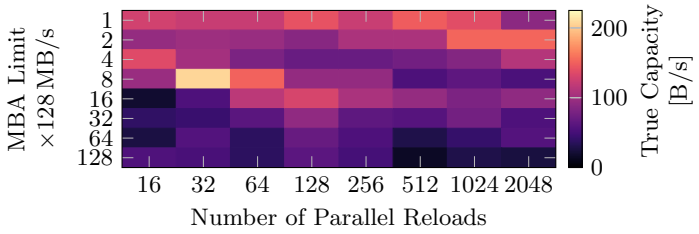


Fig. 6: True capacity (B/s) of a covert channel through MBA rate limits.

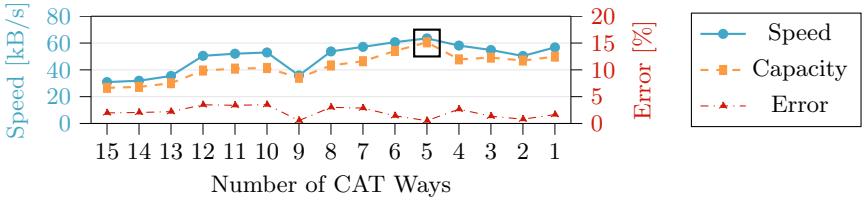


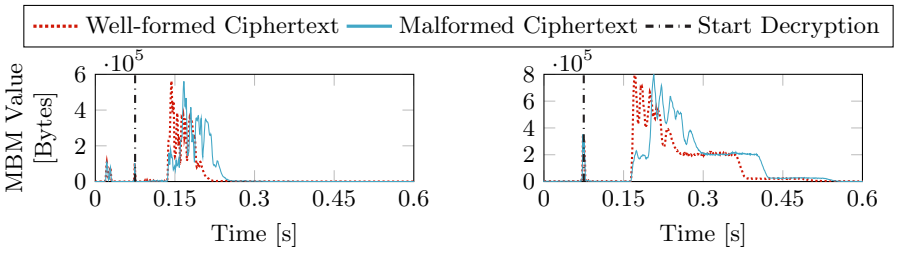
Fig. 7: CAT restrictions amplify the Prime+Probe-based covert channel, from 26.57 kB/s capacity at 15 ways, to 60.54 kB/s at 5 ways.

from the most restrictive (128 MB/s) to unlimited (2048×128 MB/s) with varied parallel reloads. For each setting, we evaluate a range of time slices by transmitting 512 B and computing the true capacity (based on the error rate), and report the best time slice for each setting in Figure 6. As expected, the capacity is lower for higher MBA limits. The maximum capacity of 205.8 B/s reaches at the 8×128 MB/s limit. Lower limits already affect sender and receiver performance and are more susceptible to noise from unrelated activity.

CAT. Prior work demonstrated CAT-amplified Prime+Probe attacks [38, 69], restricting cache ways available to sender and receiver. For completeness, we also evaluate this covert channel. Our machine has a non-inclusive L3 cache, and both parties run on different cores, *i.e.*, they do not share L1 or L2 caches. Hence, we work with eviction sets covering the L2 and L3 cache [71], with 5 cache lines more than that fits in the corresponding sets. As shown in Figure 7, limiting cache ways substantially improves the covert channel's capacity. On our machine, the capacity rises from 26.57 kB/s (1.97 % error rate) at 15 cache ways to 60.54 kB/s (0.52 % error rate) at 5 ways. We find the capacity to drop slightly at the lowest way count, possibly due to sender and receiver having too few cache lines to work with, leading to more self-evictions, noise, and slower execution.

6 Bleichenbacher Attack Exploiting MBM

In this section, we demonstrate that a malicious hypervisor can distinguish behavior within an SEV-SNP CVM, utilizing MBM. With CAT, this leakage can be amplified. We evaluate an RSA implementation which Kairo [25] recently



(a) Unrestricted CVM (16 MiB LLC) (b) CAT-enforced CVM (1 MiB LLC)

Fig. 8: Using MBM readings, a malicious hypervisor can distinguish valid from malformed RSA ciphertext decryptions in an SEV-SNP CVM, enabling Bleichenbacher attacks on RSA. Averaged over 1 000 valid and malformed decryptions, consecutive MBM readings differ when the CVM logs errors internally.

demonstrated was vulnerable to Bleichenbacher attacks [6]. In such attacks, an attacker aims to distinguish whether a ciphertext is malformed or well-formed, e.g., by the time it takes for decryption to complete. A ciphertext is well-formed if it conforms to the PKCS padding format, and malformed otherwise. We refer to the original publication [6] for a detailed explanation. Several implementations of PKCS #1 v1.5 were found to be vulnerable in 2023 [25], and optimizations in the past reduced the number of oracle calls down to 9 400 on average [4].

