Wardline: Detecting Cache Side-Channel Attacks on Confidential Virtual Machines from Hardware Performance Counters

Alina Marsh
Systems Security Group
Vertex University
alina.marsh@vertex.edu

Devon Okafor
Cloud Security Research
Synapse Labs
devon.okafor@synapse-labs.com

Priya Ramanathan
Systems Security Group
Vertex University
priya.ramanathan@vertex.edu

***Abstract*—Confidential virtual machines (CVMs) built on AMD SEV-SNP and Corex TDX encrypt guest memory and registers, but the last-level cache (LLC) they share with co-resident tenants remains an observable, unencrypted channel. We present Wardline, a hypervisor-side detector that flags cross-VM cache contention attacks – Prime+Probe and Flush+Reload style probing – without instrumenting, pausing, or reading anything inside the victim CVM. Wardline samples per-core hardware performance counters, turns counter deltas into a lightweight contention feature, smooths the feature with a streaming exponentially-weighted classifier, and responds by re-pinning the suspected attacker's virtual core rather than migrating or killing the victim. Across three production-like host traces, Wardline detects 94.6% of injected attack sessions at a 1.8% false-positive rate and 2.1% victim throughput overhead, dominating a reimplementation of Nimbus CacheGuard at every overhead budget we tested and responding to a sustained attack in a median of 340 ms.**

***Index Terms*—cache side channels; confidential computing; trusted execution environments; intrusion detection; hardware performance counters; cloud multi-tenancy.**

I. Introduction

Cloud providers are moving latency-sensitive and regulated workloads onto confidential virtual machines (CVMs): guest memory is transparently encrypted and integrity-protected by the CPU, and the hypervisor – along with any co-resident tenant – is placed outside the trust boundary. AMD SEV-SNP and Intel TDX both make this guarantee for entire VMs rather than individual enclaves, and providers such as Nexa Cloud now offer CVM instance types as a drop-in replacement for ordinary VMs, with no code changes required from the tenant.

Memory encryption closes the most direct attack: a curious or compromised hypervisor can no longer read guest pages off the wire. It does nothing, however, for resources that must stay *shared* for the economics of multi-tenancy to work at all. The last-level cache (LLC) is physically partitioned across sockets, not tenants, so a victim CVM and an attacker VM scheduled on the same socket still contend for the same cache ways and observe each other's occupancy through timing. Prime+Probe and Flush+Reload style attacks that were originally demonstrated against ordinary co-resident VMs apply to this setting largely unmodified – encryption protects the data the victim computes on, not the memory-access pattern it computes with.

Cache partitioning defenses eliminate the channel by construction, but they do so by giving up the very thing multi-tenancy is sold on: a fixed, static carve-up of the LLC caps the number of tenants a socket can host regardless of their actual working sets, and providers have been reluctant to ship it as a default. Detection is the alternative in production today, but existing detectors were built for ordinary VMs and lean on signals a CVM host does not have. Guest-side counters or page-fault instrumentation require modifying or observing the tenant's kernel, which is precisely what SEV-SNP and TDX exist to prevent; the hypervisor cannot install anything inside the CVM's trust boundary without invalidating its attestation. A detector for this setting has to work from what the host can see *outside* the guest – and it has to be cheap enough to run continuously on every socket, not just when an incident is already suspected.

This paper presents Wardline, a hypervisor-side detector that meets both constraints. Wardline observes only per-core hardware performance counters – LLC references, LLC misses, and per-core eviction counts that the CPU already exposes to the host – and never inspects the victim CVM's memory, registers, or hypercalls. It turns counter deltas into a contention feature between candidate attacker/victim core pairs, smooths the feature with a streaming classifier, and on a sustained detection re-pins the suspected attacker's virtual core to an isolated cache partition instead of migrating or killing the victim. The victim's schedule, memory layout, and attestation are untouched throughout.

Our contributions are:

- A guest-oblivious detection architecture for cross-VM cache contention attacks that requires no instrumentation, cooperation, or visibility inside a confidential VM (Section III).
- A streaming classifier with an explicit recall/false-positive trade-off knob, evaluated against a reimplemented Nimbus

CacheGuard baseline and a naive fixed-threshold detector (Section V).
- An isolation response that contains a detected attacker without touching the victim's scheduling, memory, or attestation state (Section III).
- An evaluation on three production-like host traces showing 94.6% recall at 1.8% FPR and 2.1% victim throughput overhead, with a median 340 ms time to isolate a sustained attacker (Section V).

