34th USENIX Security Symposium

I Know What You Said: Unveiling Hardware Cache Side-Channels in Local Large Language Model Inference

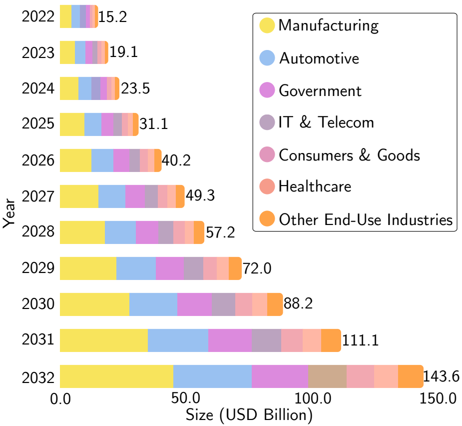
Zibo Gao[1,2], Junjie Hu[1,2], Feng Guo[1,2], Yixin Zhang[1,2], Yinglong Han[1,2], Siyuan Liu[1,2], Haiyang Li[1,2], and Zhiqiang Lv[1,2]

[1]Institute of Information Engineering, Chinese Academy of Sciences.

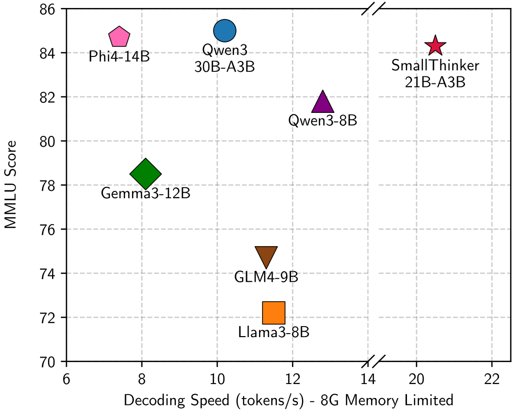
[2]School of Cyber Security, University of Chinese Academy of Sciences.

Local LLM Deployment in Today's Internet

- Escalating privacy concerns are driving the adoption of local LLMs
- Edge devices are increasingly efficient at running LLMs



Mark size of on-device AI [1]

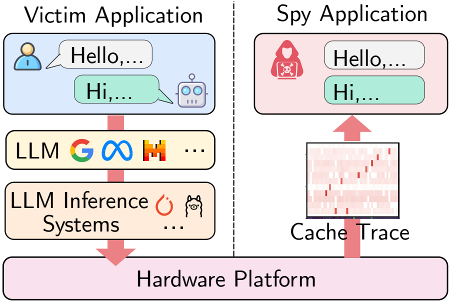


On-device LLM performance [2]

[1] J. Xu et al. On-Device Language Models: A Comprehensive Review. CoRR, 2024.
[2] SmallThinker. Technical Report, SJTU IPADS, 2025.

Local LLM (In)security

- User Belief: Local LLMs appear private and secure
- Reality: Hardware-level attacks bypass software/OS protections
- **Research Gap**: Prior LLM privacy works have not yet studied the hardware-level cache side-channel threats



Research Question

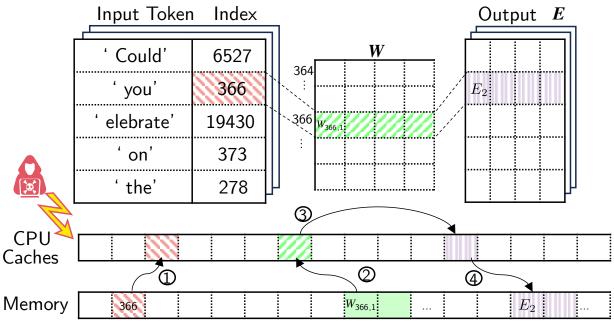
Can adversaries reconstruct user prompts and LLM responses through hardware-level cache side channels via co-located unprivileged malware?