We hypothesize that diverging paths may not only show a timing difference, but may require a differing amount of code to execute, amplified more by any instrumentation added by developers. This difference in code, if transferred from memory to cache, could be observed using MBM. This comes with the caveat that code of neither branches should be in cache and that a distinguishable amount of each branch's code should transfer between memory and cache. This can be achieved either by evicting the cache before a branch is taken or by reducing the size of cache, e.g., with CAT, to introduce self-evicting behavior.

6.1 Experimental Setup

We choose M2Crypto [7], a Python wrapper for OpenSSL, as the attack target for a Bleichenbacher attack [25]. Upon decrypting a malformed ciphertext, OpenSSL returns an error, causing M2Crypto to raise a custom `RSAError` exception. We use the last susceptible versions of M2Crypto (0.38.0) and OpenSSL (3.0.2), as both have been patched since 2023. A developer may assume the wrapper brings no security issues, since cryptography is OpenSSL's responsibility. Furthermore, while system-wide packages may receive automatic updates [59], Python packages are often pinned for stability [23], making updates easy to overlook.

We launch a 2 vCPU CVM on two physical cores, assigning both vCPUs to the same CoS and RMID. We evaluate two scenarios: unrestricted L3 cache (16 MiB) and maximally restricted (1 MiB), with the CVM and the rest of the system assigned to mutually exclusive cache ways. Calibration between the CVM

and hypervisor is achieved with a high-resolution counting thread [53] writing to pre-shared memory [44], since hypervisor and CVM do not share the same TSC view in SEV-SNP [24]. The hypervisor randomly signals the CVM over SSH to decrypt well-formed or malformed ciphertexts, taking MBM readings every 100 µs over 1.5 s, recording 1 000 traces per ciphertext type (2 000 total).

6.2 Evaluation

Figure 8 shows the average MBM increase over 1 000 traces of well-formed and malformed ciphertexts. We observe a change in data transfers both with unrestricted operation (Figure 8a) and maximally restricted operation (Figure 8b). In both figures, the vertical dashed line indicates the decryption start.

We see that there is a difference in consecutive MBM readings between well-formed and malformed ciphertexts in both cases. It takes roughly 0.04 s for the decryption operation to complete, taking slightly longer for the CAT-restricted VM, after which we see a lateral shift form between the well-formed and malformed ciphertexts, arising from error reporting. This lateral shift is 0.024 s with the unrestricted VM and 0.036 s in the CAT-restricted VM. Thus, our attack clearly demonstrates that MBM reveals differing amounts of branch code entering the cache, with CAT extending the window by 0.012 s.

Like the original attack, the timing difference over network alone can be used as the side channel. However, SEV-SNP VMs are known to be subject to high network latency [24], making the timing attack noisy when mounted *only* over network. In such cases, the MBM-based side channel can reliably utilized with CAT as an amplifier, the attacker being a malicious hypervisor. We conclude that mitigating known attacks such as Bleichenbacher attacks can remain a challenge when the environment where the code is run changes drastically and e.g., new leakage channels are introduced inadvertently.

7 Keystroke Detection

In this section, we mount a keystroke detection attack where a malicious hypervisor attempts to detect user keypresses when connected to an SEV-SNP CVM. The hypervisor's aim is to mount an inter-keystroke timing attack [45], where the time between consecutive key presses indirectly leak information about the text being typed [28,55]. We consider a typical thin client setup where the software for employees runs on a centralized server, which may be on-premise or off-premise hosted at a cloud provider. Given the trust promises of CVMs, companies may turn towards CVMs for security and data protection reasons, especially when hosted in the cloud. The user, who is connected to the CVM with VNC, interacts with applications with the mouse and keyboard.