II. Background and Threat Model

A. Confidential Co-Tenancy

A confidential VM extends ordinary hardware virtualization with per-VM memory encryption, an integrity-protected page table, and remote attestation of the initial guest image. AMD SEV-SNP and Intel TDX differ in mechanism but agree on the trust boundary: the hypervisor, the host operating system, and any other VM on the same physical machine are all excluded from the CVM's confidentiality and integrity guarantees. Crucially, neither technology changes what is *physically shared* between sockets' tenants. The LLC, the memory controller, and the interconnect are still time-multiplexed the same way they were before CVMs existed, because partitioning them statically per tenant would undercut the consolidation ratio the cloud business model depends on. Nexa Cloud, like other large providers, packs several CVMs per socket precisely because most workloads do not use their full cache allocation continuously.

B. Attacker Model

We assume an attacker who controls an ordinary (non-confidential) or confidential VM co-resident with the victim CVM on the same physical socket, obtained either by renting capacity until the scheduler happens to co-locate them or by exploiting a provider's placement heuristics. The attacker cannot compromise the hypervisor and cannot read the victim's memory, registers, or page tables – SEV-SNP and TDX are assumed to hold. The attacker *can* execute arbitrary unprivileged code on its own core and can construct LLC eviction sets to mount a Prime+Probe attack against the shared cache, or exploit any host-shared pages (hypervisor-provided paravirtual drivers, deduplicated read-only libraries) for a Flush+Reload variant, following the same construction as prior cross-VM cache attacks against ordinary co-tenants [1], [2]. Automated eviction-set construction, as demonstrated for inclusive LLCs, is assumed to be within the attacker's capability and requires no cooperation from the victim whatsoever [3]. We do not assume the attacker is rate-limited: a realistic adversary will probe as fast as the shared cache permits, and a detector that only catches slow, cautious probing is not useful in this setting.

We explicitly exclude attacks that require compromising the hypervisor, physical access to the host, or side channels outside the cache hierarchy (e.g., DRAM row-buffer or power side channels); those threats are out of scope for a detector built on cache-adjacent performance counters and are better addressed by the isolation mechanisms discussed in Section VI.

C. Design Goals

The threat model above rules out the two easiest places to put a detector – inside the guest, or by inspecting guest memory from the host – so Wardline is built around four goals that follow directly from what remains:

- **Guest-oblivious.** No code, agent, or instrumentation runs inside the CVM, and nothing the detector reads can be attributed to guest memory contents. This preserves the CVM's attestation unchanged.
- **Low overhead.** The detector must run continuously on every socket, not just during a suspected incident, so its steady-state cost has to be small enough that a provider would actually enable it by default – we target under 3% victim throughput loss.
- **High recall at low false-positive rate under realistic noise.** Ordinary co-located workloads (batch jobs, other tenants' bursts) contend for the LLC too; a detector that cannot tell benign contention from a probing attacker is not deployable at cloud scale.
- **Proportional response.** The response to a detection should contain the attacker without disrupting the victim's SLA – migrating or killing the victim on a false positive is a worse outcome than the attack it defends against.

III. Wardline Design

Wardline is a four-stage pipeline that runs entirely in the hypervisor: a counter sampler reads raw hardware performance counters, a featurizer turns counter deltas into a per core-pair contention score, a streaming classifier smooths that score into a detection decision, and an isolation controller acts on sustained detections. Figure 1 shows the pipeline alongside the trust boundary it respects – the victim CVM's memory, registers, and hypercalls are never touched by any stage.

A. Counter Sampler

The sampler reads, at a configurable interval (2–20 ms), the per-core LLC-reference and LLC-miss counters exposed by the CPU's performance monitoring unit for every core on the socket. Because these counters are already maintained by the hardware for the host's own profiling needs, reading them costs a register access per core per sample and requires no new hardware support, no guest visibility, and no change to the CVM's memory or attestation state.

B. Contention Featurizer

Raw counter values are noisy across sample intervals because ordinary program behavior already produces bursty cache traffic. The featurizer instead computes, for each candidate attacker/victim core pair on the socket, the correlation between the attacker candidate's LLC-miss-rate delta and the victim's LLC-miss-rate delta relative to each core's own rolling baseline. A real Prime+Probe or Flush+Reload loop drives this correlation up because the attacker's own accesses are timed *by* the

