GateBleed: Exploiting On-Core Accelerator Power Gating for High Performance and Stealthy Attacks on AI

Joshua Kalyanapu
North Carolina State University
Raleigh, USA
jkalyan@ncsu.edu

Farshad Dizani*
North Carolina State University
Raleigh, USA
fdizani@ncsu.edu

Darsh Asher*
North Carolina State University
Raleigh, USA
dkasher@ncsu.edu

Azam Ghanbari
North Carolina State University
Raleigh, USA
aghanba2@ncsu.edu

Rosario Cammarota
Intel
San Diego, USA
rosario.cammarota@intel.com

Aydin Aysu
North Carolina State University
Raleigh, USA
aaysu@ncsu.edu

Samira Mirbagher Ajorpaz
North Carolina State University
Raleigh, USA
smirbag@ncsu.edu

Abstract

As power consumption from AI training and inference continues to increase, AI accelerators are being integrated directly into the CPU. Intel's Advanced Matrix Extensions (AMX) is one such example, debuting in the 4th Generation Intel Xeon Scalable CPU, attaining significant gains in the metrics of performance/watt and decreased memory offloading penalty. This paper discloses a timing side and covert channel, GateBleed, caused by the aggressive power gating utilized to keep the CPU within operating limits.

This paper shows that the GateBleed side channel is a threat to AI privacy, as many ML models such as Transformers and CNNs make critical computationally-heavy decisions based on private values like confidence thresholds and routing logits. Timing delays from the selective powering down of AMX components mean that each matrix multiplication is a potential leakage point when executed on the AMX accelerator. This paper identifies over a dozen potential gadgets across popular ML libraries (Hugging Face, PyTorch, TensorFlow, etc.), revealing that they can leak sensitive and private information, including class labels and internal states. GateBleed poses a risk for local and remote timing inference, even under previous protective measures.

GateBleed gadgets can also be used as a a generic high performance, stealthy magnifier for microarchitectural attacks to bypass timer resolution coarsening defenses and create robust and realistic side channels in noisy environments, such as remote attacks on networks with high traffic. This paper shows that when GateBleed gadget is used as a transmission channel for Spectre, it can leak arbitrary memory addresses of the victim with high performance (0.067 bps), and evade the state-of-the-art microarchitectural attack detectors for the first time.

This paper implements an end-to-end membership inference attack with 81% accuracy on a Transformer model optimized with Intel AMX and 99% accuracy on an early-exit CNN classifier. GateBleed achieves 0.89 precision while leaking expert choice in a Transformer mixture-of-experts (MoE) with 100% accuracy. These attacks do not rely on confidence scores or model outputs, but only on the execution time of attacker-controlled AMX instructions on the shared hardware accelerator with power gating. To the authors' knowledge, this is the first side-channel attack on AI privacy that exploits hardware accelerator power optimizations. The paper also suggests effective mitigations and measures their trade-off between *power consumption* and *performance*.

CCS Concepts

• **Security and privacy → Hardware attacks and countermeasures**; **Hardware reverse engineering**; • **Computer systems organization → Special purpose systems**; • **Hardware → Platform power issues**; • **Computing methodologies → Neural networks**.

ACM Reference Format:
Joshua Kalyanapu, Farshad Dizani, Darsh Asher, Azam Ghanbari, Rosario Cammarota, Aydin Aysu, and Samira Mirbagher Ajorpaz. 2025. GateBleed: Exploiting On-Core Accelerator Power Gating for High Performance and Stealthy Attacks on AI. In *58th IEEE/ACM International Symposium on Microarchitecture (MICRO '25), October 18–22, 2025, Seoul, Republic of Korea.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3725843.3756097

1 Introduction

AI applications are power intensive [2, 38]. Advanced power optimizations such as power gating are being deployed in AI accelerators to reduce the power consumption overhead and ease this burden [13, 66, 75, 76, 89, 134], however, their privacy and security risks on AI applications and remote settings are poorly understood.

*Both authors contributed equally

MICRO '25, Seoul, Republic of Korea

ACM ISBN 979-8-4007-1573-0/25/10
https://doi.org/10.1145/3725843.3756097

As we enter a new era of computing, each leap in processor technology not only increases performance and efficiency, but alters the landscape of cybersecurity threats [131]. Microarchitectural optimizations can introduce side channels that can expose sensitive data to adversaries. Spectre [71] and Meltdown [83] show how speculative and out-of-order execution can be exploited as side channels exposing private data through subtle timing differences. Subsequent research uncovered an avalanche of vulnerabilities exploiting various microarchitectural components, such as the cache [46, 101, 146], internal CPU buffers [17, 18, 52, 69, 71, 73, 83, 90, 91, 94, 95, 103, 114, 125–127, 138, 142], prefetchers [23, 45, 81, 130], pointer authentication [106], TLB [44, 74, 118], execution ports [12, 15], scheduler queues [39], micro-op cache [107], RAPL [82], and DVFS [84, 135, 136], showing that hardware-based timing side channels undermine the very foundations of higher-level isolation and trust—rendering traditional defenses such as encryption, sandboxing, memory isolation, and privilege separation insufficient in the face of hardware-level information leakage.

MLaaS interfaces and multi-tenant cloud services are increasing in deployment, but little attention has been paid to the impact of optimizing hardware accelerators on the security and privacy of AI applications, as well as user privacy. Previous works focus on attacks that extract architectures and hyperparameters [22, 61, 100, 122, 133, 144], alongside inference attacks that use output confidences or logits and training a proxy model that behaves similarly to the under-attack model using the under-attack model's output to deduce model properties such as training-set membership (MIAs) [24, 32, 48, 50, 51, 53, 80, 86, 87, 112, 116, 148], attributes of user inputs [43, 63, 92, 150], and decisions in adaptive models such as early exits and MoE routing [16, 28, 54, 79, 121]. Restricting model outputs to labels rather than confidence scores [116] is largely effective against many membership inference attacks. Differentially private training [8, 62] prevents a single training sample from significantly affecting the output. Finally, execution padding with dummy instructions in adaptive DNNs hides dynamic routing and early-exit timing[11, 16, 79]. However, no work has studied whether these leaks can originate below the API —in hardware optimization itself, where higher-level isolation, such as output masking and DP training, is utterly insufficient to secure the data.

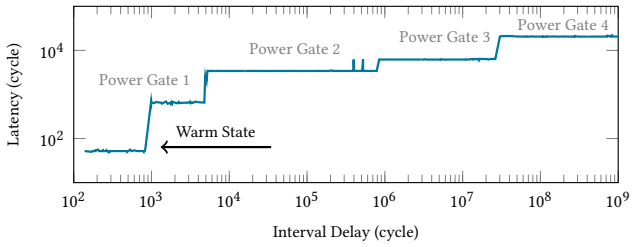


Figure 1: Performance States of TMUL due to Power Gating Leaks Private AI Parameters or Arbitrary Data and the Secure Recommended Intel AMX Operational State (Warm)

Conventional timing side-channels that exploit hardware optimizations to target AI privacy leak data stored in memory or in registers in the form of bits, not properties of the data. For example, Cache Telepathy [144] uses cache-based timing to recover model architectures and hyperparameters present in memory. Hertzbleed [135] or Platypus [82] use operand-dependent CPU frequency throttling to leak register contents. On the other hand, *AI privacy attacks* obtain information that is *not* stored as bytes, e.g., training-set membership or expert routing choice. The leaked information exists only as latent properties of the learned model, and in some cases, the execution leaves observable footprints on hardware. This paper investigates the extraction of properties of the data used to train the AI model via timing discrepancies due to the *hardware optimization* of on-core accelerator power gating.

With the sharp increase in power consumption required by modern AI [2, 38], hardware designers have been looking to further increase the energy efficiency of AI inference and training. Intel recently introduced Advanced Matrix Extensions (AMX), an on-core AI accelerator available in Scalable Xeon CPUs [96, 98, 128]. AMX achieves up to 10× inference and training performance and 3× performance per watt improvement over the previous generation. AMX delivers high throughput for the multiplication of int8 and bfloat16 matrices, achieving more than 30 TFLOPS by executing up to 1024 MACs per cycle, culminating in up to a 75% lower Total Cost of Ownership (TCO) than the previous generation of Scalable Xeon CPUs [1]. However, no prior works have explored the security and privacy implications of Intel AMX.

We performed a complete characterization of AMX performance and power and found novel timing channels in the form of a *reuse-distance-dependent* timing difference of up to 20,000 cycles with a single AMX instruction (see Figure 1). This difference is even visible when AMX is run inside an Intel SGX secure enclave [113], allowing observation of AMX utilization even within a trusted execution environment. We found the root cause of the high timing discrepancy to be an undisclosed power gating optimization, a ubiquitous power-saving measure that entails the powering off of unused chip components to save power [13, 75, 89]. While power-gating does reduce power consumption, power-gating also incurs an extra measurable latency when waking up a powered-off component, hence enabling a novel reuse-distance-dependent timing difference. We thus introduce a side-channel attack, GateBleed, that *bleeds* information via the wakeup latency induced by power *gating*.

By monitoring accelerator timing, we can determine training-set membership with high accuracy. The ability to determine whether specific data were used to train a deployed model has implications for user privacy, security, and corporate liability. Leaking membership in an ML model reveals whether a specific input was part of the training set, directly threatening data privacy by violating confidentiality guarantees of MLaaS systems. Attackers who know a model's training corpus can better craft adversarial or targeted attacks [9, 20, 102]. Recent lawsuits involving Stability AI [3], Google [35, 57], and Meta [129] highlight that companies may face legal exposure if one can demonstrate that proprietary or unauthorized data were used during training [3, 6, 35, 57, 64, 108, 129].

Membership inference attacks (MIAs) exploit the tendency of models to be more confident in training examples than in unseen examples. In the original black-box MIA [116], the adversary trains shadow models that mimic the target and also trains a

meta-classifier using the shadow model outputs (labels and confidences) for known members and non-members; the meta-classifier is then applied to the target model's outputs. Subsequent work extended MIAs beyond classifiers and to more restrictive APIs, including attacks on non-classifier models [32, 48, 49, 51, 86, 87], models that use only top-K confidences [112], or even label-only outputs [24, 50, 80, 104, 148]. In practice, the strongest MIAs generally rely on access to logits, which are often hidden in production systems. Such leakages threaten data privacy and may raise compliance concerns [97]. However, many MIAs degrade over noisy channels (e.g., real-world networks) and become less effective when only hard labels are exposed. To our knowledge, GateBleed is the first timing side-channel attack utilizing a hardware accelerator power optimization to target user data privacy without relying on the confidence scores or even an output label.

Leveraging the insight that ML models tend to exhibit higher confidence on training-set members than non-members [11], we utilize GateBleed to detect AMX usage of an adaptive neural network running on the same core as an attacker-controlled program; the AMX usage patterns allow the attacker to infer whether or not the adaptive neural network was trained on a particular input. Inputs that were in the training set are more likely to produce high-confidence predictions [116], triggering different behavior in adaptive neural networks [11]. Adaptive neural networks have been deployed in products by Google [78], Meta [67], Amazon [123], and Alibaba [88]. We show that by detecting early exits or smaller expert execution, the attacker infers membership without accessing logits, output probabilities, or model internals, compromising user data privacy through power gating.

GateBleed can also infer private decisions such as expert routing in mixture-of-experts transformers by correlating accelerator timing with model behavior. In particular, *Mixtral (HF)* [56], *TensorFlow MoE* [119], *DeepSpeed MoE* [26], *ONNX RunTime MoE* [111], and *Mixtral (llama.cpp)* [42] exhibit AMX-backed conditional expert execution that is exploited in our GateBleed attack. For example, recent works have proposed heterogeneous experts [132] or a different number of experts activated for each token in the Transformer [55], leading to an observable AMX timing state. GateBleed harnesses this difference in computation to leak the *expert routing index* through the power gating footprints. This work identifies more than a dozen exploitable timing gadgets across popular machine learning stacks (including PyTorch, TensorFlow and HuggingFace implementations), showing the footprint of accelerator power gating in modern AI pipelines and agents. GateBleed enables a broader family of accelerator-driven privacy leaks. Any residency- or power-managed compute path—on-core AI engines (e.g., AMX/NPUs), GPU tensor cores, or dedicated inference IP—can imprint input- or model-dependent timing. The resulting signals need not expose stored bytes; they can encode decisions about membership, routing, tool selection in Agentic AI, or early exits across processes, VMs, and TEEs, and can be magnified for remote settings.

Because GateBleed exploits hardware design, the vulnerability cannot be fixed with a simple software update. Effective mitigation likely requires hardware changes that take years to reach deployed systems and have high power overhead. Because GateBleed doesn't rely on model outputs such as confidence scores, GateBleed evades existing defenses built for inference attacks. For example, proposed software-level defenses such as confidence score masking [116] and differential privacy [34] do not work to mitigate GateBleed.

Furthermore, this paper explores using GateBleed as the covert channel for a remote Spectre-v1 [71] and as a microarchitectural magnifier to measure a microarchitectural state in an environment in which only a coarse timer is available. The remote Spectre-v1 attack achieves never-before-seen stealthiness and leakage rate on a noisy, production network while bypassing state-of-the-art microarchitectural attack detectors like Evax [10], PerSpectron [93], and RHMD [68]. Utilizing GateBleed as a magnifier allowed distinguishing between an L1 cache hit or an L3 cache miss using an exceptionally coarse timer. The ability to use GateBleed as a general-purpose covert channel and as a microarchitectural magnifier highlights additional threats posed by power-gating optimizations.

We revisit the state-of-the-art microarchitectural attack detectors and show that they achieve near-100% detection of recent magnifiers [117, 143] with high accuracy due to the magnifiers' visible execution patterns, reliance on hardware optimizations (e.g., branch predictor, cache, prefetcher), and high repetition rates. However, microarchitectural attack detectors fail to detect GateBleed attack variants in time, making GateBleed the only tested microarchitectural side channel attack that evades microarchitectural attack detectors, because detectors (i) do not account for the misuse of hardware *power* optimizations such as power gating exploited in this attack, (ii) cannot resist the low repetition rates required for GateBleed to succeed, (iii) do not capture the passive reset phase of GateBleed, which avoids anomalous microarchitectural instructions, and (iv) miss accelerator-specific features and counters from their design.