Although there has been prior work that explore keystroke detection within a CVM via other side channels, e.g., unprotected IO [31], our work utilizes MBM values. Figure 9 (top) and Figure 9 (middle) show the difference of consecutive MBM values when a types text into VS Code (in the CVM) and a password

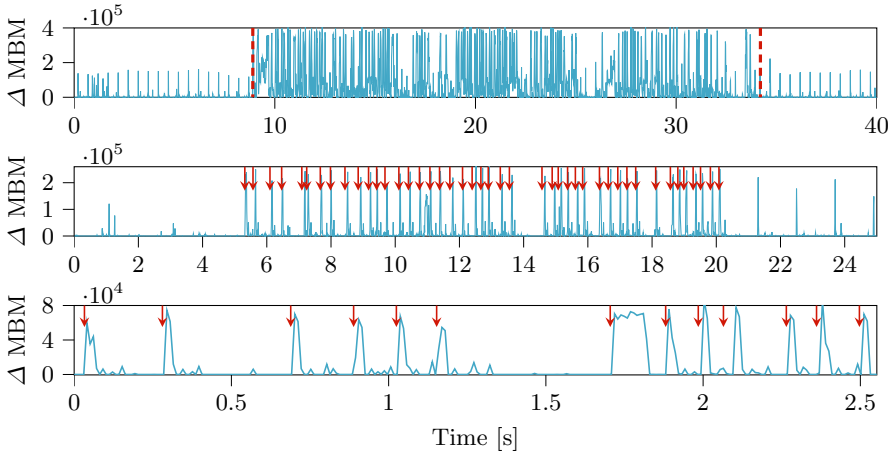


Fig. 9: Keypress detection via MBM readings on an SEV-SNP CVM, achieving 96.05 % F_1 score in gedit (maximal CAT) at a typing speed of 7.2 keys/s. Arrows mark keypresses. **Top**: VS Code (keypresses between dashed lines). **Middle**: Firefox, typing a password on x.com. **Bottom**: Gedit, 2.5 s window.

on Firefox, respectively. For both of these plots, we employed a setup where the remote user was more than 7 000 km and at least 16 hops away from the CVM, introducing a high amount of jitter in the timing channel. We observe that the key presses (arrows) still align very clearly with spikes in the difference of consecutive MBM readings, indicating that key presses cause a high amount of data to be transferred between memory and cache regardless of application.

Our experimental setup consists of three systems: 1) the remote user's system, 2) the CVM, and 3) the hypervisor. On the remote user's system, a keylogger records the ground truth. A human user types a 400-character text averaging 7.2 keys per second, in line with a skilled typist [42]. The hypervisor reads MBM values at 100 Hz. We evaluate our keystroke detection attack on gedit, a text editor, with a user remotely connected to a CVM through the VNC protocol using the Remmina desktop client. For simplicity, we assume that gedit is pinned to a specific vCPU and the hypervisor reads the MBM values of that physical core. An attacker can use profiling to determine on which core gedit runs.

We find that a high difference in consecutive MBM values indicates keypresses. Figure 9 (bottom) shows a 2.5 s window, where the arrows represent the remote user's key press. We heuristically define a threshold above which the difference in MBM is considered a key press. In Table 2, we present our results.

Without any QoS enforcement features (CAT, MBA, or CDP), the keystroke detection F_1 score is already at 89.11 % with a temporal resolution of 96 ms and an average of 34 ms ($\sigma_{\bar{x}}$=0.9) off from the ground truth. With the hypervisor enforcing CAT with the strictest restriction, the F_1 score increases to 96.05 %; here, the temporal resolution is 95 ms, with an average of 27 ms ($\sigma_{\bar{x}}$=0.9) off from

Table 2: QoS restrictions and their effects on keystroke detection in gedit. Symbols indicate maximal restriction (●) or no restriction (○).

MBA	CAT	CDP	Average [ms]	MBM Threshold	F_1 Score
●	●	○	49 ($\sigma_{\bar{x}}$=1.5)	13 000	76.58
●	○	○	51 ($\sigma_{\bar{x}}$=1.5)	14 000	76.48
○	●	○	27 ($\sigma_{\bar{x}}$=0.9)	42 000	96.05
○	○	○	34 ($\sigma_{\bar{x}}$=1.4)	34 000	89.11
○	○	●	4 ($\sigma_{\bar{x}}$=0.2)	41 000	79.64

the ground truth. We see that CAT amplifies the MBM information leakage since it reduces the LLC size, forcing more data to repeatedly flow between cache and memory. Enforcing the strictest CDP setting possible (exclusive 1 MiB for code and data each) decreases the F_1 score to 79.6 %. Interestingly, employing CDP results in observations which are the least off of ground truth (4 ms ($\sigma_{\bar{x}}$=0.2)). Contrarily, applying the strictest MBA limit reduces the F_1 score to 76.5 %. This reduction is possibly due to MBA applying a rate limit, a hindrance that negatively influences the leakage. Applying CAT *and* MBA still results in a F_1 score of 76.6 %, indicating that MBA significantly worsens the side channel.