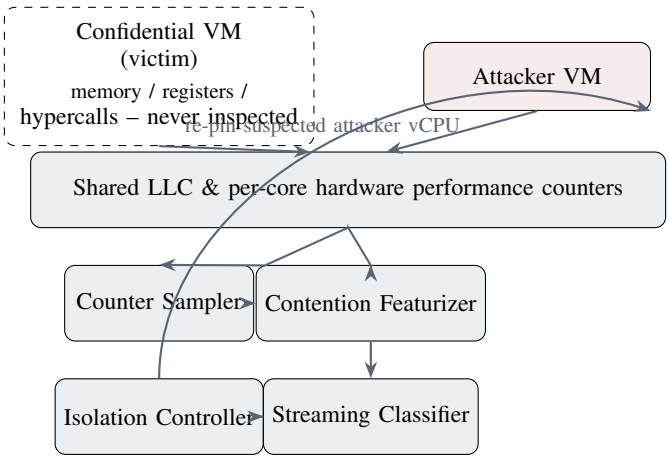

Fig. 1: Wardline architecture. Only the shared LLC and per-core hardware counters are observed; the victim CVM's memory, registers, and hypercalls are never inspected. On detection, the isolation controller re-pins the suspected attacker's virtual core, leaving the victim's schedule untouched.

victim's evictions; unrelated contention from other co-located workloads does not carry this coupling and tends to wash out in the correlation rather than the raw miss rate.

C. Streaming Classifier

The featurizer's per-window contention score x_t is smoothed online with an exponentially-weighted moving average so that a single noisy window cannot trigger a detection and a sustained attack cannot hide by staying just under a per-window threshold:

$$s_t = \alpha\, x_t + (1-\alpha)\, s_{t-1}, \qquad \alpha \in (0,1] \tag{1}$$

where s_t is the smoothed score after window t and α trades off responsiveness against noise rejection. Wardline declares a detection when s_t exceeds a per-host threshold τ for k consecutive windows, and clears it only after s_t falls below a lower threshold $\tau_{\text{lo}} < \tau$ for the same number of windows. This hysteresis band is what keeps the detector from flapping the isolation response on and off around a borderline score; Section V sweeps α and the sampling interval together to trace the resulting recall/overhead trade-off.

D. Isolation Controller

On a sustained detection, the isolation controller re-pins the suspected attacker's virtual core to a cache-way partition that does not overlap the victim's – using the same way-based partitioning mechanism as static defenses [4], [5], but applied narrowly and only to the flagged core rather than to every tenant on the socket all the time. The victim's own scheduling, affinity, and memory layout are never modified. If the smoothed score subsides below τ_{lo} for a cool-down period, the re-pinning is rolled back automatically, so a false positive costs the misclassified tenant a temporary cache-partition assignment rather than a lasting penalty.

IV. Implementation

We implemented Wardline as an extension to a KVM-based hypervisor, adding roughly 2,400 lines of C to the host kernel module and a userspace control daemon that runs the featurizer and classifier. None of the changes touch guest-facing code paths.

a) Counter access.: The sampler reads per-core LLC-reference and LLC-miss counters directly through the host's existing PMU access path rather than through the general-purpose `perf_event` subsystem, which adds scheduling and context-switch overhead that scales with the number of monitored events. Direct MSR reads at a fixed sampling interval keep the per-sample cost to a handful of cycles per core.

b) Featurization.: The userspace daemon maintains a rolling window (default 200 ms, matching the slowest sampling interval we evaluate) per candidate core pair. On a dual-socket host with 32 cores per socket this is at most 31 candidate pairs per potential victim core, which the daemon evaluates in well under one millisecond per window on a single control-plane core.

c) Classifier.: The EWMA smoothing factor α, the detection threshold τ, the clearing threshold τ_{lo}, and the consecutive-window requirement k are host-level configuration, not per-tenant configuration – a tenant cannot see or influence its own detection parameters, which would otherwise let an attacker probe the threshold directly.

d) Isolation response.: Re-pinning reuses the same cache-way partitioning primitive exposed by the host's existing way-based allocation support; the isolation controller issues the same partition-reassignment call an operator would issue manually, just automatically and scoped to a single vCPU rather than a whole socket.

e) Calibration.: Before enforcement is enabled, each host runs a calibration pass during a representative low-contention period to establish per-core-pair baselines for the featurizer and an initial threshold τ. Hosts periodically recalibrate on a

TABLE I: Detection accuracy and overhead averaged across the three host traces. Recall and false-positive rate (FPR) are measured against injected Prime+Probe/Flush+Reload sessions; overhead is victim throughput loss relative to no defense.