We propose mitigating GateBleed with a microcode update that maintains AMX in a single stage or using compiler-inserted AMX dummy operations to maintain the AMX power stage. This work shows that these mitigations increase power consumption overhead by up to 12%, but secures against GateBleed.

Contributions: This paper makes the following contributions:

- We uncover and characterize the undocumented power gating in AMX, revealing five distinct latency states that vary based on time of last usage.
- We present GateBleed, the first microarchitectural *privacy* side channel that exploits on-core accelerator power gating to infer training-set membership, MoE routing, and early exits—without logits/confidences or access to stored artifacts. We exploit existing benign ML libraries, leaking sensitive private ML secrets and inference paths across OS and VM. Our end-to-end attack on MoE achieves 100% success rate with 0% FPR. Similarly, the early-exit CNN attack achieves 99.72% accuracy with just 0.54% false positives. Our membership inference attack on early-exit Transformers, leveraging the same timing mechanism of GateBleed, yields 81% accuracy (78% TPR, 84% TNR).
- We present a variant of GateBleed as a generic stealthy magnifier in intensifying a 200-cycle timing difference to a 10,000-cycle difference. We evaluated GateBleed as a covert

channel that achieves a 70,000× higher leakage rate than NetSpectre on our production network, where the prior covert channels fail. We show that GateBleed evades malware detectors trained on microarchitectural attack patterns due to minimal instruction footprint and absence of cache or TLB flushing. GateBleed also defeats timer coarsening defenses by operating with timing margins of up to 20,000 cycles.
- We discuss mitigations to close this attack vector and measure the mitigations' power and performance overhead, concluding that the most practical solution, powering off AMX on a context switch, results in a power consumption overhead between 2% and 12% depending on the context switch rate.

Responsible Disclosure: We disclosed this issue to Intel from May 2023 to May 2024. Intel confirmed our findings, and Lenovo released a firmware mitigation. In June 2024, Lenovo released a UEFI update, Version 3.20, Build ID ESE126H [Critical], which mitigates the most critical GateBleed variants, particularly those exploiting AMX power gates 3, 4, or 5. We have open-sourced the GateBleed proofs-of-concept (PoCs)[1].

Paper Organization: Section 2 provides background on timing channels, timing channel defenses, adaptive neural networks, and AI model vulnerabilities. Section 3 presents the threat model and attack requirements. Section 4 presents a detailed overview of Intel AMX architecture and the GateBleed vulnerability. Section 5 introduces the GateBleed attack, demonstrating its use in side channels, covert channels, and magnification gadgets across real ML workloads. Section 6 provides the practical results. Section 7 proposes effective mitigations and evaluates their overhead. Finally, Section 8 concludes the paper.

2 Background

Timing Microarchitectural Attacks. Covert channels exploit shared microarchitectural states to transmit information using timing differences. Classical cache-based attacks, such as *Prime+Probe* [101], *Flush+Flush* [46], and *Flush+Reload* [146], rely on eviction or access timing to leak fine-grained memory behavior. TLB-based attacks such as *TLBleed* [44] and *Binoculars* [151] extend cache-based covert channels to memory translation structures, using page-level state or page-walker contention. Branch predictor attacks [36] infer secret control flow through the predictor state, typically requiring simultaneous multithreading (SMT). Transient attacks such as Spectre [71] and Meltdown [83], and MDS-type attacks [71, 83, 114, 127] exploit microarchitectural optimizations such as out-of-order execution or speculation to access unauthorized data transiently; speculative execution attacks rely on benign code that has unintended effects on the hardware. Values leaked in a speculative execution attack are exfiltrated via a covert channel. Hardware mitigations like speculative barriers and buffer flushing limit, but do not eliminate such attacks. Remote variants like Netspectre [115] demonstrated feasibility even without attacker code on the victim machine, albeit at extremely low bandwidth, which fails on a real production network because of the network noise masking timing differences.

Power and Frequency Optimization Based Attacks. *Platypus* [82] exploits the RAPL interface to leak secrets from SGX enclaves via data-dependent power consumption. Hertzbleed [135] shows how data-dependent power consumption can leak information via timing due to CPU frequency throttling. Wang et al. [136] extend frequency throttling via data-dependent power consumption to on-board graphics. Rauchscher et al. [105] exploit CPU sleep states, or C-states, to introduce timing differences. Thor [29] exploits an operand-dependent timing difference observed in Intel AMX's lowest power state to leak operand sparsity in matrix multiplications. These channels are powerful but vulnerable to noise and can be suppressed through frequency locking, access restrictions, added noise to power measurements [82], and reduced measurement rate.

Microarchitectural Magnifiers. Microarchitectural magnifiers are techniques to magnify a microarchitectural effect that is hard to measure with conventional means. Hacky Racers [143] constructs ILP-based magnification gadgets to induce a potentially unbounded timing difference, defeating timer coarsening defenses such as the standardized 5 μs timer resolution [140] implemented in Chrome, Firefox, Edge, and Safari and enabling microarchitectural side-channel attacks in the browser which exploit differences on the order of nanoseconds. Bypassing the same timer coarsening, Spring [141] encodes a secret into multiple cache lines instead of a single one. Microscope [117] is a modification to the OS page fault handler attacking trusted execution environments (TEEs) like Intel SGX. Attacks on TEEs assume a privileged attacker. By modifying the page fault handler to constantly retry, small sections of victim code repeat execution, making side-channels that rely on extremely subtle effects like execution port contention [12] easily visible with only one architectural run of the victim code. However, existing magnifiers require attacker-controlled code or complex code that is unlikely to exist in a benign codebase; furthermore, our experiments have shown that the existing magnifiers are highly detectable by the state-of-the-art microarchitectural attack detectors [10, 93].

Attacks on AI Security and Privacy. *Data security attacks* target the model artifacts themselves (e.g., parameters, hyperparameters, or proprietary architectures) resident on the victim machine (often in process memory). Adversaries exploit physical and microarchitectural side channels such as cache access patterns [144], timing variations [33], electromagnetic emanations[14], and power consumption [30, 139] to recover sensitive model information or keys, or to guide model extraction. Other avenues include training high-fidelity surrogates to replicate a target model's decision boundary [61] and cryptanalytic/model-extraction techniques that recover parameters or architectures from query accesses or compressed representations[19, 21].

User privacy attacks, by contrast, exploit input–output behavior to infer sensitive information about the data used to train or query the model—items not directly present in process memory. This includes membership inference[24, 48, 50, 51, 80, 86, 87, 116] and input data attribute inference [43, 92, 150]. Adaptive or early-exit DNNs introduce additional leakage channels because inputs may traverse different subnetworks [31, 54, 121]; routing choices and early exits can be inferred remotely via end-to-end latency measurements

[1]Code available at https://zenodo.org/records/17019733. All results in this paper are reproducible within 40 minutes following the instructions at https://github.com/jkalya/gatebleed, and have been verified through the MICRO artifact evaluation.

Library / Model	Leaked Parameter/ Variable	Leakage Path and Threat Model (Query / Timing)
I. Input-Dependent Routing Gadgets		
Mixtral (HF) [56]	Expert routing index/Routing logits	AMX matmuls execute only for selected experts; Query+Time reveals routing decisions.
AdaptiveLogSoftmax [25]	Cluster membership/Target label	Cluster-based branches vary in compute time; Query+Time reveals true class.
TensorFlow MoE [119]	Router activation/Routing threshold	AMX triggered only for active experts; timing reveals routing threshold decisions.
DeepSpeed MoE [26]	Sparse expert maskMoE gating pattern	Expert selection alters AMX load; Query+Time timing reveals active paths.
ONNX Runtime MoE [111]	Active expert path/Expert selection	Expert-specific AMX ops in If-nodes; timing-only observer can infer selected branch.
Mixtral (llama.cpp) [42]	Gate threshold/Router logits	First expert triggering AMX leaks routing; fine-grained timing from query reveals selection.
RGATConv (PyG) [27]	Edge-type dependency/Edge attribute	Conditional projection varies by edge type; timing-only observer reveals structural edges.
LangChain [77]	Tool dispatch category/Classification logits	Tool path alters execution time; Query+Time reveals decision path.
ggml [4]	Batch size	Executes AMX only if batch size is not 1. AMX usage leaks if batch size is 1
II. Confidence and Early-Exit Gadgets		
BranchyNet [121]	Exit stage decision/Confidence score	Exit stage changes AMX pattern; Query+Time/AMX usage reveals model confidence.
MSDNet [54]	Early-exit threshold/Prediction entropy	Prediction entropy modulates early-exit logic. Query+Time reveals entropy.
SkipNet / BlockDrop [137]	Layer skipping pattern/Routing mask	Conditional skips affect AMX reuse. Latency pattern encodes layer execution mask.
HF Agent [56]	Action type/Action logits	Decoder used only for tool tokens; timing from query reveals action category.
AutoGen [120]	Loop interval/Planner state	Internal planner invokes AMX conditionally. Timing leaks planning state.
III. Session, Configuration, and Static Context Gadgets		
LLaMA KV Cache [56]	KV reuse vs. recompute/Context reuse	Reused prompt avoids recomputation; query+timing reveals reuse and leaks prompt history.
ONNX Runtime KV Cache [110]	Session persistence/Session reuse	Warm sessions reduce AMX setup time; timing-only observer detects session reuse.
llama.cpp Quant Dispatch [42]	Model format/Quantization type	Model quantization (int8 vs fp32) toggles between AMX/AVX. Timing reveals format.
GoogLeNet [25]	Training mode/Training flag	Auxiliary classifier invoked only in training. Latency reveals execution mode (train vs infer).
Generic CNN [5]	Layer type / Architecture toggle	Architecture flagged (e.g., conv vs MLP) alter AMX invocation. Latency exposes layer type.
OpenAI Function API [99]	Completion state/Finish signal	Completion path triggered AMX only for function tool. Latency leaks endpoint behavior.

Table 1: Categorized GateBleed gadgets and threat models. Each row describes a parameter that influences AMX usage and may leak via timing or query-time side channels. The final column summarizes the leakage path and attacker model.

even when model parameters are fixed [16, 79]. Timing signals can also leak membership in adaptive networks: overconfident routing can make training members exit earlier than non-members, yielding faster inference on members [11]. These timing-based attacks typically require precise synchronization and stable network conditions; in contrast, GateBleed enables an attacker to infer private information by measuring only its own program's latency on shared hardware, removing the need for end-to-end timing of the victim model.

3 Threat Model

Here we enumerate the threat models we assume for the GateBleed attack —the attack targets and attacker capabilities. Table 1 shows GateBleed gadgets present in real-world codebases along with the threat model we assume for leaking with that gadget.

The GateBleed *attack target* is any internal model parameter or decision whose unintended disclosure through timing variations can compromise user privacy, model confidentiality, or operational integrity. Specifically, these secret parameters include: (1) intermediate confidence scores or prediction entropy values; this leakage enables precise membership inference attacks by distinguishing training-set members with typically lower intermediate entropy from non-members [116]; (2) internal routing decisions such as expert selection in Mixture-of-Experts (MoE) models, where the activated expert is determined by private input-dependent logits; and (3) operational or contextual flags, such as session reuse indicators (key-value cache usage) or quantization configurations, revealing sensitive model runtime states or deployment settings.

Attacker capabilities vary depending on the target gadget. The "Threat Model" column in Table 1 highlights the range of attacker capabilities required to exploit each gadget. Namely, we investigate three threat models - (1) *Query+Time* where the attacker remotely queries a MLaaS model and times the response time to leak details about the model, (2) *Timing* where the attacker remotely sniffs packets on the network to determine the response time of a user's request to the MLaaS model to leak details about the user's input, and (3) *AMX Usage* where the local attacker colocated on the same core as the MLaaS program can both induce the model to run and time its own AMX operations to leak.

The most constrained *remote* attackers (Query+Time, Timing threat models) can leak several targets. For example, an adversary over the network can send carefully chosen queries to an MLaaS API and measure response times to infer facts about the model. This minimal attacker model only needs to observe overall request latency, and GateBleed's signal remains detectable despite network noise because of the significant timing difference. The attacker does not know model weights, architectural parameters, or outputs and is unprivileged, but using response latencies can build a correlation of inputs to response times.

Gadgets become even more pronounced if the attacker is on the same host as the MLaaS model due to the ability to obtain a more fine-grained view of AMX usage than what end-to-end timing reveals (AMX Usage threat model). Assuming the attacker can start its own program on the same core as the model, the attacker can time its own AMX operations in parallel with the model inference to see when AMX was used by the model. Since the OS will interleave execution of both workloads, a slow AMX operation means AMX was not used by the model recently, while a fast AMX operation implies otherwise. At the cost of the more relaxed threat model, all gadgets can be utilized with the bonus of

fine-grained observations, implying fewer iterations required for secret leakage. We emphasize that this threat model does not require end-to-end timing of the MLaaS model, nor does it require access to the model logits or confidence scores, as in most already-discovered membership inference attacks.

As shown in the column of the Leakage Path and Threat Model in Table 1, some gadgets are exploitable remotely, while others need a local perspective, highlighting that the attack surface of GateBleed ranges from remote services to trusted environments. Even Intel SGX enclaves running vulnerable ML code show AMX cold start timing visible to a monitoring attacker in the host OS. A privileged adversary can infer secret-dependent decisions in the enclave by timing operations. This means that OS, VM, and SGX isolation does not stop GateBleed - the timing signal is visible to any observer capable of timing operations.

The side-channel attack described in Section 6.4 operates off the assumption that a Spectre-style gadget is available in pre-existing, legitimate, non-malicious network-exposed code; we assume no attacker-controlled trojan/spy code running on the victim, modified code running on the victim, or victim collusion; this is the same threat model as Netspectre [115]. GateBleed turns the secret-dependent speculative AMX usage into a high-resolution timing channel via the Query+Time and Timing threat models.