8 Website Fingerprinting

In this section, we show that a malicious hypervisor can utilize QoS features to perform a top-20 open-world website fingerprinting attack on an SEV-SNP CVM. Similar to the premise in Section 7, we assume a scenario in which employees connect their thin clients to CVMs in the cloud. Once connected, they carry out their work, which may include visiting certain websites using the browser within the CVM. The malicious hypervisor's goal is to distinguish which website the user visits using only QoS features: monitoring features to leak information and enforcement features to amplify it. We evaluate this scenario over 6 CAT restrictions and 11 MBA limits, along with 1 unrestricted case. Given the 18× measurements, we opted for a smaller top-20 open-world scenario to keep recording time reasonable on our shared system, totaling between 60 h to 65 h.

8.1 Experimental Setup

We launch a 3 vCPU SEV-SNP VM on isolated physical cores, with Firefox 141.0 pinned to two cores and the third reserved for system use. The hypervisor assigns the two Firefox cores to a single RMID and CoS, while the third to a separate one. In practice, a malicious hypervisor would assign the entire CVM to one RMID and CoS and filter background system noise. The hypervisor signals the CVM to open a website and records CMT and MBM values at 100 Hz.

We collect traces for the top-20 websites from semrush.com and 600 open-world websites from DataForSEO.com, with 30 readings per top-20 site over 10 s

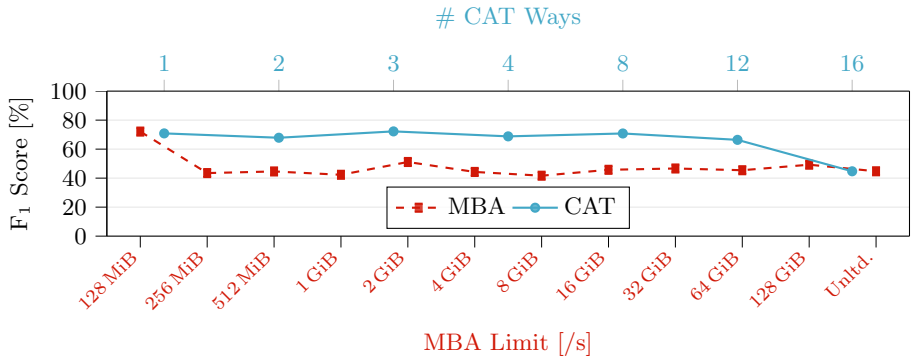


Fig. 10: F_1 score of top-20 open-world website fingerprinting with the MBM difference feature for varying CAT ways and MBA limits.

each. Open-world traces are recollected per-QoS-restriction to avoid restriction-specific bias. Within the CVM, xdotool automates Firefox navigation, restarting the browser every 10 websites to eliminate artifacts while the website order is randomized to remove any bias. We repeat the experiment across 11 MBA limits, 6 CAT restrictions, and 1 unrestricted baseline (18 runs total). For CAT experiments, the hypervisor and CVM are assigned mutually exclusive cache ways, except in the unrestricted case where the CVM receives all 16 ways.

8.2 Results

From the time-series data, we define five features: the difference in MBM and CMT values from the initial value, whether the consecutive MBM and CMT values increased or decreased, and the absolute value of CMT. Absolute MBM is excluded as it is monotonically increasing. We train each feature individually using a KNN time series classifier [57], with an 80-20 train-test split. We also tried random forest classifier, but found it to perform worse, regardless of PCA.

Of the five, the difference in MBM values from the initial value results in the highest F_1 scores, shown in Figure 10 with the 3-nearest neighbors. Without any QoS enforcement, we observe an F_1 score of 44.72 %, better than random guessing at 5 %. Similar to keystroke detection, we observe that restricting cache ways tends to a higher F_1 score, between 66.4 % to 77.23 %. In contrast, we achieve the highest 72.10 % F_1 score with the most-restrictive MBA limit. With the absolute CMT values, we find the highest F_1 scores to range between 43 % to 49 % with CAT. We find the features based on consecutive MBM and CMT values to perform the worst, attaining a maximum F_1 score of 36 % with CAT.

Performance Impact. At stricter MBA limits, we find that browsing experience within the CVM is negatively affected. According to Speedometer3.0, the browser performs 10× faster with unlimited MBA than with the most-restrictive limit. CAT restrictions have less impact than MBA limits; the benchmark reports only 2× faster with all cache ways versus one cache way.

9 Discussion & Related Work

We demonstrated how quality-of-service features enable and amplify microarchitectural side-channel attacks. Neither Intel nor AMD consider side channels a security concern and instead they simply refer users to "best practices" against side channels, e.g., constant-time principles [2]. Prior work [11] showed that this advice reaches its limits when combined with the idea to run entire unmodified VM images as CVMs in a trusted-execution environment.