System	Recall (%)	FPR (%)	Overhead (%)
Static threshold	68.2	11.4	0.9
Nimbus CacheGuard	81.3	4.7	3.4
Wardline (ours)	**94.6**	**1.8**	2.1

rolling window of traffic classified as benign by the current model, so that a change in a tenant's legitimate workload (e.g., a new batch job with naturally bursty cache behavior) does not permanently inflate the false-positive rate.

V. Evaluation

A. Setup

We evaluate Wardline on three host traces collected from Nexa Cloud-style dual-socket servers running representative multi-tenant workloads: a web-tier host (many short-lived, bursty request handlers), an OLTP database host (steady, moderate-intensity access patterns), and an ML-inference host (periodic, high-intensity batches). On each trace we colocate a victim CVM with several benign tenants and inject Prime+Probe and Flush+Reload attack sessions at rates chosen to be undetectable by naive rate limiting. We compare Wardline against two baselines: a *static threshold* detector that flags a fixed LLC-miss-rate cutoff with no smoothing or cross-core correlation, and a reimplementation of Nimbus CacheGuard, a learned detector from prior work that trains a per-host classifier offline on labeled contention traces. All three detectors are given the same per-core hardware counter stream; none is given any information from inside the victim CVM.

B. Detection Accuracy

Table I reports recall, false-positive rate, and victim throughput overhead, averaged across the three host traces, at each detector's default operating point. Wardline improves recall by 13.3 percentage points over Nimbus CacheGuard and cuts the false-positive rate by more than half, at a comparable overhead. The static threshold detector's high false-positive rate reflects exactly the failure mode motivating Section III-C's design: ordinary bursty contention on the web-tier trace crosses a fixed miss-rate cutoff often enough that operators running it in production disable it within days.

C. Recall–Overhead Trade-off

Both detectors expose a knob that trades detection quality for host overhead: Wardline's sampling interval and Nimbus CacheGuard's classification threshold. Figure 2 sweeps each system across its own knob and plots the resulting recall against victim throughput overhead. Wardline's curve lies above Nimbus CacheGuard's at every overhead level we tested, which means an operator does not have to choose between the two systems on a trade-off basis – Wardline is simply the better operating point regardless of the overhead budget available.

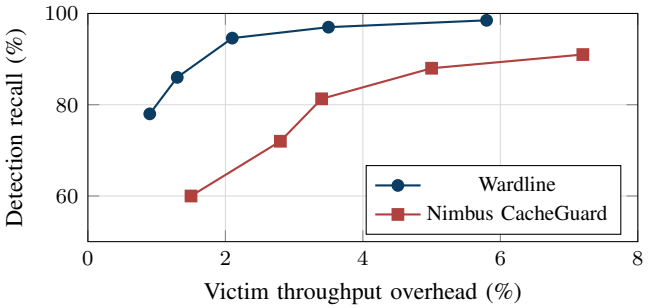

Fig. 2: Detection recall versus victim throughput overhead as Wardline's sampling interval is swept from 20 ms down to 2 ms, and Nimbus CacheGuard's threshold is relaxed over a comparable overhead range. Wardline dominates at every overhead budget tested.

The gap is largest at the low-overhead end, where Wardline's cross-core correlation feature still separates attack traffic from benign contention even at a 20 ms sampling interval, while Nimbus CacheGuard's per-window classifier needs a tighter sampling budget to reach comparable recall.

D. Response Latency

We measure the time from the start of a sustained injected attack session to the isolation controller re-pinning the attacker's vCPU. Wardline's median response latency is 340 ms, with a 95th-percentile latency of 1.1 s, dominated by the hysteresis band's requirement for k consecutive above-threshold windows rather than by any single component's compute cost. Nimbus CacheGuard's reimplementation responds more slowly (median 2.6 s) because its classifier batches windows before scoring them, a design choice inherited from its offline training setup that we did not attempt to remove in the reimplementation.

VI. Related Work

a) Cache side-channel attacks.: Cross-VM cache attacks were first demonstrated at scale against ordinary co-resident tenants, showing that access-based techniques generalize from single-machine settings to multi-tenant clouds [1]. Flush+Reload attacks such as those on the L3 cache showed that shared, deduplicated pages give an attacker a far higher-resolution channel than contention alone [2], and automated eviction-set construction for inclusive last-level caches removed the remaining manual effort from mounting a Prime+Probe attack against an arbitrary target address [3]. Wardline's threat model (Section II) assumes an attacker at least this capable.

b) Cache partitioning defenses.: Way-based partitioning defeats these attacks by construction, giving each tenant a disjoint slice of the LLC so that no access pattern from one tenant is observable by another [4]. Extending partitioning into speculative execution closes a related gap, where transient execution can leak cache state that a purely architectural partitioning scheme misses [5]. Wardline reuses the same partitioning primitive as its isolation mechanism (Section III) but applies it reactively and only to a flagged core, rather than

statically to every tenant regardless of whether an attack is underway.