4 Reverse Engineering

In this section, we describe our process for reverse engineering the AMX hardware beginning with a description of the documented AMX features, continuing with our discovery of a reuse-distance-dependent AMX latency, and concluding with a root cause analysis.

4.1 Intel AMX Architecture

AMX is an on-core AI accelerator in the Intel Xeon 4th (Sapphire Rapids) [98], 5th (Emerald Rapids) [96], and 6th (Granite Rapids) [128] generation scalable CPUs featuring high-throughput tile-based matrix operations such as `TDPBSSD` and `TDPBF16PS` [59, 60]. Unlike off-core accelerators, AMX instructions execute within the core pipeline.

Crucially, AMX introduces eight new 1 KB architectural *tile registers* capable of holding 16×64 `int8` or 16×32 `bfloat16` matrices [59], resulting in a total tile storage overhead of 8 KB, one-fourth of a 32 KB L1 cache. Due to this, AMX achieves up to 1024 int or 512 bfloat16 FMA operations per cycle, outperforming AVX-512 [58].

Before execution, AMX tiles must be configured using the `LDTILECFG` instruction to specify the number of rows (up to 16), bytes per row (number of columns), and starting row [65]. Data is transferred into tiles using `TILELOADD/TILELOADDT1`. The matrix multiplication instructions are `TDPBF16PS` for `bfloat16` input and `TDPBSSD`, `TDPBSUD`, `TDPBUSD`, and `TDPBUUD` for signed and unsigned 8-bit integers [60]. After computation, tiles are stored back into memory via `TILESTORED`. `TILEZERO` zeroes tile registers, and `TILERELEASE` deactivates AMX, minimizing context switch overhead by avoiding the need to save 8 KB of tile register data.

4.2 Power-Gating, Latency & Privacy Leakage

In this experiment, we measured how long it takes to execute a single AMX multiplication instruction while varying the intervals between consecutive executions. Figure 1 shows that the latency of a `TDPBSSD` (signed-signed 8-bit integer matrix multiplication) differs depending on the time since the AMX unit was last used; that is, *reuse distance*. By changing the length of these intervals, we observed five distinct execution times for the AMX multiplication instruction.

We noticed that this latency goes through five distinct *performance stages* which correspond to a 50, 600, 6000, 9000, and 20,000 CPU cycle latency to perform a `TDPBSSD`. We classified these into performance states, with the shortest execution time labeled as the *Warm State* and the longer execution times labeled as *Cold States* - specifically Cold State 1 (the second shortest), Cold State 2, Cold State 3, and Cold State 4 (the longest). This is the foundation of our GateBleed attack, enabling both covert and side-channel attacks. In the next section, we reverse engineer the root cause of this behavior and show that these stages constitute a class of timing leakage rooted entirely in on-core AMX accelerator power management.

GateBleed exploit the timing difference created by Intel AMX power gating cross process and remote with the knowledge of the library (for example having the routing logic in MoE or early exit in MSDNet shown in Table 1), to leak secrets such as the membership of the input or private parameters of a deployed model. Figure 3 shows that the TMUL latency is over 4000 CPU cycles different when executed after a member inference in CNN or a non-member inference, enabling a cross-process GateBleed attack on CNNs. We show that the leakage remains resilient even under moderate scheduling interference, with timing margins exceeding 4000 cycles between member and non-member AMX invocation illustrated in Figure 3. This phenomenon happens on the individual AMX TMUL operations due to power gating (we discuss the root cause in the next section), but we can see the effect cascading over the entire block of AMX instructions, even in an end-to-end large-scale Transformer. For example Figure 2 compares (a) non-member (b) member resulting in a 500,000 CPU cycle timing difference in an end-to-end inference of a transformer model, depending on the membership status of the input, enabling a remote MIA attack on the Transformer Model.

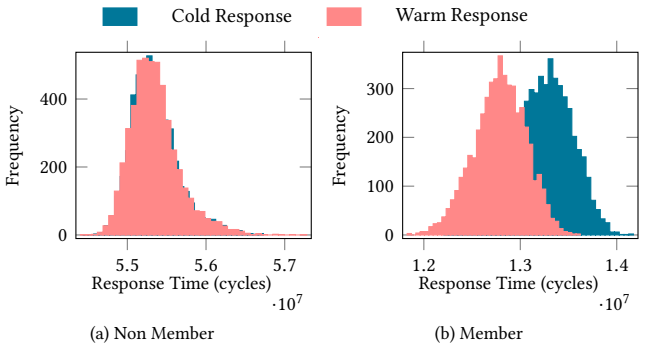


Figure 2: The response time distributions of an end-to-end *transformer* model inference.

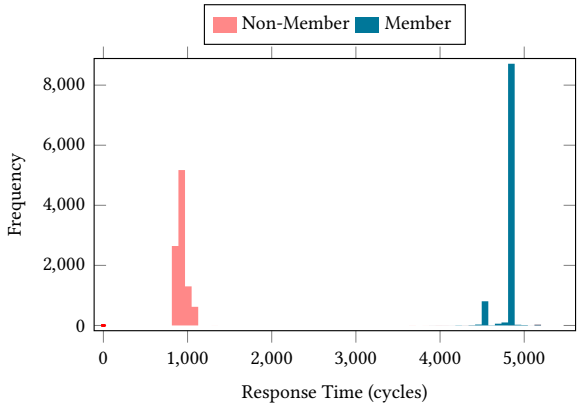


Figure 3: AMX Operation Latency Cross-Process after CNN Inference for Member vs. Non Member.

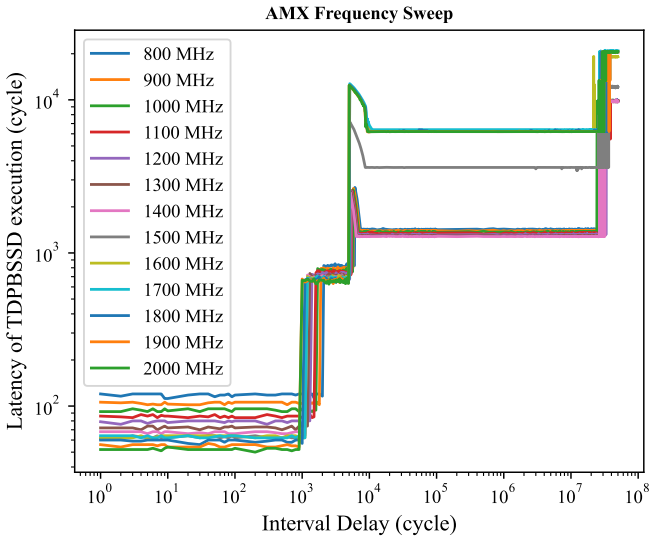


Figure 4: GateBleed leakage under various fixed core frequencies.

4.3 Root Cause Analysis

To understand what novel capabilities GateBleed provide to the attacker and what privileges are required for this exploit, as well as how to mitigate it, we systematically investigated the root cause of Intel AMX performance stages and confirm that the root cause is power-state transitions driven by AMX's independent power management, rather than core frequency scaling, operand dependencies, SMT, value dependency, SGX, or any of the core power savings settings; that is, C-states and C1E.

The following is a summary of the main findings.

Frequency Scaling and Throttling Effect. Frequency scaling and the Intel Turbo Boost feature are the root cause of many software covert channels, such as [72, 82, 84, 135] or even physical side channels [14, 139]. However, disabling Turbo Boost did not have an impact on AMX performance stages, which this attack exploits, confirming that CPU frequency scaling (DVFS) is not responsible for observed timing variations. Fixing the CPU frequency showed that while timing differences changed with frequency, even at 800 MHz, a 9,000-cycle gap remained exploitable. Figure 4 shows a frequency and wait-time delay sweep, observing `TDPBSSD` timings, which showcases the same five-stage performance trend across all core frequencies.

Thus, the attacker does not require the privilege of Turbo Boost or the ability to set a fixed frequency on the victim's CPU for the attack to work. In addition, disabling Turbo Boost or frequency locking does not mitigate GateBleed, unlike Hertzbleed [84, 135]. This significantly expands attackers' capability over frequency throttling-based covert channels[84, 135].

Core C-states. C-states are CPU power-saving modes. C-states are numbered as follows: C0 – The core is fully active and executes instructions. C1 – The core is idle, but can return to C0 quickly. C2, C3, …, Cn – Deeper sleep states; progressively more parts of the core are turned off. Each deeper state saves more power, but takes longer to wake up. Although great for efficiency, they can unintentionally leak information through timing side channels when switching between power states - a root cause exploited by IdleLeak [105]. We enabled and disabled the C-state feature in the system BIOS and observed that the performance stages remain unchanged; see table 2. CPU C-states are not the root cause.

C1E, Busy Waiting & usleep. We check if CPU C states can affect AMX execution times via the `sleep` function, a function that explicitly puts the CPU in C1 for short waits and C6 for long waits. Using `sleep` instead of busy waiting (to activate C states C1 and C6), if we disable enhanced C1 (C1E) while C states are enabled, cold stage 4 remains visible at a 20,000-cycle latency. C1E is an "Enhanced Idle" CPU state that combines clock gating and voltage reduction. It is deeper than C1, but still has low latency. With C states and C1E both enabled, we observe that cold stage 4 induces a 9,000-cycle latency while cold stage 2 induces a 3,000-cycle latency, indicating that while conventional C states do not affect AMX power gating, C1E does: C1E prevents AMX from entering deepest sleep, but it does not remove all exploitable stages. Thus, C1E is not a cause itself and therefore, disabling C1E or C-state does not mitigate GateBleed.

Value Dependency. We varied the operand value and noticed that this does not affect the five performance stages, ruling out value dependence as the root cause of five AMX performance modes.

Power Limit. Some covert channels are limited to only the lower power limits of the CPU [82, 84, 135]. Thus, we varied the power limit from the full range of possible power limits (126.0-454.0 watts) in the PKG domain to see if GateBleed is limited to a certain power limit. All stages were present in all power limits. Thus, power limits are not the root cause of the observed behavior in AMX stages, and changing them would not mitigate GateBleed, unlike [82, 84, 135].

Prefetching Effect. We hypothesized that the latency stage behavior is due to differing cache hits/misses incurred when loading a tile register with the `TILELOADD` command. However, we observe this latency stage behavior for AMX instructions that do not touch

the cache, such as TDPBSSD. Therefore, hardware prefetching is not the cause.

Kernel handling. We tested both RHEL 9.4 and Ubuntu 22.04 OS. All five performance stages in AMX exist in both versions.

Multi threading. Our reverse engineering shows that AMX is not shared among threads. GateBleed is not exploiting contention among threads, unlike other covert channels [44, 151] and thus cannot be mitigated by turning off multithreading/SMT.

Power Consumption. Finally, to check power consumption in the different states, we implement a workload that performs a TDPBSSD and then waits for the minimal amount of time to keep AMX in a particular stage, and another identical workload in which the TDPBSSD instruction is omitted. We run the workloads on every core and gather the average power consumption over a period of 10 seconds. By comparing the power consumption of the first workload with the power consumption of the second workload, we can isolate the contribution of AMX in the power stages. Figure 5 shows the wattages we obtained along with the percent increase.

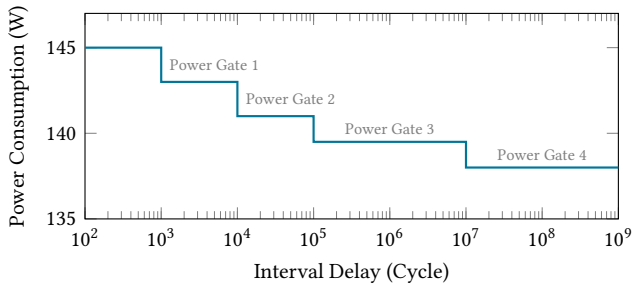


Figure 5: AMX power consumption clearly showing sharp, stepwise power gating transitions at defined interval delays.

The gradual decrease from Stage 0 (142.08W) to Stage 4 (138.49W) aligns with staged power gating, where AMX transitions through intermediate power states before full gating.

The state transitions induce latency shifts ranging from 50 to over 20,000 cycles, and corresponding changes in package-level power consumption, confirming the presence of an undocumented, staged power gating. Furthermore, we observed these effects even inside Intel SGX enclaves, indicating that AMX power residency is not confined by enclave or OS-level privilege boundaries.

Therefore, we attribute the root cause of GateBleed to a novel form of *unprivileged, AMX-local power gating*—a distinct microarchitectural mechanism not captured by any previously documented covert-channel primitive. This design-level behavior bypasses defenses targeting traditional timing channels (e.g., cache, TLB, SMT, or DVFS) and unlocks fundamentally new capabilities for attackers inherent in Intel AMX hardware implementation. These capabilities include bypassing defenses designed for cache and TLB-based covert channels, circumventing DVFS-based attack defenses, overcoming noise-based detection differences, evading detection due to high-magnitude timing leakage, achieving single-instruction activation, and operating securely within contexts such as Intel SGX enclaves.

5 GateBleed Attack

We present the GateBleed attack, its building blocks, and GateBleed gadgets. We identify real-world examples of GateBleed gadgets in Table 1.

5.1 Attack Building Blocks

Intel AMX accelerates both training and inference workloads across AI applications. Given AMX's performance profile, most modern neural networks, Transformers, GNNs, expert models, and early-exit CNNs routinely dispatch heavy matrix multiplications (matmuls) through AMX hardware. If these matrix operations are triggered conditionally based on a secret—such as token routing in a mix of experts (MoE) model, prediction confidence thresholds in an early exit model, or key presence in a KV cache—the hardware produces timing differences that correlate with the internal model state.

We define a *GateBleed Gadget* as an execution path in code that results in the triggering of Intel AMX instructions (e.g., TDPBSSD) based on a sensitive, private, or input-dependent decision variable.