Our Intuition

LLM's fundamental operations create deterministic, observable cache access patterns

Finding 1: Token Value Leakage

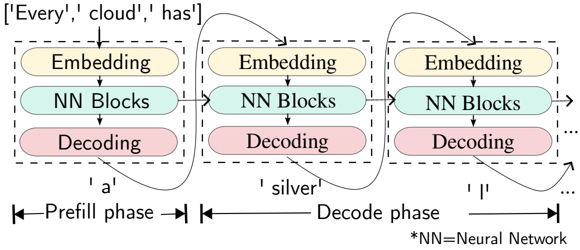
- Embedding layer acts as a lookup table
- Cache access patterns reveal token values
- Embedding is typically offloaded to CPU due to restricted GPU memory



Our Intuition

Finding 2: Token Position Leakage

- Autoregression: Prompt and response tokens both go through embedding
- Timing Signal: Response tokens unfold over multiple time steps



Put them together

Unprivileged malware on the same device can reconstruct LLM prompt and response text via observing CPU cache access patterns of the embedding layer

When Theory Meets Reality

Challenge 1

Cache side-channel noise **corrupts** the token reconstruction

- The signal-to-noise ratio (SNR) is low: 100 valid tokens/s vs. $\mathbf{5 \times 10^6}$ noise events/s when directly applying the standard Flush+Reload†
- The hardware AoP prefetcher is the root cause

- Even after overcoming the hardware prefetcher:

g C bage ertainly! Unable Here are several organi~~ge~~nic makeup brands that are exit known

False Positive

False Negative

†Using Mastik v0.02 on Intel Raptor Lake

When Theory Meets Reality

Challenge 2

Input tokens appear in **scrambled order** from the cache perspective due to parallel prefill

Observed: comm with the Rugby regional work level ways at clubs In Union national tosh does improve? the performance

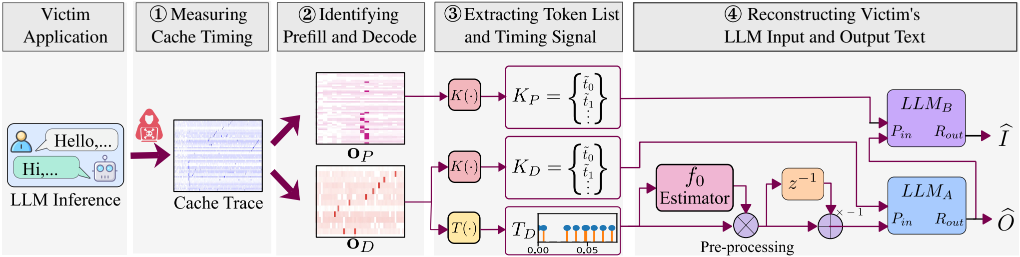
---

Original: In what ways does the Rugby Union work with regional clubs to improve performance at the national level?

Realizing the Attack: From Cache to Text

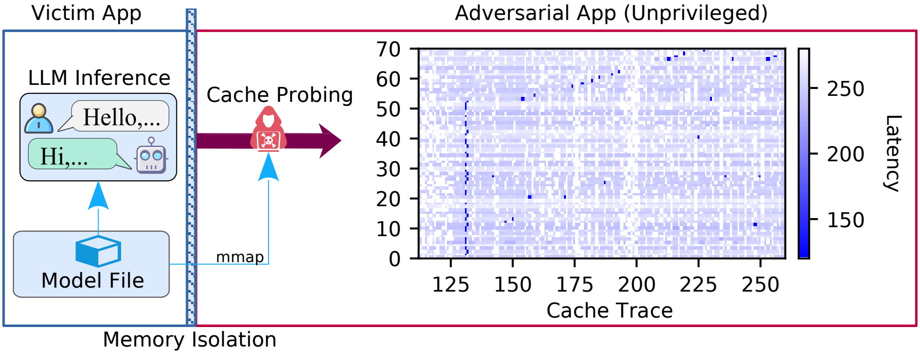
We address the aforementioned challenges via fine-tuning LLMs:

1. LLM_A: **Response Reconstruction**
2. LLM_B: **Prompt Reconstruction**



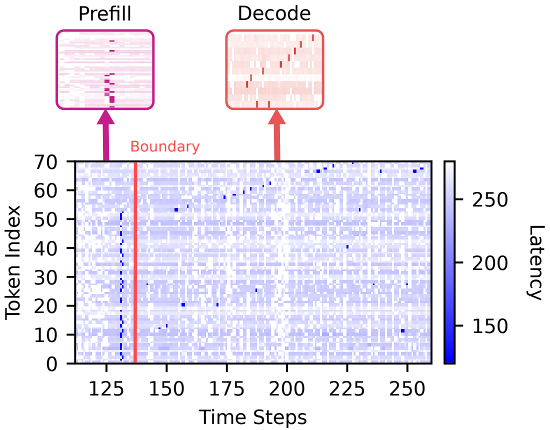
Step 1: Obtaining Cache Trace

- Use `mmap()` on the model file (exploiting the zero-copy model loaders and OS page cache)
- Calculate embedding table row addresses via model file format
- Overcome hardware prefetchers, especially the Array-of-Pointers (AoP) prefetcher
- Probe cache trace via multi-thread Flush+Reload

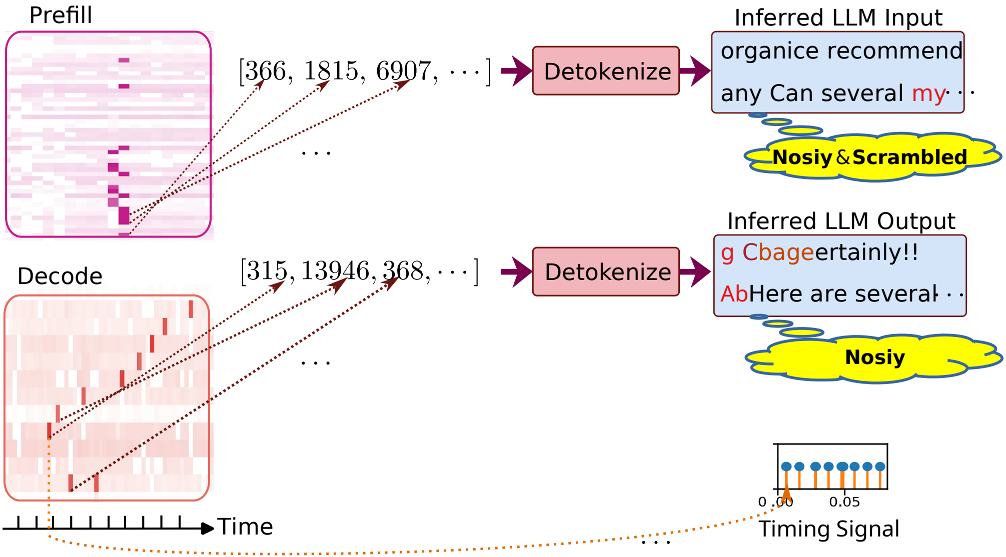


Step 2: Identifying Prefill and Decode

- The prefill stage has higher cache hit “density” than the decode stages

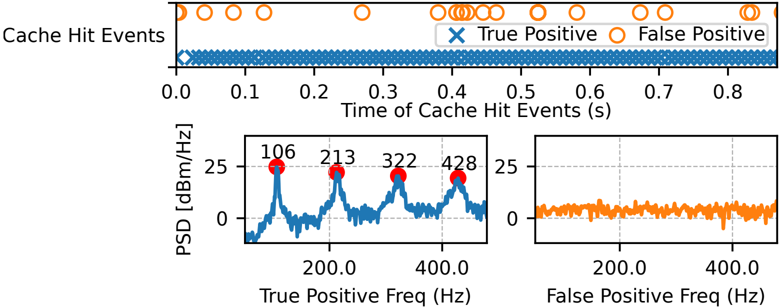


Step 3: Extracting Token List and Timing Signal



Let's Take a Deeper Look

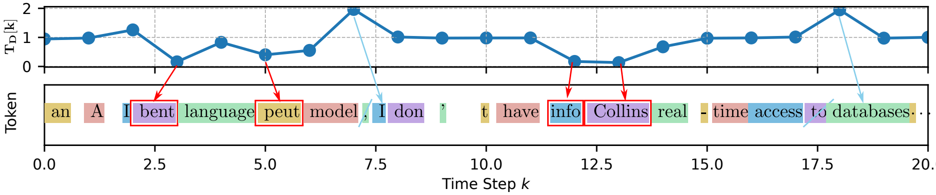
- **Problem:** Cache trace is noisy
- **Analysis:** Characterize the trace during decoding using power spectral density (PSD)



We can differentiate between false positives and true positives!