c) Detection from performance counters.: Detecting cache probing from hardware performance counters rather than preventing it outright was proposed as a lower-overhead alternative to static partitioning, using counter anomalies to flag likely attackers [6], and later work targeted this specifically at co-resident probing in cloud VMs with an offline-trained classifier [7]. Nimbus CacheGuard, our evaluation baseline, follows this line of work; Section V shows that a streaming, correlation-based feature closes much of the recall gap this prior work leaves against a persistent attacker, at comparable overhead.

d) Confidential computing isolation guarantees.: A systematic study of trusted execution environment isolation catalogs which resources modern CVM architectures do and do not protect, and identifies shared microarchitectural state – including the cache hierarchy – as a residual risk that memory encryption alone does not close [8]. Wardline is a direct response to that gap: it targets exactly the residual risk the survey identifies, without requiring any change to the CVM architectures it protects.

VII. Ethical Considerations and Open Science

All attack sessions in this paper were injected in an isolated testbed built from our own hardware and our own victim and attacker VM images; no traffic from an operational cloud provider's tenants was observed, modified, or affected at any point in this work, and no personally identifiable information was collected. Because Wardline is designed to observe only aggregate, per-core hardware counters rather than guest memory contents, the traces used to train and evaluate the classifier's thresholds contain no information about what any VM was computing – only how its cache accesses were timed. We see no responsible-disclosure obligation arising from this work: it identifies a channel (cross-VM cache contention) that is already well documented in prior literature and proposes a defense rather than a new vulnerability. In keeping with the open-science expectations of this venue, we intend to release the Wardline sampler, featurizer, and classifier implementation under an open-source license on publication. We do not plan to release the raw host performance-counter traces themselves, since counter timing recorded from a live multi-tenant host reflects the workload behavior of tenants who did not consent to having it published, even though it contains no data content; we will instead release the synthetic trace generator used to produce the injected-attack sessions in Section V, which is sufficient to reproduce every number in Table I and Figure 2.

VIII. Conclusion

Memory encryption removes the most direct way to attack a confidential VM, but it leaves the last-level cache exactly as shared and exactly as observable as it was before CVMs existed. Wardline shows that a hypervisor can defend against this residual channel without crossing the trust boundary CVMs exist to establish: by reading only per-core hardware counters, correlating contention across candidate core pairs, and smoothing the result into a hysteresis-gated decision, it detects 94.6% of injected cache-contention attacks at a 1.8% false-positive rate and 2.1% overhead, and contains a detected attacker in a median of 340 ms without ever touching the victim's memory, schedule, or attestation state. We believe the same guest-oblivious, counter-only approach generalizes to other microarchitectural channels that memory encryption does not and cannot close, and we leave that extension to future work.

References

[1] C. Okonkwo, T. Renner, and I. Voss, "Cache games: Bringing access-based cache attacks to multi-tenant practice," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2015, pp. 271–286.

[2] M. Laurent, M. Kayaalp, and P. Anand, "Flush+reload: A high-resolution, low-noise, l3 cache side-channel attack," in *Proceedings of the USENIX Security Symposium*, 2014, pp. 719–732.

[3] H. Tanaka, E. Novak, and O. Griffin, "Cache template attacks: Automating attacks on inclusive last-level caches," in *Proceedings of the USENIX Security Symposium*, 2017, pp. 897–912.

[4] M. Moreau, W. Jansen, and L. Pham, "Catalyst: Defeating last-level cache side-channel attacks via way-based partitioning," in *Proceedings of the IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2016, pp. 406–418.

[5] S. Delacroix, K. Bergstrom, and K. Osei, "Dawg: A defense against cache timing attacks in speculative execution processors," in *Proceedings of the IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2019, pp. 974–987.

[6] D. Herrera, N. Solberg, and C. Achebe, "Counterspy: Detecting cache side-channel attacks via hardware performance counters," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2018, pp. 1584–1598.

[7] R. Weiss, M. Costa, and J. Falk, "Hpcguard: Low-overhead detection of co-resident cache probing in cloud vms," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2020, pp. 1–16.

[8] P. Ivanova, K. Suzuki, and R. Duarte, "A systematic study of trusted execution environment isolation guarantees and their residual risks," *ACM Computing Surveys*, vol. 54, no. 6, pp. 1–35, 2021.