It follows a three-phase sequence:

(1) Reset phase: Ensures that AMX is in a lower power state.

(2) Trigger Phase: The Victim ML conditionally executes an AMX instruction based on the private value.

- If `secret_bit = 1`, execute an AMX operation from a cold state, inducing a high-latency transition.
- If `secret_bit = 0`, either skip AMX or execute from a warm state, causing minimal delay.

(3) Measure phase: The attacker measures the response time of the sender. If low, the receiver infers a 0; otherwise, 1.

5.2 Generic Magnifier

GateBleed also introduces a single-instruction magnification gadget that amplifies subtle microarchitectural timing differences into measurable delays, even under coarse timing conditions (e.g., 5 µs granularity in Chrome's `performance.now()`). This enables attacks in restricted environments such as browsers, edge virtual machines, or WebAssembly runtimes, where high-resolution timers and privileged instructions are not available. Intel AMX is not currently accessible from browser sandboxes, but we anticipate that as ML moves to the edge, accelerators will be accessible from these restricted environments.

Unlike Hacky Racers [143], which requires long instruction streams and exhibit a high microarchitectural footprint detectable by side-channel defense mechanisms, our approach leverages the reuse-distance-dependent latency of Intel AMX matrix multiplication instructions. Specifically, when the AMX unit is power-gated after a period of inactivity, a subsequent instruction such as TDPBSSD incurs a latency penalty of up to 20,000 cycles. We exploit this behavior by aligning the timing of an AMX instruction just before the power gate such that even a minor perturbation (e.g., from a cache miss or instruction port contention) can tip the unit into a colder power state, causing a sharp latency increase.

This setup turns otherwise unobservable microarchitectural delays (on the order of 100–200 cycles) into coarse-timer-visible effects exceeding 11,000 cycles. This effect is illustrated in Figure 6, where a typical cache hit/miss delay is amplified into a ~5.5 µs timing gap,

defeating deployed timer coarsening defenses in real-world web browsers.

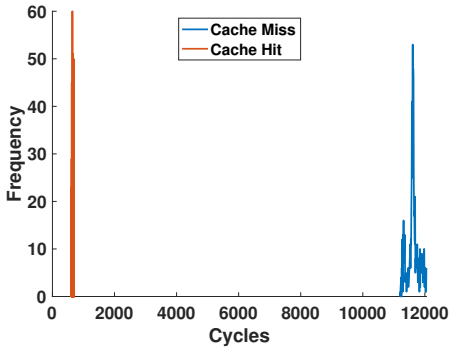


Figure 6: Timing amplification using a single AMX instruction: a 200-cycle cache miss is magnified to an ~11,000-cycle timing gap, bypassing timer resolution coarsening defenses.

The gadget requires only a single TDPBSSD invocation and avoids any speculative execution, memory flushing, or branching behavior, rendering it nearly invisible to current microarchitectural attack detectors. Its simplicity and low-profile execution pattern make it suitable for chaining with any reuse-sensitive instruction or timing channel, including cache access, port contention, and instruction ordering.

5.3 Exploitable Benign Gadgets

All listed gadgets share the same fundamental leakage mechanism. As described in Section 4.2, when an AMX instruction is invoked from a cold (power-gated) state, it incurs a substantial warm-up latency. All gadgets exploit this reuse-distance-driven latency effect: the first AMX matmul on a given execution path will be dramatically slower if the unit was idle beforehand. By making AMX invocation conditional on a secret or private branch, the timing of the code becomes correlated with the secret. Notably, this timing difference manifests even if both branches perform nominally identical workloads. For instance, even if a model tries to execute all experts in an MoE layer to avoid branching, the first semantically non-noop matmul still triggers a cold-start penalty, leaking which expert was needed. Similarly, padding or dummy computing an 'early exit' does not eliminate the initial delay when the AMX unit switches from idle to active. In every gadget, the same pattern holds: A secret-gated AMX operation creates a timing fingerprint governed by the accelerator's previous usage history (that is, the reuse distance). This uniform root cause gives us confidence that any such conditional-AMX code path can exhibit GateBleed leakage.

We identify GateBleed gadgets by searching ML libraries for the existence of branches leading to a matrix operation. The matrix multiplication can be compiled to be optimized with Intel AMX. We grouped identified gadgets into three classes: (1) *Token-routing branches* (e.g., MoE experts, tool dispatch); (2) *Confidence-gated exits* (e.g., early-exit classifiers); and (3) *Session/context-sensitive toggles* (e.g., KV-cache, quantization paths). Only classes (1) and (2) are input-dependent and security-sensitive; class (3) enables attacker-agnostic fingerprinting. Notably, all trigger real AMX matmuls.

These are not speculative code samples, but production-grade code where secret-dependent variables (e.g., confidence scores, routing decisions, session flags) conditionally guard high-throughput matrix computations optimized with AMX backends such as oneDNN and MLAS. In each case, a measurable power-gated latency difference emerges from the first semantically meaningful AMX instruction, even under balanced control flow.

GateBleed-like leakage is not a mere theoretical construct but a plausible risk across a wide range of ML libraries and models that employ conditional high-performance routines. Indeed, our investigation uncovered more than a dozen potential gadgets GateBleed in production ML frameworks. As Table 1 shows, these gadgets span real workloads in NLP (e.g., Mixtral), GNNs (e.g., RGATConv), vision (e.g., SkipNet, MSDNet), agent-based LLM frameworks (LangChain, AutoGen), and ML inference APIs (OpenAI Function Calling) across widely-used ML frameworks (e.g., Hugging Face, PyTorch, TensorFlow, ONNX Runtime, DeepSpeed).

Attribute	Value	GB	Hz	IL
P-state control	Autonomous (hardware-only)	✓	✓	✓
P-state control	Legacy (OS-only)	✓	○	✓
P-state control	Cooperative	✓	✓	✓
P-state control	Disabled	✓	○	✓
OS	RHEL 9.4	✓	✓	✓
OS	RHEL 9.5	○	✓	✓
OS	Ubuntu 22.04	✓	✓	✓
UEFI Version	SRV650-v3-3.14 (May 2024)	✓	✓	✓
UEFI Version	SRV650-v3-3.20 (June 2024)	○	✓	✓
Platform Power	Minimal Power	✓	✓	✓
Platform Power	Maximum Performance	✓	✓	✓
Platform Power	Efficiency, Favor Power	✓	✓	✓
Platform Power	Efficiency, Favor Performance	✓	✓	✓
Turbo Boost	Enabled	✓	✓	✓
Turbo Boost	Disabled	✓	**x**	✓
All prefetchers	Disabled	✓	✓	✓
C-States	Enabled	✓	✓	✓
C-States	Disabled	✓	✓	**x**
C1E	Enabled	✓	✓	✓
C1E	Disabled	✓	✓	✓

Table 2: Configuration settings for GateBleed, Hertzbleed [135], and IdleLeak [105] across various system configurations. GB, Hz, and IL refer to *GateBleed*, *Hertzbleed*, and *IdleLeak*, respectively.

6 Results

This section presents experimental settings for GateBleed results as a side-channel attack against ML models via AMX gadgets found in real-world codebases. To our knowledge, this is the first side-channel attack on machine learning privacy utilizing hardware acceleration. We then present GateBleed results as a novel passive side channel with exceptionally high bandwidth. We finally present GateBleed as a generic magnifier capable of amplifying subtle microarchitectural timing differences into visible differences in timer-constrained environments.

6.1 Experimental Setting

Our investigations utilized a server as a victim running Red Hat Enterprise Linux 9.4 with Linux Kernel 5.14, powered by an Intel Xeon Gold 5420+ CPU of the Sapphire Rapids microarchitecture. The network we used is a production network with an average daily traffic of tens of terabytes, employing no network isolation. The attacker/client is a Skylake desktop in remote settings. Table 2 summarizes the OS and UEFI settings we tested, along with how they affect the operations of two state-of-the-art side-channel attacks: Hertzbleed [135] and IdleLeak [105].

Table 1 catalogs a broad set of AMX-triggering gadgets. In this work, we implement realistic AMX-based PoCs for selected gadgets that represent each gadget class. These include Mixtral (HF), TensorFlow MoE, DeepSpeed MoE, ONNX Runtime MoE, Mixtral (llama.cpp) for expert routing, and BranchyNet/MSDNet-style early exit CNNs and transformers for confidence-based control.

6.2 Leaking Routing Decisions in Mixture of Transformer Experts (MoEs)

To reflect real-world deployments, our PoC implements a heterogeneous Mixture-of-Experts model with Intel AMX patterned directly after *Mixtral (HF)*, as listed in Table 1. Like Mixtral, our model activates a subset of experts per token (one out of two in our case), with AMX-accelerated matrix multiplications executed only for the selected expert. The higher-capacity expert matches the configuration in Table 3, while the lower-capacity expert has a reduced Transformer depth. This mirrors Mixtral's expert asymmetry and sparse dispatch behavior. We used a training set of 784 English sentences and a test set size of 300 English sentences to train the model.

Parameter	Expert	
	1 (High Capacity)	2 (Low Capacity)
Hidden size	256	256
Intermediate size	256	256
Number of heads	4	4
Attention type	Multi-headed	Multi-headed
Embedding size	256	256
Number of layers	24	**10–22 (varied)**
Dropout	0.1	0.1
Activation	ReLU	ReLU
Parameter sharing	None	None

Table 3: Transformer Specifications in Heterogeneous MoE.

GateBleed successfully infers the expert routing index in a heterogeneous Mixture-of-Experts Transformer with an overall accuracy of **100%**, indicating it reliably distinguishes between which expert was activated even when model parameters and logits are hidden. We perform a comprehensive sensitivity study: Figure 7 presents ROC curves for varying differences in expert depth. When the lower-capacity expert has 10–16 layers and the higher-capacity expert remains fixed at 24 layers (i.e., a layer gap of 8 or more), GateBleed achieves perfect separation: 100% true positive and true negative rates, with zero false positives or false negatives. As the layer gap narrows, leakage weakens but remains significant. Table 4 compares the success rate, FP, FN, TP, and TN rates numerically.

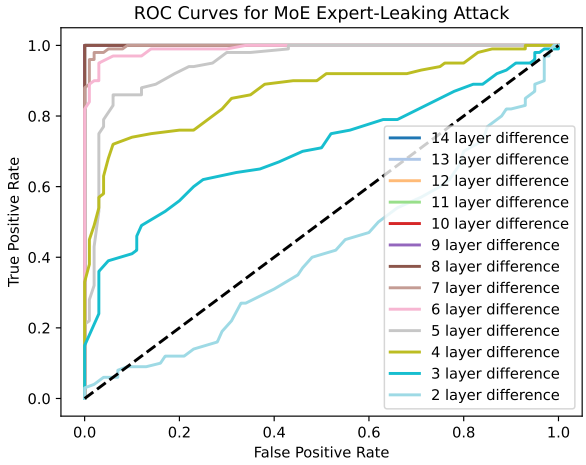


Figure 7: ROC for GateBleed attack on MoEs. 8-layer differences up to 14-layer differences have 100% accurate classification, hence they overlap over the pink curve in the plot.

6.3 Leaking Early-Exit Decisions & Membership via AMX Timing

Our PoC targets an early-exit convolutional neural network (CNN) following the structure of BranchyNet [121] described in Table 1. The model contains six layers: a convolution followed by max-pooling and ReLU, then two fully connected layers. An early-exit branch is inserted after layer 2, where the model computes a softmax over logits and exits if confidence exceeds a threshold. We use a soft threshold-based condition to simulate production behavior, and the model routes either through this shallow path or the deeper full path based on this internal decision.

A special condition under which this attack takes place is that the time taken by the early exit path and full path are essentially the same, making the timing side channel ineffective. In this attack, GateBleed achieves a classification success rate of 99.72% to infer whether the model exited early or executed the full path with the help of AMX power gating. The true positive rate—correctly identifying early exits—is 99.99%, with a false positive rate of just 0.54%. Here, we have taken the threshold as the mean of the average cycles taken by the AMX instruction following the Early Exit path and average cycles taken by the AMX instruction after the Full Path to identify the inference path. These results hold across 20,000 repeated trials.

To assess the sensitivity of the channel to architectural parameters, we vary the position of the early-exit condition and measure performance. If sufficient confidence is computed using the softmax of the logits of the exit layer, the NN computation exits. We train and evaluate this model on the MNIST dataset.

Attack	Accuracy / Success	TPR	FPR	TNR	FNR	Precision
MoE (Layer Gap ≥ 8)	100%	100%	0%	100%	0%	1.0
MoE (Layer Gap = 7)	98%	98%	2%	98%	2%	0.98
MoE (Layer Gap = 6)	96%	95%	3%	97%	5%	0.97
MoE (Layer Gap = 5)	90%	86%	6%	94%	14%	0.93
MoE (Layer Gap = 4)	82%	74%	10%	90%	26%	0.88
MoE (Layer Gap = 3)	69%	62%	25%	75%	38%	0.71
MoE (Layer Gap = 2)	46%	40%	48%	52%	60%	0.45
Early Exit CNN	99.72%	99.99%	0.54%	99.46%	0.01%	0.99
Early Exit Transformer	100%	100%	0%	100%	0%	1.0
Transformer MIA	81%	78%	16%	84%	22%	0.89

Table 4: Evaluation metrics across verified categories of end-to-end attacks with GateBleed timing/AMX usage leakage.

When the exit occurs after skipping four layers, the area under the ROC curve (AUC) reaches 1.0. With three skipped layers, AUC remains above 0.997. Even when only two layers are skipped, the timing remains distinguishable enough to support an AUC of 0.85, confirming that the leakage scales predictably with the compute disparity between exit paths. At one-layer difference, the signal begins to degrade, but still retains a measurable gap.