Related to our use of *monitoring* QoS features (CMT and MBM) is the work by Gast et al. [11] who showed how a malicious hypervisor can access *performance counters* to leak from guest VMs on Zen 3/4 architectures. AMD has since virtualized performance counters, mitigating the attack. However, there is no such possibility to virtualize QoS monitoring features. Unlike Gast et al. [11], we run the attack live without single-stepping, which slows the CVM and is easily detectable, already addressed by some implementations, e.g., Intel TDX [47]. Allocation-based attacks can be detected through the performance degradation of cache or memory bandwidth, but monitoring-based attacks are completely stealthy as they do not interact with the CVM at all.

Minkin and Kasikci [38] showed that CAT can be used to limit the cache ways available to an Intel SGX enclave, thus reducing noise in Prime+Probe attacks. We confirm that this is a viable approach, yielding higher capacities in our setting. Therefore, we believe that our work complements theirs by exploring a different threat model and a wider range of QoS features.

Mitigations. QoS features are crucial for cloud providers to load-balance and detect anomalies, potentially even for security [1, 34, 36, 56]. Furthermore, without performance counter data [3, 20] and QoS features available to the system, attackers can hide malicious workloads [13, 22, 52, 61, 73] that would otherwise be evident in the monitored performance data. One possibility to mitigate QoS *monitoring* features (CMT, MBM) is to make them more coarse-grained, using the strategy outlined by Weissteiner et al. [62]. Fuzzy increments and aggregation windows can obscure readings, leaking insufficient information for attacks, while still providing enough information for resource provisioning.

On the other hand, *allocation* QoS features (CAT, MBA, CDP) require careful mitigation: since AMD SEV CVMs already use Address Space Identifiers (ASIDs) internally, hardware could assign cache ways per-ASID instead of per-CoS when SEV is active, preventing hypervisor CAT- and CDP-based Prime+Probe. To limit MBA covert channels, hardware could enforce MBA limits per-CoS *and* per-core, isolating cores even within the same CoS.

10 Conclusion

We showed that all well-intentioned quality-of-service features (CAT, CMT, MBA, MBM, and CDP) on modern AMD server CPUs introduce or amplify cross-core side channels in SEV-SNP confidential computing environments. We showed how a malicious or compromised hypervisor can utilize QoS features to

leak information of a CVM without single stepping or similar fine-grained active control of the CVM. We showed that attack performance is improved in covert channels (2.2×), an Bleichenbacher attack on RSA, keystroke detection (96 % F_1 score), and website fingerprinting (72.1 % F_1 score). Our results clearly show that the introduction of QoS features must be considered with care when designing CPUs, especially when side channels are a concern, and future CPUs should consider the implications of QoS features on security when designing TEEs.

Acknowledgements. We thank our anonymous reviewers across conferences for their feedback. We additionally would like to thank Smriti Suresh, Stefan Gast, Thomas Lienbacher, Fabian Rauscher, Lukas Giner, Jonas Juffinger, and Lorenz Schumm for the discussions and inputs at various stages of this work. This research is supported in part by the European Research Council (ERC project FSSec 101076409), the Austrian Science Fund (FWF SFB project SPyCoDe 10.55776/F85), and the National Research Center for Applied Cybersecurity ATHENE. Additional funding was provided by a generous gift from Intel. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding parties.

References

1. Alibaba Cloud: Enable resource isolation based on the L3 cache and MBA (2024), https://www.alibabacloud.com/help/en/ack/ack-managed-and-ack-dedicated/user-guide/resource-isolation-based-on-the-l3-cache-and-mba
2. AMD: AMD SEV-SNP: Strengthening VM Isolation with Integrity Protection and More (2020), https://www.amd.com/content/dam/amd/en/documents/epyc-business-docs/white-papers/SEV-SNP-strengthening-vm-isolation-with-integrity-protection-and-more.pdf
3. AMD: AMD64 Architecture Programmer's Manual (2024)
4. Bardou, R., Focardi, R., Kawamoto, Y., Simionato, L., Steel, G., Tsay, J.K.: Efficient Padding Oracle Attacks on Cryptographic Hardware. In: CRYPTO (2012). https://doi.org/10.1007/978-3-642-32009-5_36
5. Bazm, M.M., Sautereau, T., Lacoste, M., Sudholt, M., Menaud, J.M.: Cache-Based Side-Channel Attacks Detection through Intel Cache Monitoring Technology and Hardware Performance Counters. In: Fog and Mobile Edge Computing (FMEC) (2018)
6. Bleichenbacher, D.: Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1. In: CRYPTO (1998)
7. Cepl, M.: sourcehut – mcepl/m2crypto: OpenSSL bindings for Python (2025), https://sr.ht/~mcepl/m2crypto/
8. De Meulemeester, J., Wilke, L., Oswald, D., Eisenbarth, T., Verbauwhede, I., Van Bulck, J.: BadRAM: Practical Memory Aliasing Attacks on Trusted Execution Environments. In: S&P (2025)
9. Disselkoen, C., Kohlbrenner, D., Porter, L., Tullsen, D.: Prime+Abort: A Timer-Free High-Precision L3 Cache Attack using Intel TSX. In: USENIX Security (2017)