Characterizing Noise in Cache Trace

- To identify remaining false negatives, we excluded true positives by applying a PSD-based first-order temporal difference, yielding:

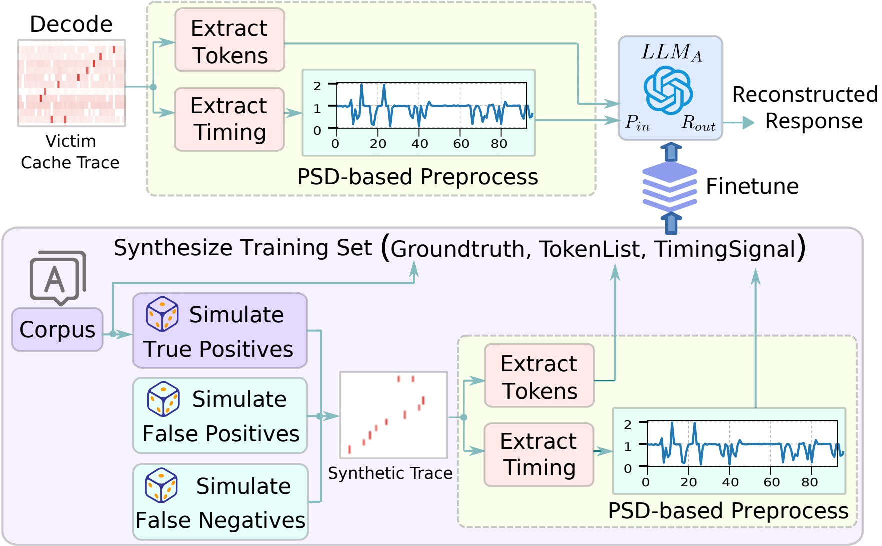


- We found that false negatives (FN) and false positives (FP) are also predictable
- Results are agnostic to hardware-specific decoding speed

Handling noise:

- Reducing false positives $\Longleftrightarrow$ Predicting and removing abnormal tokens near the valley
- Reducing false negatives $\Longleftrightarrow$ Predicting missing tokens near the peak

Reconstructing LLM Response



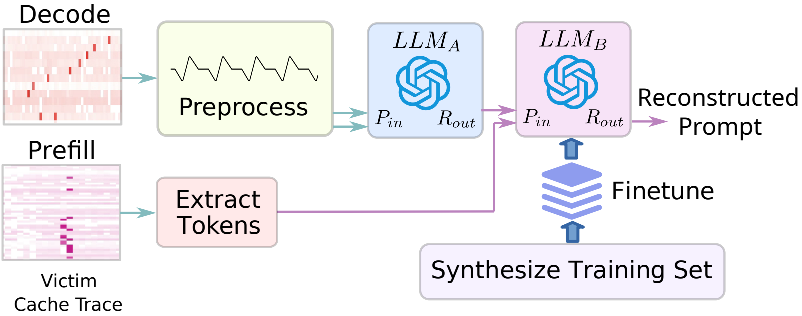
Reconstructing LLM Prompt

Problem: Scrambled token positions in the prefill stage

- **Root Cause:** Embedding operations run in parallel during prefill

Our Solution:

- Leveraging contextual dependency between LLM prompt and response
- Fusing prefill tokens



Experimental Setup

Real-world Deployment

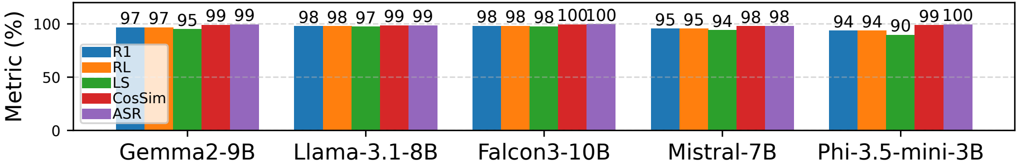
- **Various Hardware:**
  - Intel i9 14th/13th Gen (Raptor Lake)
  - Intel i7 12th Gen (Alder Lake)
  - With (Without) NVIDIA RTX 3060 GPU
- **5 LLMs:**
  - Google Gemma2, Meta Llama3.1, TII Falcon3, Mistral, Microsoft Phi3.5
- **10 LLM Inference Frameworks:**
  - HuggingFace Transformers, LM Studio, Llama.cpp, etc.