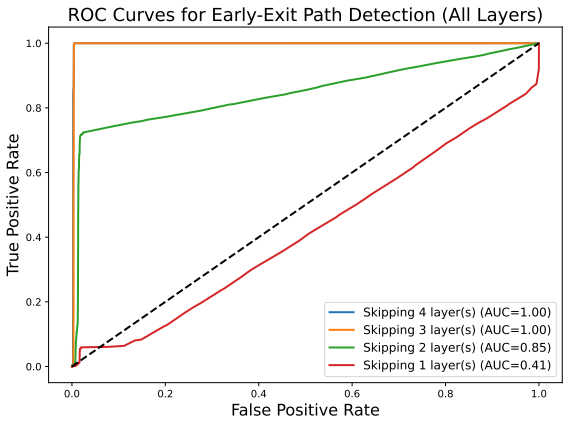


Figure 8: ROC for Early Exit CNN.

We perform an end-to-end membership inference attack using the AMX usage signal observed by GateBleed. For this experiment, we implemented an early-exit Transformer model to evaluate whether GateBleed can be used to infer training data membership through timing leakage. The model consists of 24 Transformer layers and exits after layer 12 if the softmax confidence exceeds a threshold. All matrix multiplications are dispatched to Intel AMX using `TDPBSSD`, and the architecture parameters follow Table 3. We achieve an overall accuracy of 81% with 78% of members correctly identified and only 16% of non-members misclassified. These results rival or exceed prior attacks that required full output vectors or confidence scores.

This is the first demonstration of a successful, end-to-end membership inference attack on an early-exit model deployed with hardware acceleration with no reliance on output vectors or model internals. The attacker requires only the ability to detect AMX usage, achievable via co-residency as discussed in Section 3.

6.4 Magnification for Remote Arbitrary Address Leakage

Traditional microarchitectural side channels consist of a leakage channel and transmission channel [70]. In this section, we show that GateBleed is not just a side channel targeting ML privacy—but it also serves as a highly effective transmission channel for existing microarchitectural attacks like Spectre to leak the contents of an arbitrary address, such as speculative execution vulnerabilities particularly in remote settings where prior methods fail like a realistic network.

For example, the remote Spectre attack Netspectre [115] uses the power-gating optimizations in the Intel AVX-2 and AVX-512 to transmit the leaked secret over a network. However, we find that a timing channel built on a timing difference of a few hundred cycles [115, 135] is impractical in a realistic production network.

The difference in execution between a fully powered and power-gated AVX-512 unit is about 150 cycles on Intel Xeon processors vs. 20,000 cycles for AMX. This two-order-of-magnitude gap in timing (AMX vs. AVX) provides a fundamentally stronger signal, which we show by comparing them as two covert channels in our scheme. Using an already available GateBleed gadget in the victim code can enable remote leakage of arbitrary addresses where NetSpectre fails. This is mainly because prior microarchitectural attacks have often shown effectiveness in networks where the traffic from other users was eliminated. A real network consists of uncontrolled traffic from multiple users, services, and devices, including jitter, congestion, and firewall delays. In such environments, we show that GateBleed was the only channel resilient enough to operate.

Figure 9 compares GateBleed to AVX-512 as a covert channel on the same network. Even at 1000 trials per bit, AVX-based timing differences are fully drowned in latency noise. In contrast, GateBleed maintains visible signal margins per bit due to AMX power-state transitions of up to 20,000 cycles.

With a GateBleed as a transmission channel, we leak arbitrary information across realistic network conditions by exploiting AMX

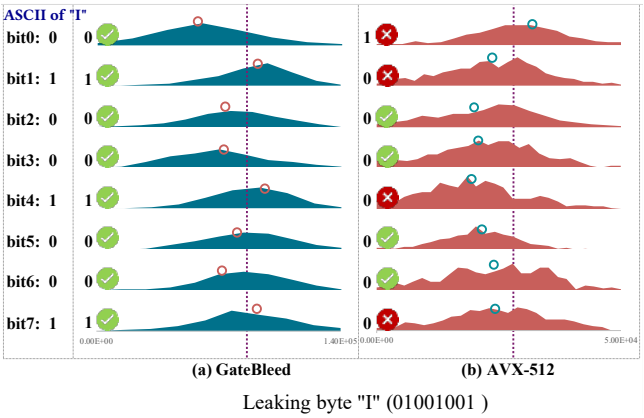


Figure 9: Leaking byte `I = 01001001` over a production network. GateBleed (left) shows separation in response-time distributions; NetSpectre with AVX-512 as transmission channel (right) fails to distinguish bits. The x-axis is response time over the network to the attacker's and the y-axis is frequency. Each row shows timing histograms per bit; dots show means.

warm-up latency. Our PoC demonstrates successful bit-wise recovery over a 1-hop Ethernet link at 0.07 bps (1 bit every 15 seconds) - a **70,000× improvement** over the 10^{-6} bps observed by AVX-512 under identical conditions. In contrast, Hertzbleed failed to leak any bits reliably across the same network, confirming that timing margins below 200 cycles collapse under real-world jitter.

In our production environment with no network isolation and tens of terabytes of daily traffic, original NetSpectre failed to reliably leak even one byte, while using GateBleed as the transmission channel, it succeeded in transmitting 8-bit secrets with high accuracy. Therefore, GateBleed as a transmission channel enables side channel attacks like Spectre to succeed in realistic remote conditions by serving as a high-bandwidth, low-noise, undetectable transmission layer.

6.5 Noise Resilience on Real Network

We define *noise resilience* as the maximum noise level at which a covert channel can maintain 99% confidence for a fixed trial count, with noise level equal to the variance of the network response time. We have previously seen in Figure 11 that the response latencies in the networks we tested can be modeled approximately as normal distributions with different variances. In Figure 10, we simulate network noise by applying additive white Gaussian noise of a particular power (x-axis). As network noise increases, the AVX-512 covert channel experiences a sharp decline in accuracy, approaching 0% almost immediately. In contrast, GateBleed maintains full resilience up to our measured 1-hop connection variance, significantly outperforming the state of the art. Note that our measured 1-hop environment had $\sigma \approx 30,000$ cycles.

Figure 11 shows that exploiting the AVX-512 power gating fails on a 1-hop network connection with an entirely overlapping distribution for the secret bit 0/1. GateBleed leaks with high performance and stealth in both a local and a realistic production network.

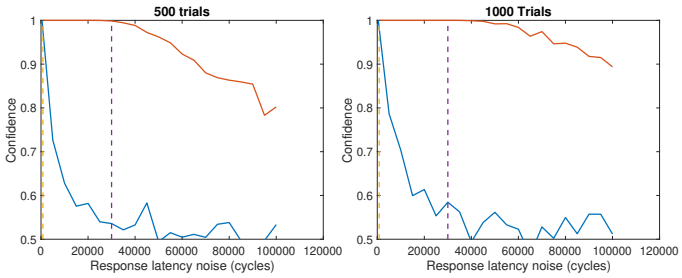


Figure 10: Noise resilience at 500 trials and 1000 trials. The orange line is GateBleed, the blue line is AVX-512, the yellow dotted line is the localhost noise level, and the purple dotted line is our 1-hop noise level.

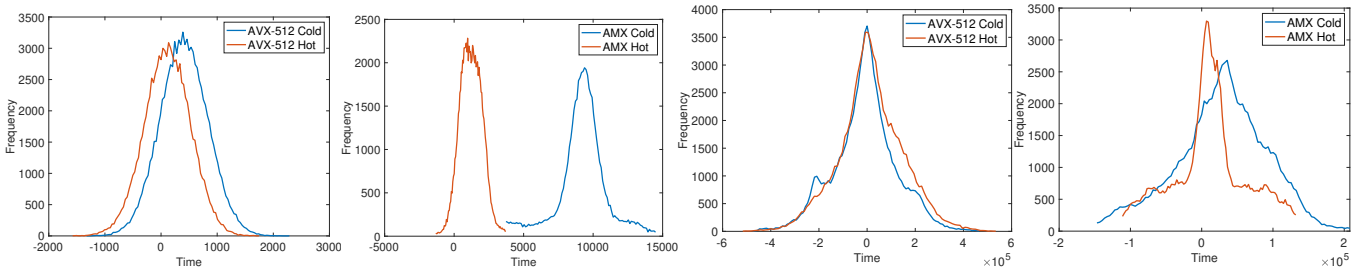
6.6 Timer Coarsening

GateBleed circumvents timer coarsening by exploiting AMX power-gating stages, which introduce latency shifts as large as 20,000 cycles. To suppress this channel, the timer resolution must be degraded to 10 μs—a 20,000× coarsening over the 0.5 ns TSC in Sapphire Rapids - far beyond what is deployed in real systems (e.g., 5 μs in Chrome [7]).

Our results show that AVX-512 and prior side channels (e.g., Hertzbleed) collapse under even moderate timer coarsening and network noise. GateBleed, by contrast, maintains a detectable signal at resolutions where others fail. Even when operating with only 500 trials, it achieves 99% classification confidence, demonstrating that AMX's latency gap acts as a built-in timing magnifier, defeating traditional timer-based defenses. This makes it the only channel in our evaluation that consistently achieves high-confidence leakage under production-like noise and coarse timer constraints. These results validate that power-state transitions in AMX accelerator create a far stronger and more resilient timing source than AVX and traditional microarchitectural power based effects.

6.7 Stealth Study

GateBleed completely eludes state-of-the-art HPC-based detection systems by leaving no observable microarchitectural footprint: no cache activity, TLB usage, or branch mispredictions. Contemporary detectors such as EVAX [10], PerSpectron [93], and RHMD [68] rely on frequent performance counter sampling to flag anomalies such as cache misses, branch mispredictions, or TLB faults. These approaches are effective against conventional covert channels, including Flush+Flush [46], Binoculars [151], and HackyRacers [143], all of which inherently produce repetitive and visible side effects. In contrast, GateBleed uses only a single AMX instruction after a passive reset phase and does not invoke any microarchitecturally anomalous instructions. With no clflush, TLB thrashing, or high-rate events, the attack resembles benign idle behavior from the detector's perspective. The only AMX-related hardware performance counter is `EXE.AMX_BUSY` which counts the number of cycles in which Intel AMX was used; we found that including this performance counter did not improve the models' performances.



(a) AVX-512 localhost (b) GateBleed localhost (c) AVX-512 1 hop (d) GateBleed 1 hop

Figure 11: Comparison of local vs. remote side channel attack timing observability.

Even after extensive retraining, these detectors do not detect GateBleed with useful accuracy. As shown in Table 5, EVAX, PerSpectron, and RHMD achieve less than 10% accuracy on GateBleed (despite being retrained on 100 million labeled samples), while achieving more than 90% accuracy on detectable channels. This ineffectiveness is due to the nature of GateBleed itself: it takes advantage of the architectural latency of the AMX power-gate rather than any detectable microarchitectural side effect. The attacker merely waits for AMX to idle naturally and then issues a single matrix multiplication, producing a measurable latency gap with no suspicious footprint. This low-instruction, low-repetition channel fundamentally bypasses the pattern recognition logic of current detection techniques, rendering GateBleed effectively invisible to today's HPC-based side-channel defenses.

7 Countermeasures

Based on the root cause analysis in the section 4.3, methods such as disabling TurboBoost, C-states, C1E, fixing the frequency, disabling RAPL, adding noise, or deploying cache defenses that mitigate prior attacks do not mitigate GateBleed. Increasing the CPU's timer resolution by 20,000x is also unacceptable. Relying on state-of-the-art microarchitectural attack detectors also fails due to the high evasiveness of GateBleed. The root cause is not speculative execution, cache usage, or DVFS—it is a hardware-level power gating state in AMX, This form of leakage operates *without speculation*, undermining traditional mitigations like LFENCE or retpolines [124].

Constant-time programming is a widely used defense against timing side channels on cryptographic algorithms, requiring that all code paths execute the same instructions regardless of secret inputs. This approach has been widely studied in the context of cryptographic routines, where several attacks have shown how cache-timing channels can leak secret keys even in seemingly secure implementations [37, 40, 41, 47, 85, 109, 145–147, 149].

For example, in a Mixture-of-Experts (MoE) model that normally activates only 2 of 16 experts per input, constant-time enforcement requires executing *all 16* experts and discarding the unused outputs. This not only increases runtime by up to 8×, but also fully activates the compute units (e.g., AMX) for each path, causing a sharp increase in both energy and latency. This makes it impractical for high-throughput AI systems.

Attack / Gadget	EVAX [10]	PerSpectron [93]	RHMD [68]
GateBleed	**10%**	**9%**	**6%**
Microscope [117]	80%	78%	63%
Flush+Flush [46]	99%	87%	72%
Binoculars [151]	98%	97%	85%
NetSpectre [115]	97%	95%	94%
Hacky Racers [143]	100%	98%	90%

Table 5: Detection accuracy of state-of-the-art detectors on known covert channels and magnifiers. GateBleed remains undetected by all three, despite retraining.

7.1 Proposed Defenses & Trade-offs

We discuss multiple defense strategies: stage locking (always-on or fixed-state AMX), context switch-aware resetting, and hardware- or firmware-level redesigns.

7.1.1 ***Always-Warm vs. Always-Cold Trade-offs.*** In the first class of defenses, one can enforce a fixed AMX power stage throughout execution. For example, keeping AMX always warm by forcing it into the warm state eliminates any warm-up delays and thus fully masks timing variation. However, this defense imposes the highest power overhead, reaching 12% in our measurement.

On the other extreme, keeping AMX fully cold at Power Gate 4 yields no additional power draw, but results in the worst-case performance penalty of 35%, as each AMX operation incurs the maximum cold-start latency of 10 μs. Intermediate fixed-stage settings offer tunable tradeoffs.

For instance, Power Gate-1 reduces power overhead to 8.1% while still preserving performance, with only a 2.5% execution time increase. Power Gate-2 further lowers power usage to 5%, though with higher latency overhead (11.1%). This pattern demonstrates that fixed-stage defenses offer a spectrum of options with a trade-off between energy and speed. These results are shown in Figure 12.