10. Fiedler, C., Juffinger, J., Neela, S.R., Heckel, M., Weissteiner, H., Yağlıkçı, A.G., Adamsky, F., Gruss, D.: Memory Band-Aid: A Principled Rowhammer Defense-in-Depth. In: NDSS (2026)
11. Gast, S., Weissteiner, H., Schröder, R.L., Gruss, D.: CounterSEVeillance: Performance-Counter Attacks on AMD SEV-SNP. In: NDSS (2025)
12. Giner, L., Neela, S.R., Gruss, D.: Cohere+Reload: Re-enabling High-Resolution Cache Attacks on AMD SEV-SNP. In: DIMVA (2025)
13. Gruss, D., Lipp, M., Schwarz, M., Genkin, D., Juffinger, J., O'Connell, S., Schoechl, W., Yarom, Y.: Another Flip in the Wall of Rowhammer Defenses. In: S&P (2018)
14. Gruss, D., Maurice, C., Fogh, A., Lipp, M., Mangard, S.: Prefetch Side-Channel Attacks: Bypassing SMAP and Kernel ASLR. In: CCS (2016)
15. Gruss, D., Maurice, C., Wagner, K., Mangard, S.: Flush+Flush: A Fast and Stealthy Cache Attack. In: DIMVA (2016)
16. Gruss, D., Spreitzer, R., Mangard, S.: Cache Template Attacks: Automating Attacks on Inclusive Last-Level Caches. In: USENIX Security (2015)
17. Gullasch, D., Bangerter, E., Krenn, S.: Cache Games – Bringing Access-Based Cache Attacks on AES to Practice. In: S&P (2011)
18. Hund, R., Willems, C., Holz, T.: Practical Timing Side Channel Attacks against Kernel Space ASLR. In: S&P (2013)
19. Intel: Intel 64 and IA-32 Architectures Software Developer's Manual, Volume 3 (3A, 3B & 3C): System Programming Guide (2024)
20. Intel: Intel Trust Domain Extensions Module Base Architecture Specification (2024), `https://www.intel.com/content/www/us/en/developer/tools/trust-domain-extensions/documentation.html`
21. Irazoqui, G., Inci, M.S., Eisenbarth, T., Sunar, B.: Wait a minute! A fast, Cross-VM attack on AES. In: RAID (2014)
22. Jang, Y., Lee, J., Lee, S., Kim, T.: SGX-Bomb: Locking Down the Processor via Rowhammer Attack. In: SysTEX (2017)
23. jazzband: pip-tools (2025), `https://github.com/jazzband/pip-tools`
24. Juffinger, J., Neela, S.R., Gruss, D.: Not So Secure TSC. In: ACNS (2025)
25. Kairo, H.: Everlasting ROBOT: The Marvin Attack. In: ESORICS (2023)
26. Kaplan, D., Powell, J., Woller, T.: AMD Memory Encryption (2016)
27. Kernel, T.L.: User Interface for Resource Control feature (resctrl) (2025), `https://docs.kernel.org/filesystems/resctrl.html`
28. Kurth, M., Gras, B., Andriesse, D., Giuffrida, C., Bos, H., Razavi, K.: NetCAT: Practical Cache Attacks from the Network. In: S&P (5 2020)
29. Lenovo: Tuning UEFI Settings for Performance and Energy Efficiency on 4th Gen AMD EPYC Processor-Based ThinkSystem Servers (2024)
30. Li, D., Zhu, Z., Shen, J., Zhang, Y., Shi, G., Meng, D.: Flush+Revisit: A Cross-CCX Side-Channel Attack on AMD Processors. In: TrustCom (2023)
31. Li, M., Zhang, Y., Lin, Z., Solihin, Y.: Exploiting Unprotected I/O Operations in AMD's Secure Encrypted Virtualization. In: USENIX Security (2019)
32. Li, M., Zhang, Y., Wang, H., Li, K., Cheng, Y.: CIPHERLEAKS: Breaking Constant-time Cryptography on AMD SEV via the Ciphertext Side Channel. In: USENIX Security (2021)
33. Lipp, M., Aga, M.T., Schwarz, M., Gruss, D., Maurice, C., Raab, L., Lamster, L.: Nethammer: Inducing Rowhammer Faults through Network Requests. In: SILM Workshop (2020)
34. Liu, F., Ge, Q., Yarom, Y., Mckeen, F., Rozas, C., Heiser, G., Lee, R.B.: CATalyst: Defeating Last-Level Cache Side Channel Attacks in Cloud Computing. In: HPCA (2016)

35. Liu, F., Yarom, Y., Ge, Q., Heiser, G., Lee, R.B.: Last-Level Cache Side-Channel Attacks are Practical. In: S&P (2015)
36. Makrani, H.M., Sayadi, H., Nazari, N., Khasawneh, K.N., Sasan, A., Rafatirad, S., Homayoun, H.: Cloak & Co-locate: Adversarial Railroading of Resource Sharing-based Attacks on the Cloud. In: Secure and Private Execution Environment Design (SEED) (2021)
37. Maurice, C., Weber, M., Schwarz, M., Giner, L., Gruss, D., Alberto Boano, C., Mangard, S., Römer, K.: Hello from the Other Side: SSH over Robust Cache Covert Channels in the Cloud. In: NDSS (2017)
38. Minkin, M., Kasikci, B.: ZipChannel: Cache Side-Channel Vulnerabilities in Compression Algorithms. In: Dependable Systems and Networks (DSN) (2024)
39. Mishra, N., Arya, K., Bhattacharya, S., Saxena, P., Mukhopadhyay, D.: "OOPS!": Out-Of-Band Remote Power Side-Channel Attacks on Intel SGX and TDX. In: Design Automation Conference (DAC) (2025)
40. Nguyen, K.T.: Introduction to Cache Allocation Technology in the Intel®Xeon®Processor E5 v4 Family (2016), https://www.intel.com/content/www/us/en/developer/articles/technical/introduction-to-cache-allocation-technology.html
41. Osvik, D.A., Shamir, A., Tromer, E.: Cache Attacks and Countermeasures: the Case of AES. In: CT-RSA (2006)
42. Pinet, S., Ziegler, J.C., Alario, F.X.: Typing Is Writing: Linguistic Properties Modulate Typing Execution. Psychon Bull Rev **23**(6), 1898–1906 (4 2016)
43. Pons, L., Selfa, V., Sahuquillo, J., Petit, S., Pons, J.: Improving System Turnaround Time with Intel CAT by Identifying LLC Critical Applications. In: European Conference on Parallel Processing (2018)
44. QEMU: Inter-VM Shared Memory device (2025), https://www.qemu.org/docs/master/system/devices/ivshmem.html
45. Qiu, M., Chuang, L., Kim, D., Qu, H., Chen, T., Kwong, A.: KeyTAR: Practical Keystroke Timing Attacks and Input Reconstruction. In: S&P (2026)
46. Rauscher, F., Fiedler, C., Kogler, A., Gruss, D.: A Systematic Evaluation of Novel and Existing Cache Side Channels. In: NDSS (2025)
47. Rauscher, F., Wilke, L., Weissteiner, H., Eisenbarth, T., Gruss, D.: TDXploit: Novel Techniques for Single-Stepping and Cache Attacks on Intel TDX. In: USENIX Security (2025)
48. Schlüter, B., Sridhara, S., Bertschi, A., Shinde, S.: WeSee: Using Malicious# VC Interrupts to Break AMD SEV-SNP. In: S&P (2024)
49. Schlüter, B., Sridhara, S., Kuhne, M., Bertschi, A., Shinde, S.: Heckler: Breaking confidential vms with malicious interrupts. In: USENIX Security (2024)
50. Schlüter, B., Shinde, S.: RMPocalypse: How a Catch-22 Breaks AMD SEV-SNP. In: CCS (2025)
51. Schlüter, B., Wech, C., Shinde, S.: Heracles: Chosen Plaintext Attack on AMD SEV-SNP. In: CCS (2025)
52. Schwarz, M., Gruss, D., Weiser, S., Maurice, C., Mangard, S.: Malware Guard Extension: Using SGX to Conceal Cache Attacks. In: DIMVA (2017)
53. Schwarz, M., Weiser, S., Gruss, D., Maurice, C., Mangard, S.: Malware Guard Extension: Using SGX to Conceal Cache Attacks. In: DIMVA (2017)
54. Sohal, P., Bechtel, M., Mancuso, R., Yun, H., Krieger, O.: A Closer Look at Intel Resource Director Technology (RDT). In: Real-Time Networks and Systems (RTNS) (2022)
55. Song, D.X., Wagner, D., Tian, X.: Timing Analysis of Keystrokes and Timing Attacks on SSH. In: USENIX Security (2001)