7.1.2 ***Context Switch-Aware Mitigation (OS-level).*** A second class of defenses leverages the operating system to reset AMX state on each context switch, preventing information leakage between users or VMs. This OS-level mitigation can be implemented by issuing a `TILERELEASE` or similar reset instruction during task switching, which guarantees that each process starts from the coldest AMX state. In scenarios where AMX-based models are co-resident (e.g.,

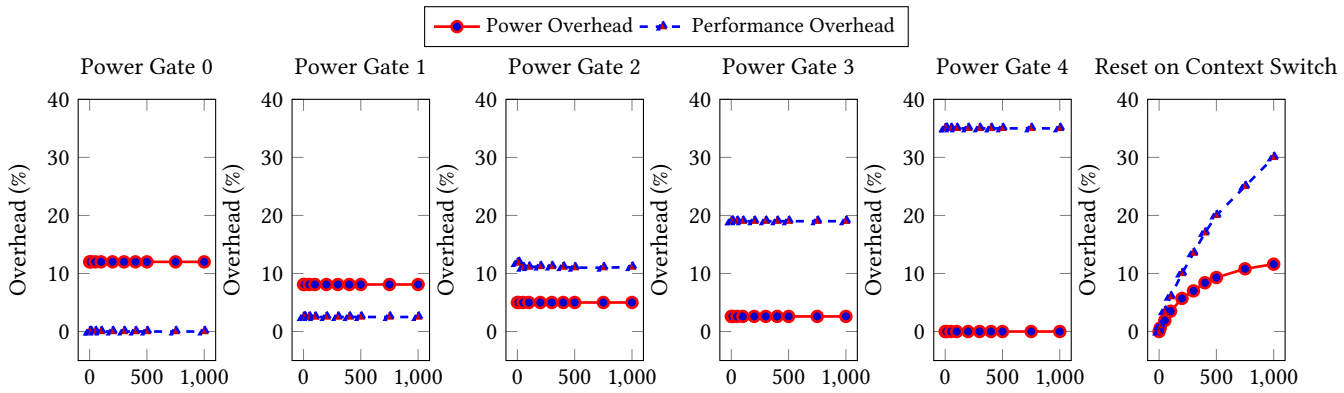


Figure 12: Comparison of AMX mitigation strategies. Each subplot shows the power (solid red) and performance (dashed blue) overhead as a function of context switch rate (per sec).

MLaaS environments, containers, enclaves), this prevents reuse-based side channels. However, this comes at a dynamic cost: if context switches are frequent, the cold-start latency is repeatedly reintroduced. *At low switching rates (e.g., <10 switches/sec), this overhead is negligible—less than 2% for both power and performance.*

But as shown in Figure 12, as the switch rate approaches 1000/sec, power cost climbs to 11.6% and performance overhead reaches 30%, closely matching the always-cold extremes. Unlike fixed-stage defenses, however, this approach maintains AMX's power-saving behavior for workloads that are not switching often. This provides a tunable tradeoff for secure, multi-tenant systems, with low impact in realistic workloads. This OS-level strategy offers a practical, efficient mitigation for shared-core deployments, isolating AMX timing state without permanent power-on cost.

For workloads with predictable AMX usage patterns, compiler support could inject dummy AMX instructions in conditional paths to maintain constant-time behavior. This compiler-level padding can be selectively applied only to known leakage-prone structures such as MoE dispatch and early-exit classifiers. Finally, hardware vendors should consider integrating a secure runtime control plane that allows compilers or OSes to set AMX residency policy directly—e.g., "warm mode", "reset-on-swap", or "cold-safe"—to reflect the sensitivity and latency demands of the running code. One option is to modify the AMX microcode so that every `TMUL` instruction—regardless of prior usage—executes at a fixed latency. Alternatively, AMX's power-gating transitions could be smoothed or disabled to keep it semi-active without a full shutdown.

Future work should refine Intel AMX power management to balance security, power, and performance, considering GateBleed attacks.

8 Conclusion

We present a security analysis of Intel AMX and reveal a novel timing vulnerability GateBleed, which exploits reuse-distance-dependent latency caused by power gating to leak information across OS, VM, and enclave boundaries with high signal strength and minimal attacker control.

In the ML domain, developers of sensitive models (e.g., private or MLaaS deployments) must now consider timing leaks related to power optimization, potentially requiring defenses like the proposed defense or model logic redesign. For instance, MoE or early-exit networks may need to be avoided or confined to low-risk contexts. The discovery of gadgets in widely used frameworks means library maintainers may need to patch the relevant code. OS and hardware manufacturers may need to get updated and issue guidance (e.g., resetting AMX during context switch, inserting dummy AMX ops, or disabling AMX within enclaves) and consider more secure designs for Intel AMX power optimization to mitigate this risk.

Acknowledgments

The authors thank anonymous reviewers for their helpful comments and feedback. This work was supported by Semiconductor Research Corporation (SRC) contract #2025-HW-3306 and Intel Labs.

References

[1] [n. d.]. 4th Gen Intel® Xeon® Scalable Processors. https://www.intel.com/content/dam/www/central-libraries/us/en/documents/2023-09/4th-gen-xeon-revised-product-brief.pdf. [Accessed 01-04-2025].
[2] [n. d.]. AI and compute. https://openai.com/index/ai-and-compute/. [Accessed 01-04-2025].
[3] [n. d.]. Getty Images (US) Inc and Others v. Stability AI Ltd [2025] EWHC 38 (Ch). https://www.judiciary.uk/judgments/getty-images-and-others-v-stability-ai/ Neutral citation: [2025] EWHC 38 (Ch).
[4] [n. d.]. GitHub - ggml-org/ggml: Tensor library for machine learning — github.com. https://github.com/ggml-org/ggml. [Accessed 13-06-2025].
[5] [n. d.]. GitHub - keras-team/keras: Deep Learning for humans — github.com. https://github.com/keras-team/keras. [Accessed 12-04-2025].
[6] [n. d.]. Thomson Reuters Enterprise Centre GmbH *et al.* v. ROSS Intelligence Inc. https://www.ded.uscourts.gov/sites/ded/files/opinions/20-613_5.pdf Memorandum Opinion.
[7] 2022. Feature: Align performance API timer resolution to cross-origin isolated capability. https://chromestatus.com/feature/6497206758539264. [Accessed 11-09-2024].
[8] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 308–318.
[9] Hojjat Aghakhanii, Dongyu Meng, Yu-Xiang Wang, Christopher Kruegel, and Giovanni Vigna. 2021. Bullseye Polytope: A Scalable Clean-Label Poisoning Attack with Improved Transferability.
[10] Samira Mirbagher Ajorpaz, Daniel Moghimi, Jeffrey Neal Collins, Gilles Pokam, Nael Abu-Ghazaleh, and Dean Tullsen. 2022. EVAX: Towards a Practical, Proactive & Adaptive Architecture for High Performance & Security. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 1218–1236. https://doi.org/10.1109/MICRO56248.2022.00085

[11] Ayomide Akinsanya and Tegan Brennan. 2024. Timing Channels in Adaptive Neural Networks. *Proceedings 2024 Network and Distributed System Security Symposium* (2024). https://api.semanticscholar.org/CorpusID:267617613
[12] Alejandro Cabrera Aldaya, Billy Bob Brumley, Sohaib ul Hassan, Cesar Pereida García, and Nicola Tuveri. 2018. *Port Contention for Fun and Profit.* Technical Report. Available from https://eprint.iacr.org/2018/1060.pdf.
[13] Manish Arora, Srilatha Manne, Indrani Paul, Nuwan Jayasena, and Dean M Tullsen. 2015. Understanding idle behavior and power gating mechanisms in the context of modern benchmarks on cpu-gpu integrated systems. In *2015 IEEE 21st international symposium on high performance computer architecture (HPCA)*. IEEE, 366–377.
[14] Lejla Batina, Shivam Bhasin, Dirmanto Jap, and Stjepan Picek. 2019. CSI{NN}: Reverse engineering of neural network architectures through electromagnetic side channel. In *28th USENIX Security Symposium (USENIX Security '19)*. 515–532.
[15] Atri Bhattacharyya, Alexandra Sandulescu, Matthias Neugschwandtner, Alessandro Sorniotti, Babak Falsafi, Mathias Payer, and Anil Kurmus. 2019. SMoTherSpectre: exploiting speculative execution through port contention. *arXiv preprint arXiv:1903.01843* (2019).
[16] Tegan Brennan, Nicolas Rosner, and Tevfik Bultan. 2020. JIT Leaks: Inducing timing side channels through just-in-time compilation. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1207–1222. https://doi.org/10.1109/SP40000.2020.00069
[17] Claudio Canella, Daniel Genkin, Lukas Giner, Daniel Gruss, Moritz Lipp, Marina Minkin, Daniel Moghimi, Frank Piessens, Michael Schwarz, Berk Sunar, et al. 2019. Fallout: Leaking data on meltdown-resistant CPUs. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 769–784.
[18] Claudio Canella, Jo Van Bulck, Michael Schwarz, Moritz Lipp, Benjamin von Berg, Philipp Ortner, Frank Piessens, Dmitry Evtyushkin, and Daniel Gruss. 2019. A Systematic Evaluation of Transient Execution Attacks and Defenses. In *USENIX Security Symposium*.
[19] Nicholas Carlini, Jorge Chávez-Saab, Anna Hambitzer, Francisco Rodríguez-Henríquez, and Adi Shamir. 2024. Polynomial Time Cryptanalytic Extraction of Deep Neural Networks in the Hard-Label Setting. *arXiv preprint arXiv:2410.05750* (2024).
[20] Nicholas Carlini, Matthew Jagielski, Christopher A. Choquette-Choo, Daniel Paleka, Will Pearce, Hyrum Anderson, Andreas Terzis, Kurt Thomas, and Florian Tramer. 2024. Poisoning Web-Scale Training Datasets is Practical . In *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Los Alamitos, CA, USA, 407–425. https://doi.org/10.1109/SP54263.2024.00179
[21] Nicholas Carlini, Matthew Jagielski, and Ilya Mironov. 2020. Cryptanalytic Extraction of Neural Network Models. In *Advances in Cryptology – CRYPTO 2020*, Daniele Micciancio and Thomas Ristenpart (Eds.). Springer International Publishing, Cham, 189–218.
[22] Varun Chandrasekaran, Kamalika Chaudhuri, Irene Giacomelli, Somesh Jha, and Songbai Yan. 2020. Exploring Connections Between Active Learning and Model Extraction. In *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, 1309–1326. https://www.usenix.org/conference/usenixsecurity20/presentation/chandrasekaran
[23] Boru Chen et al. 2024. GoFetch: Breaking constant-time cryptographic implementations using data memory-dependent prefetchers. In *USENIX Security*.
[24] Christopher A Choquette-Choo, Florian Tramer, Nicholas Carlini, and Nicolas Papernot. 2021. Label-only membership inference attacks. In *International conference on machine learning*. PMLR, 1964–1974.
[25] PyTorch Contributors. 2024. *PyTorch: An open source machine learning framework.* PyTorch Foundation. https://github.com/pytorch/pytorch Accessed: 2025-03-28.
[26] DeepSpeed. 2023. Mixture of Experts (MoE). https://deepspeed.readthedocs.io/en/latest/moe.html.
[27] PyG Developers. 2024. *PyG: PyTorch Geometric – Graph Neural Network Library.* PyG Team. https://github.com/pyg-team/pytorch_geometric Accessed: 2025-03-28.
[28] Ruyi Ding, Tianhong Xu, Xinyi Shen, Aidong Adam Ding, and Yunsi Fei. 2025. MoEcho: Exploiting Side-Channel Attacks to Compromise User Privacy in Mixture-of-Experts LLMs. *arXiv preprint arXiv:2508.15036* (2025).
[29] Farshad Dizani, Azam Ghanbari, Joshua Kalyanapu, Darsh Asher, and Samira Mirbagher Ajorpaz. 2025. Thor: A Non-Speculative Value Dependent Timing Side Channel Attack Exploiting Intel AMX. *IEEE Computer Architecture Letters* (2025).
[30] Anuj Dubey, Rosario Cammarota, and Aydin Aysu. 2020. Maskednet: The first hardware inference engine aiming power side-channel protection. In *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 197–208.
[31] Florian Dubost, Gerda Bortsova, Hieab Adams, M Arfan Ikram, Wiro Niessen, Meike Vernooij, and Marleen de Bruijne. 2019. Hydranet: data augmentation for regression neural networks. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2019: 22nd International Conference, Shenzhen, China, October 13–17, 2019, Proceedings, Part IV 22*. Springer, 438–446.
[32] Vasisht Duddu, Antoine Boutet, and Virat Shejwalkar. 2020. Quantifying privacy leakage in graph embedding. In *MobiQuitous 2020-17th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*. 76–85.
[33] Vasisht Duddu, Debasis Samanta, D. Vijay Rao, and Valentina E. Balas. 2018. Stealing neural networks via timing side channels. arXiv preprint arXiv:1812.11720.
[34] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*. Springer, 265–284.
[35] European Data Protection Board. 2024. *Opinion 28/2024 on certain data protection aspects related to the processing of personal data in the context of AI models.* Opinion 28/2024. European Data Protection Board (EDPB), Brussels, Belgium. https://www.edpb.europa.eu/system/files/2024-12/edpb_opinion_202428_ai-models_en.pdf Adopted on 17 December 2024; issued pursuant to Article 64(2) GDPR.
[36] Dmitry Evtyushkin, Ryan Riley, Nael Abu-Ghazaleh, and Dmitry Ponomarev. 2018. BranchScope: A new side-channel attack on directional branch predictors. In *ASPLOS*.
[37] Cesar Pereida García and Billy Bob Brumley. 2017. Constant-Time Callees with Variable-Time Callers. In *26th USENIX Security Symposium (USENIX Security 17)*. USENIX Association, Vancouver, BC, 83–98. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/garcia
[38] Eva García-Martín, Crefeda Faviola Rodrigues, Graham Riley, and Håkan Grahn. 2019. Estimation of energy consumption in machine learning. *J. Parallel and Distrib. Comput.* 134 (2019), 75–88.
[39] S. Gast, J. Juffinger, M. Schwarzl, G. Saileshwar, A. Kogler, S. Franza, M. Köstl, and D. Gruss. 2023. SQUIP: Exploiting the Scheduler Queue Contention Side Channel. In *2023 2023 IEEE Symposium on Security and Privacy (SP) (SP)*. IEEE Computer Society, Los Alamitos, CA, USA, 468–484. https://doi.org/10.1109/SP46215.2023.00027
[40] Daniel Genkin, Lev Pachmanov, Eran Tromer, and Yuval Yarom. 2018. Drive-By Key-Extraction Cache Attacks from Portable Code. In *Applied Cryptography and Network Security: 16th International Conference, ACNS 2018, Leuven, Belgium, July 2-4, 2018, Proceedings* (Leuven, Belgium). Springer-Verlag, Berlin, Heidelberg, 83–102. https://doi.org/10.1007/978-3-319-93387-0_5
[41] Daniel Genkin, Luke Valenta, and Yuval Yarom. 2017. May the Fourth Be With You: A Microarchitectural Side Channel Attack on Several Real-World Applications of Curve25519. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (Dallas, Texas, USA) *(CCS '17)*. Association for Computing Machinery, New York, NY, USA, 845–858. https://doi.org/10.1145/3133956.3134029
[42] Georgi Gerganov and Contributors. 2023. *llama.cpp: Efficient CPU inference of Meta's LLaMA model.* ggerganov. https://github.com/ggerganov/llama.cpp Accessed: 2025-03-28.
[43] Neil Zhenqiang Gong and Bin Liu. 2018. Attribute inference attacks in online social networks. *ACM Transactions on Privacy and Security (TOPS)* 21, 1 (2018), 1–30.
[44] Ben Gras, Kaveh Razavi, Herbert Bos, and Cristiano Giuffrida. 2018. Translation leak-aside buffer: Defeating cache side-channel protections with TLB attacks. In *USENIX Security*.
[45] Daniel Gruss, Clémentine Maurice, Anders Fogh, Moritz Lipp, and Stefan Mangard. 2016. Prefetch side-channel attacks: Bypassing SMAP and kernel ASLR. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 368–379.
[46] Daniel Gruss, Clémentine Maurice, and Stefan Mangard. 2016. Flush+Flush: A fast and stealthy cache attack. In *DIMVA*.
[47] David Gullasch, Endre Bangerter, and Stephan Krenn. 2011. Cache Games – Bringing Access-Based Cache Attacks on AES to Practice. In *2011 IEEE Symposium on Security and Privacy*. 490–505. https://doi.org/10.1109/SP.2011.22
[48] Umang Gupta, Dimitris Stripelis, Pradeep K Lam, Paul Thompson, Jose Luis Ambite, and Greg Ver Steeg. 2021. Membership inference attacks on deep regression models for neuroimaging. In *Medical Imaging with Deep Learning*. PMLR, 228–251.
[49] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. 2017. Logan: Membership inference attacks against generative models. *arXiv preprint arXiv:1705.07663* (2017).
[50] Yu He, Boheng Li, Liu Liu, Zhongjie Ba, Wei Dong, Yiming Li, Zhan Qin, Kui Ren, and Chun Chen. 2025. Towards label-only membership inference attack against pre-trained large language models. In *USENIX Security*.
[51] Benjamin Hilprecht, Martin Härterich, and Daniel Bernau. 2019. Monte carlo and reconstruction membership inference attacks against generative models. *Proceedings on Privacy Enhancing Technologies* (2019).
[52] Jann Horn. 2018. speculative execution, variant 4: speculative store bypass. https://bugs.chromium.org/p/project-zero/issues/detail?id=1528. Accessed: 2023-04-22.
[53] Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S Yu, and Xuyun Zhang. 2022. Membership inference attacks on machine learning: A survey.