56. Sprabery, R., Evchenko, K., Raj, A., Bobba, R.B., Mohan, S., Campbell, R.: Scheduling, Isolation, and Cache Allocation: A Side-Channel Defense. In: International Conference on Cloud Engineering (2018)
57. Tavenard, R., Faouzi, J., Vandewiele, G., Divo, F., Androz, G., Holtz, C., Payne, M., Yurchak, R., Rußwurm, M., Kolar, K., Woods, E.: Tslearn, A Machine Learning Toolkit for Time Series Data. Journal of Machine Learning Research (2020)
58. Tootoonchian, A., Panda, A., Lan, C., Walls, M., Argyraki, K., Ratnasamy, S., Shenker, S.: ResQ: Enabling SLOs in Network Function Virtualization. In: NSDI (2018)
59. Ubuntu Server Documentation: Automatic updates (2025), `https://documentation.ubuntu.com/server/how-to/software/automatic-updates/`
60. Wang, W., Li, M., Zhang, Y., Lin, Z.: PwrLeak: Exploiting Power Reporting Interface for Side-Channel Attacks on AMD SEV. In: DIMVA (2023)
61. Weiser, S., Mayr, L., Schwarz, M., Gruss, D.: SGXJail: Defeating Enclave Malware via Confinement. In: RAID (2019)
62. Weissteiner, H., Rauscher, F., Schröder, R.L., Juffinger, J., Gast, S., Wichelmann, J., Eisenbarth, T., Gruss, D.: TEEcorrelate: An Information-Preserving Defense against Performance Counter Attacks on TEEs. In: USENIX Security (2025)
63. Werner, J., Mason, J., Antonakakis, M., Polychronakis, M., Monrose, F.: The severest of them all: Inference attacks against secure virtual enclaves. In: AsiaCCS (2019)
64. Wieczorkiewicz, P., Branco, R., Lee, B.: On the Effectiveness of Intel's CAT as a Side Channel Mitigation. In: LangSec Workshop (2025)
65. Wilke, L., Wichelmann, J., Morbitzer, M., Eisenbarth, T.: SEVurity: No Security Without Integrity–Breaking Integrity-Free Memory Encryption with Minimal Assumptions. In: S&P (2020)
66. Wilke, L., Wichelmann, J., Rabich, A., Eisenbarth, T.: SEV-Step: A Single-Stepping Framework for AMD-SEV. In: CHES (2024)
67. Yao, F., Fang, H., Doroslovački, M., Venkataramani, G.: COTSknight: Practical Defense against Cache Timing Channel Attacks using Cache Monitoring and Partitioning Technologies. In: Hardware Oriented Security and Trust (HOST) (2019)
68. Yarom, Y., Falkner, K.: Flush+Reload: a High Resolution, Low Noise, L3 Cache Side-Channel Attack. In: USENIX Security (2014)
69. Zhang, R., Cheu, A., Gascon, A., Moghimi, D., Schoppmann, P., Schwarz, M., Suciu, O.: SNPeek: Side-Channel Analysis for Privacy Applications on Confidential VMs. In: NDSS (2026)
70. Zhang, R., Gerlach, L., Weber, D., Hetterich, L., Lü, Y., Kogler, A., Schwarz, M.: CacheWarp: Software-based Fault Injection using Selective State Reset. In: USENIX Security (2024)
71. Zhang, R., Hornetz, T., Gerlach, L., Schwarz, M.: Taming the Linux Memory Allocator for Rapid Prototyping. In: DIMVA (2025)
72. Zhang, R., Kim, T., Weber, D., Schwarz, M.: (M)WAIT for It: Bridging the Gap between Microarchitectural and Architectural Side Channels. In: USENIX Security (2023)
73. Zhang, Z., Zhang, X., Li, Q., Sun, K., Zhang, Y., Liu, S., Liu, Y., Li, X.: See through Walls: Detecting Malware in SGX Enclaves with SGX-Bouncer. In: AsiaCCS (2021)