ACM Computing Surveys (CSUR) 54, 11s (2022), 1–37.

[54] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q. Weinberger. 2017. Multi-Scale Dense Networks for Resource Efficient Image Classification. *arXiv preprint arXiv:1703.09844* (2017). https://arxiv.org/abs/1703.09844

[55] Quzhe Huang, Zhenwei An, Nan Zhuang, Mingxu Tao, Chen Zhang, Yang Jin, Kun Xu, Liwei Chen, Songfang Huang, and Yansong Feng. 2024. Harder tasks need more experts: Dynamic routing in moe models. *arXiv preprint arXiv:2403.07652* (2024).

[56] Hugging Face Inc. 2024. *Transformers: State-of-the-art Natural Language Processing for PyTorch and TensorFlow*. Hugging Face. https://github.com/huggingface/transformers Accessed: 2025-03-28.

[57] Information Commissioner's Office (ICO). 2023. *Guidance on AI and Data Protection (version 2.0.38)*. Guidance. Information Commissioner's Office (UK). https://ico.org.uk/media2/ga4lfb5d/guidance-on-ai-and-data-protection-all-2-0-38.pdf Updated on 15 March 2023; version 2.0.38.

[58] Intel. 2021. *Intel Architecture Instruction Set Extensions and Future Features Programming Reference*. Available: https://www.intel.com/content/dam/develop/external/us/en/documents/architecture-instruction-set-extensions-programming-reference.pdf.

[59] Intel. 2023. Advanced Matrix Extensions (AMX) for AI Acceleration. Intel Corporation. https://www.intel.com/content/www/us/en/products/docs/accelerator-engines/advanced-matrix-extensions/ai-solution-brief.html Accessed: 2023-04-12.

[60] Intel. 2024. Intel® 64 and IA-32 Architectures Optimization Reference Manual. https://www.intel.com/content/www/us/en/content-details/814198/intel-64-and-ia-32-architectures-optimization-reference-manual-volume-1.html.

[61] Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. 2020. High Accuracy and High Fidelity Extraction of Neural Networks. arXiv:1909.01838 [cs.LG] https://arxiv.org/abs/1909.01838

[62] Bargav Jayaraman and David Evans. 2019. Evaluating Differentially Private Machine Learning in Practice. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association, Santa Clara, CA, 1895–1912. https://www.usenix.org/conference/usenixsecurity19/presentation/jayaraman

[63] Bargav Jayaraman and David Evans. 2022. Are Attribute Inference Attacks Just Imputation?. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* (Los Angeles, CA, USA) *(CCS '22)*. Association for Computing Machinery, New York, NY, USA, 1569–1582. https://doi.org/10.1145/3548606.3560663

[64] Andrew Jeong. [n. d.]. Federal court says copyrighted books are fair use for AI training. https://www.washingtonpost.com/technology/2025/06/25/ai-copyright-anthropic-books/?utm_source=chatgpt.com. *The Washington Post* ([n. d.]). Accessed: 2025-09-17.

[65] Dhiraj Kalamkar, Dheevatsa Mudigere, Naveen Mellempudi, Dipankar Das, Kunal Banerjee, Sasikanth Avancha, Dharma Teja Vooturi, Nataraj Jammalamadaka, Jianyu Huang, Hector Yuen, et al. 2019. A study of BFLOAT16 for deep learning training. *arXiv preprint arXiv:1905.12322* (2019).

[66] Vijay Kandiah, Scott Peverelle, Mahmoud Khairy, Junrui Pan, Amogh Manjunath, Timothy G Rogers, Tor M Aamodt, and Nikos Hardavellas. 2021. AccelWattch: A power modeling framework for modern GPUs. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*. 738–753.

[67] Hitesh Khandelwal, Viet Ha-Thuc, Avishek Dutta, Yining Lu, Nan Du, Zhihao Li, and Qi Hu. 2021. Jointly Optimize Capacity, Latency and Engagement in Large-scale Recommendation Systems. In *Proceedings of the 15th ACM Conference on Recommender Systems*. 559–561.

[68] Khaled N. Khasawneh, Nael Abu-Ghazaleh, Dmitry Ponomarev, and Lei Yu. 2017. RHMD: Evasion-Resilient Hardware Malware Detectors. In *Annual IEEE/ACM International Symposium on Microarchitecture*.

[69] Vladimir Kiriansky and Carl Waldspurger. 2018. Speculative buffer overflows: Attacks and defenses. *arXiv preprint arXiv:1807.03757* (2018).

[70] Vladimir Kiriansky, Carl Waldspurger, and Joel Emer. 2018. DAWG: Defense against wayward gossip. In *ISCA*.

[71] Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, et al. 2020. Spectre attacks: Exploiting speculative execution. *Commun. ACM* 63, 7 (2020), 93–101.

[72] Andreas Kogler, Jonas Juffinger, Lukas Giner, Lukas Gerlach, Martin Schwarzl, Michael Schwarz, Daniel Gruss, and Stefan Mangard. 2023. {Collide+ Power}: Leaking Inaccessible Data with Software-based Power Side Channels. In *32nd USENIX Security Symposium (USENIX Security 23)*. 7285–7302.

[73] Esmaeil Mohammadian Koruyeh, Khaled N Khasawneh, Chengyu Song, and Nael Abu-Ghazaleh. 2018. Spectre returns! speculation attacks using the return stack buffer. In *12th USENIX Workshop on Offensive Technologies (WOOT 18)*.

[74] Jakob Koschel, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. 2020. TagBleed: Breaking KASLR on the isolated kernel address space using tagged TLBs. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 309–321.

[75] Stephen Kosonocky. 2011. Practical power gating and dynamic voltage/frequency scaling. In *2011 IEEE Hot Chips 23 Symposium (HCS)*. IEEE, 1–62.

[76] Rakesh Kumar, Alejandro Martínez, and Antonio González. 2014. Efficient power gating of simd accelerators through dynamic selective devectorization in an hw/sw codesigned environment. *ACM Transactions on Architecture and Code Optimization (TACO)* 11, 3 (2014), 1–23.

[77] LangChain. 2023. LangChain: Build context-aware, reasoning applications. https://www.langchain.com/.

[78] Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*. PMLR, 19274–19286.

[79] Zheng Li, Yiyong Liu, Xinlei He, Ning Yu, Michael Backes, and Yang Zhang. 2022. Auditing membership leakages of multi-exit networks. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22)*. ACM. https://doi.org/10.1145/3548606.3559359 15 pages.

[80] Zheng Li and Yang Zhang. 2021. Membership leakage in label-only exposures. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 880–895.

[81] Moritz Lipp, Daniel Gruss, and Michael Schwarz. 2022. {AMD} prefetch attacks through power and time. In *31st USENIX Security Symposium (USENIX Security 22)*. 643–660.

[82] Moritz Lipp, Andreas Kogler, David Oswald, Michael Schwarz, Catherine Easdon, Claudio Canella, and Daniel Gruss. 2021. PLATYPUS: Software-based Power Side-Channel Attacks on x86. 355–371. https://doi.org/10.1109/SP40001.2021.00063

[83] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, et al. 2020. Meltdown: Reading kernel memory from user space. *Commun. ACM* 63, 6 (2020), 46–56.

[84] Chen Liu, Abhishek Chakraborty, Nikhil Chawla, and Neer Roggel. 2022. Frequency throttling side-channel attack. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 1977–1991.

[85] Fangfei Liu, Yuval Yarom, Qian Ge, Gernot Heiser, and Ruby B. Lee. 2015. Last-Level Cache Side-Channel Attacks are Practical. In *2015 IEEE Symposium on Security and Privacy*. 605–622. https://doi.org/10.1109/SP.2015.43

[86] Hongbin Liu, Jinyuan Jia, Wenjie Qu, and Neil Zhenqiang Gong. 2021. Encodermi: Membership inference against pre-trained encoders in contrastive learning. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 2081–2095.

[87] Kin Sum Liu, Chaowei Xiao, Bo Li, and Jie Gao. 2019. Performing co-membership attacks against deep generative models. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 459–467.

[88] Shichen Liu, Fei Xiao, Wenwu Ou, and Luo Si. 2017. Cascade ranking for operational e-commerce search. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1557–1565.

[89] Anita Lungu, Pradip Bose, Alper Buyuktosunoglu, and Daniel J Sorin. 2009. Dynamic power gating with quality guarantees. In *Proceedings of the 2009 ACM/IEEE international symposium on Low power electronics and design*. 377–382.

[90] G. Maisuradze and C. Rossow. 2018. ret2spec: Speculative execution using return stack buffers. In *ACM Conference on Computer and Communications Security (CCS)*.

[91] Ross Mcilroy, Jaroslav Sevcik, Tobias Tebbi, Ben L. Titzer, and Toon Verwaest. 2019. Spectre is here to stay: An analysis of side-channels and speculative execution. https://arxiv.org/abs/1902.05178

[92] Shagufta Mehnaz, Sayanton V Dibbo, Ehsanul Kabir, Ninghui Li, and Elisa Bertino. 2022. Are your sensitive attributes private? novel model inversion attribute inference attacks on classification models. In *31st USENIX Security Symposium (USENIX Security 22)*. 4579–4596.

[93] S. Mirbagher-Ajorpaz, G. Pokam, E. Mohammadian-Koruyeh, E. Garza, N. Abu-Ghazaleh, and D. A. Jiménez. 2020. PerSpectron: Detecting Invariant Footprints of Microarchitectural Attacks with Perceptron. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*.

[94] Daniel Moghimi. 2023. Downfall: Exploiting speculative data gathering. In *32nd USENIX Security Symposium (USENIX Security 23)*. 7179–7193.

[95] Daniel Moghimi, Moritz Lipp, Berk Sunar, and Michael Schwarz. 2020. Medusa: Microarchitectural Data Leakage via Automated Attack Synthesis. In *USENIX Security Symposium*.

[96] Ashley O Munch, Nevine Nassif, Carleton L Molnar, Jason Crop, Rich Gammack, Chinmay P Joshi, Goran Zelic, Kambiz Munshi, Min Huang, Charles R Morganti, et al. 2024. 2.3 Emerald Rapids: 5th-Generation Intel® Xeon® Scalable Processors. In *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, Vol. 67. IEEE, 40–42.

[97] Sasi Kumar Murakonda and Reza Shokri. 2020. Ml privacy meter: Aiding regulatory compliance by quantifying the privacy risks of machine learning. *arXiv preprint arXiv:2007.09339* (2020).

[98] Nevine Nassif, Ashley O Munch, Carleton L Molnar, Gerald Pasdast, Sitaraman V Lyer, Zibing Yang, Oscar Mendoza, Mark Huddart, Srikrishnan Venkataraman,

Sireesha Kandula, et al. 2022. Sapphire rapids: The next-generation intel xeon scalable processor. In *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, Vol. 65. IEEE, 44–46.

[99] OpenAI. 2023. Function calling - OpenAI API. https://platform.openai.com/docs/guides/gpt/function-calling.

[100] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. 2018. Knockoff Nets: Stealing Functionality of Black-Box Models. *CoRR* abs/1812.02766 (2018). arXiv:1812.02766 http://arxiv.org/abs/1812.02766

[101] Dag Arne Osvik, Adi Shamir, and Eran Tromer. 2006. Cache attacks and countermeasures: the case of AES. In *CT-RSA*.

[102] Nicolas Papernot, Patrick Mcdaniel, and Ian J. Goodfellow. 2016. Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples. *ArXiv* abs/1605.07277 (2016). https://api.semanticscholar.org/CorpusID:17362994

[103] Hany Ragab, Alyssa Milburn, Kaveh Razavi, Herbert Bos, and Cristiano Giuffrida. 2021. CrossTalk: Speculative Data Leaks Across Cores Are Real. In *2021 IEEE Symposium on Security and Privacy (SP)*. 1852–1867. https://doi.org/10.1109/SP40001.2021.00020

[104] Shadi Rahimian, Tribhuvanesh Orekondy, and Mario Fritz. 2020. Sampling attacks: Amplification of membership inference attacks by repeated queries. *arXiv preprint arXiv:2009.00395* (2020).

[105] Fabian Rauscher, Andreas Kogler, Jonas Juffinger, and Daniel Gruss. 2024. Idleleak: Exploiting idle state side effects for information leakage. In *Network and Distributed System Security Symposium 2024: NDSS 2024*.

[106] Joseph Ravichandran, Weon Taek Na, Jay Lang, and Mengjia Yan. 2022. PACMAN: Attacking ARM Pointer Authentication with Speculative Execution. In *Proceedings of the 49th Annual International Symposium on Computer Architecture* (New York, New York) *(ISCA '22)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3470496.3527429

[107] Xida Ren, Logan Moody, Mohammadkazem Taram, Matthew Jordan, Dean M. Tullsen, and Ashish Venkat. 2021. I See Dead μops: Leaking Secrets via Intel/AMD Micro-Op Caches. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. 361–374. https://doi.org/10.1109/ISCA52012.2021.00036

[108] Reuters. [n. d.]. Apple sued by authors over use of books in AI training. https://www.reuters.com/sustainability/boards-policy-regulation/apple-sued-by-authors-over-use-books-ai-training-2025-09-05/. *Reuters* ([n. d.]). Accessed: 2025-09-17.

[109] Eyal Ronen, Robert Gillham, Daniel Genkin, Adi Shamir, David Wong, and Yuval Yarom. 2019. The 9 Lives of Bleichenbacher's CAT: New Cache ATtacks on TLS Implementations. In *2019 IEEE Symposium on Security and Privacy (SP)*. 435–452. https://doi.org/10.1109/SP.2019.00062

[110] ONNX Runtime. 2023. Generate API (Preview). https://onnxruntime.ai/docs/genai/.

[111] ONNX Runtime. 2023. Mixture of Experts (MoE) — DeepSpeed 0.16.6 documentation. https://deepspeed.readthedocs.io/en/latest/moe.html.

[112] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. 2018. ML-Leaks: Model and data independent membership inference attacks and defenses on machine learning models. arXiv preprint arXiv:1806.01246.

[113] Matthias Schunter. 2016. Intel Software Guard Extensions: Introduction and Open Research Challenges. In *Proceedings of the 2016 ACM Workshop on Software PROtection* (Vienna, Austria) *(SPRO '16)*. Association for Computing Machinery, New York, NY, USA, 1. https://doi.org/10.1145/2995306.2995307

[114] Michael Schwarz et al. 2019. ZombieLoad: Cross-privilege-boundary data sampling. In *ACM CCS*.

[115] Michael Schwarz, Martin Schwarzl, Moritz Lipp, and Daniel Gruss. 2018. Netspectre: Read arbitrary memory over network. *arXiv preprint arXiv:1807.10535* (2018).

[116] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 3–18. https://doi.org/10.1109/SP.2017.41

[117] Dimitrios Skarlatos, Mengjia Yan, Bhargava Gopireddy, Read Sprabery, Josep Torrellas, and Christopher W. Fletcher. 2020. MicroScope: Enabling Microarchitectural Replay Attacks. *IEEE Micro* (2020).

[118] Andrei Tatar, Daniël Trujillo, Cristiano Giuffrida, and Herbert Bos. 2022. TLB;DR: Enhancing TLB-based Attacks with TLB Desynchronized Reverse Engineering. In *USENIX Security Symposium*.

[119] Google Brain Team and TensorFlow Contributors. 2024. *TensorFlow: An end-to-end open source machine learning platform*. Google. https://github.com/tensorflow/tensorflow Accessed: 2025-03-28.

[120] Microsoft AutoGen Team. 2024. *AutoGen: Enabling Next-Gen LLM Applications with Multi-Agent Collaboration*. Microsoft. https://github.com/microsoft/autogen Accessed: 2025-03-28.

[121] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. 2016. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd international conference on pattern recognition (ICPR)*. IEEE, 2464–2469.

[122] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. 2016. Stealing Machine Learning Models via Prediction APIs. *CoRR* abs/1609.02943 (2016). arXiv:1609.02943 http://arxiv.org/abs/1609.02943

[123] Ekaterina Trimbach, Badr Abdallaoui, and Paul Missault. 2025. Cost-efficiency trade-offs for neural cascade rankers in web search. (2025). https://www.amazon.science/publications/cost-efficiency-trade-offs-for-neural-cascade-rankers-in-web-search

[124] P. Turner. 2018. Retpoline: a software construct for preventing branch-target-injection. https://support.google.com/faqs/answer/7625886.

[125] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F Wenisch, Yuval Yarom, and Raoul Strackx. 2018. Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution. In *27th USENIX Security Symposium (USENIX Security 18)*. 991–1008.

[126] Jo Van Bulck, Daniel Moghimi, Michael Schwarz, Moritz Lippi, Marina Minkin, Daniel Genkin, Yuval Yarom, Berk Sunar, Daniel Gruss, and Frank Piessens. 2020. LVI: Hijacking Transient Execution through Microarchitectural Load Value Injection. In *2020 IEEE Symposium on Security and Privacy (SP)*.

[127] Stephan van Schaik, Alyssa Milburn, Sebastian Österlund, Pietro Frigo, Giorgi Maisuradze, Kaveh Razavi, Herbert Bos, and Cristiano Giuffrida. 2019. RIDL: Rogue In-flight Data Load. In *S&P*.

[128] Raj R Varada, Rohini Krishnan, Ajith Subramonia, Rathish Chandran, Kalyana Chakravarthy, Uttpal D Desai, Sumedha Limaye, Puneesh Puri, David R Mulvihill, Mike Bichan, et al. 2025. 2.3 Granite Rapids-D: Intel Xeon 6 SoC for vRAN, Edge, Networking, and Storage. In *2025 IEEE International Solid-State Circuits Conference (ISSCC)*, Vol. 68. IEEE, 48–50.

[129] Michael Veale, Reuben Binns, and Lilian Edwards. 2018. Algorithms that remember: model inversion attacks and data protection law. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 376, 2133 (2018), 20180083.

[130] Jose Rodrigo Sanchez Vicarte, Michael Flanders, Riccardo Paccagnella, Grant Garrett-Grossman, Adam Morrison, Christopher W. Fletcher, and David Kohlbrenner. 2022. Augury: Using Data Memory-Dependent Prefetchers to Leak Data at Rest. In *2022 IEEE Symposium on Security and Privacy (SP)*. 1491–1505. https://doi.org/10.1109/SP46214.2022.9833570

[131] Jose Rodrigo Sanchez Vicarte, Pradyumna Shome, Nandeeka Nayak, Caroline Trippel, Adam Morrison, David Kohlbrenner, and Christopher W Fletcher. 2021. Opening pandora's box: A systematic study of new ways microarchitecture can leak private data. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 347–360.

[132] An Wang, Xingwu Sun, Ruobing Xie, Shuaipeng Li, Jiaqi Zhu, Zhen Yang, Pinxue Zhao, JN Han, Zhanhui Kang, Di Wang, et al. 2024. Hmoe: Heterogeneous mixture of experts for language modeling. *arXiv preprint arXiv:2408.10681* (2024).

[133] Binghui Wang and Neil Zhenqiang Gong. 2018. Stealing Hyperparameters in Machine Learning. *CoRR* abs/1802.05351 (2018). arXiv:1802.05351 http://arxiv.org/abs/1802.05351

[134] Po-Han Wang, Chia-Lin Yang, Yen-Ming Chen, and Yu-Jung Cheng. 2011. Power gating strategies on GPUs. *ACM Transactions on Architecture and Code Optimization (TACO)* 8, 3 (2011), 1–25.

[135] Yu Wang et al. 2022. Hertzbleed: Turning power side-channel attacks into remote timing attacks on x86. In *USENIX Security*.

[136] Yingchen Wang, Riccardo Paccagnella, Alan Wandke, Zhao Gang, Grant Garrett-Grossman, Christopher W Fletcher, David Kohlbrenner, and Hovav Shacham. 2023. DVFS frequently leaks secrets: Hertzbleed attacks beyond SIKE, cryptography, and CPU-only data. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2306–2320.

[137] Yiding Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E. Gonzalez. 2017. SkipNet: Learning Dynamic Routing in Convolutional Networks. *arXiv preprint arXiv:1711.09485* (2017). https://arxiv.org/abs/1711.09485

[138] Daniel Weber, Ahmad Ibrahim, Hamed Nemati, Michael Schwarz, and Christian Rossow. 2021. Osiris: Automated Discovery of Microarchitectural Side Channels. In *USENIX Security Symposium*.

[139] Lingxiao Wei, Bo Luo, Yu Li, Yannan Liu, and Qiang Xu. 2018. I know what you see: Power side-channel attack on convolutional neural network accelerators. In *Proceedings of the 34th Annual Computer Security Applications Conference*. 393–406.

[140] Yoav Weiss and W3C Web Performance Working Group. 2024. *High Resolution Time (Working Draft)*. Working Draft. World Wide Web Consortium (W3C). https://www.w3.org/TR/hr-time-3/ Accessed: 2025-09-22.

[141] Johannes Wikner, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. 2022. Spring: Spectre returning in the browser with speculative load queuing and deep stacks. In *Workshop On Offensive Technologies (WOOT)*.

[142] Johannes Wikner, Daniël Trujillo, and Kaveh Razavi. 2023. Phantom: Exploiting Decoder-detectable Mispredictions. In *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture* (<conf-loc>, <city>Toronto</city>, <state>ON</state>, <country>Canada</country>, </conf-loc>) *(MICRO '23)*. Association for Computing Machinery, New York, NY,

USA, 49–61. https://doi.org/10.1145/3613424.3614275

[143] Haocheng Xiao and Sam Ainsworth. 2023. Hacky Racers: Exploiting Instruction-Level Parallelism to Generate Stealthy Fine-Grained Timers. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2* (Vancouver, BC, Canada) *(ASPLOS 2023)*. Association for Computing Machinery, New York, NY, USA, 354–369. https://doi.org/10.1145/3575693.3575700

[144] Mengjia Yan, Christopher W. Fletcher, and Josep Torrellas. 2020. Cache telepathy: Leveraging shared resource attacks to learn DNN architectures. In *29th USENIX Security Symposium (USENIX Security '20)*. 2003–2020.

[145] Mengjia Yan, Read Sprabery, Bhargava Gopireddy, Christopher Fletcher, Roy Campbell, and Josep Torrellas. 2019. Attack Directories, Not Caches: Side Channel Attacks in a Non-Inclusive World. In *2019 IEEE Symposium on Security and Privacy (SP)*. 888–904. https://doi.org/10.1109/SP.2019.00004

[146] Yuval Yarom and Katrina Falkner. 2014. FLUSH+RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack. In *Proceedings of the 23rd USENIX Conference on Security Symposium* (San Diego, CA) *(SEC'14)*. USENIX Association, USA, 719–732.

[147] Yuval Yarom, Daniel Genkin, and Nadia Heninger. 2016. CacheBleed: a timing attack on OpenSSL constant-time RSA. *Journal of Cryptographic Engineering* 7 (2016), 99 – 112. https://api.semanticscholar.org/CorpusID:7895014

[148] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. 2018. Privacy risk in machine learning: Analyzing the connection to overfitting. In *31st IEEE Computer Security Foundations Symposium (CSF)*. IEEE, 268–282. https://doi.org/10.1109/CSF.2018.00027

[149] Yinqian Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. 2012. Cross-VM side channels and their use to extract private keys. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security* (Raleigh, North Carolina, USA) *(CCS '12)*. Association for Computing Machinery, New York, NY, USA, 305–316. https://doi.org/10.1145/2382196.2382230

[150] Benjamin Zi Hao Zhao, Aviral Agrawal, Catisha Coburn, Hassan Jameel Asghar, Raghav Bhaskar, Mohamed Ali Kaafar, Darren Webb, and Peter Dickinson. 2021. On the (in) feasibility of attribute inference attacks on machine learning models. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 232–251.

[151] Zirui Neil Zhao, Adam Morrison, Christopher W Fletcher, and Josep Torrellas. 2022. Binoculars: Contention-based side-channel attacks exploiting the page walker. In *USENIX Security*.