Evaluation

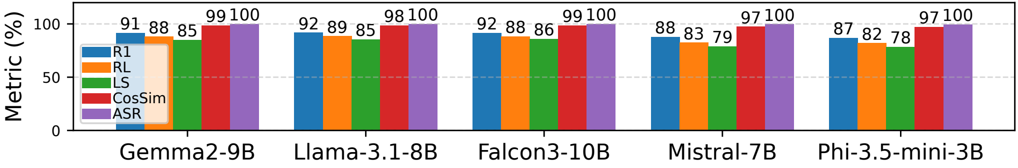
- **Constructed Datasets:**
  - Sources: 5 LLM benchmark datasets (UltraChat, NQ-Open...)
  - Total: $212,535$ prompt tokens after random sampling
  - Partition: 60% training corpus, 20% validation, 20% testing, with data cleaning

Attack Performance Across Models

- Highly accurate for response and prompt reconstruction
- Largely agnostic to LLM type (tested on 5 model families)



Average Response Reconstruction Performance†



Average Prompt Reconstruction Performance†

†Evaluated on llama.cpp with GPU acceleration

Generalization Across LLM Inference Frameworks

- Succeeded on **10** popular LLM inference frameworks[†]
- No need to retrain the attacker model
- The attack is fairly agnostic across different LLM inference frameworks

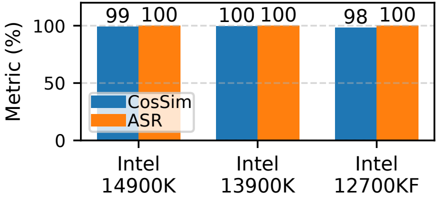
Framework	Github Stars[‡]	CPU		GPU	
		ϕ_O	ϕ_I	ϕ_O	ϕ_I
LM Studio	N/A	96.6	97.0	97.4	97.3
HuggingFace Transformers	138k	98.0	74.5	N/A	N/A
Ollama	108k	92.0	95.7	99.7	96.1
llama.cpp	71k	99.5	95.2	99.2	97.8
GPT4All	71k	97.6	95.6	98.9	94.3
LocalAI	28k	99.1	97.6	99.0	96.4
Microsoft BitNet	12k	96.1	76.0	98.3	74.5
PowerInfer	8k	98.0	96.5	98.5	96.2
Intel IPEX-LLM	7k	88.6	93.8	96.6	96.1
koboldcpp	6k	97.6	94.9	99.1	95.5

[†]Validated on the microbenchmark with 20 random samples

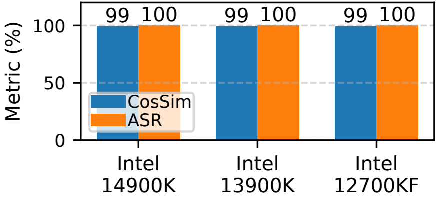
[‡]Cut-off date: January 21, 2025

Generalization Across Hardware

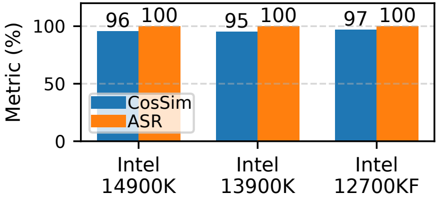
- **Hardware Agnostic:** Succeeded on several hardware configurations (microbenchmark)



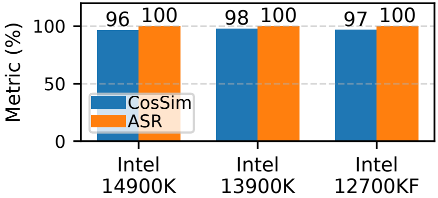
Restoring Response (CPU Victim)



Restoring Response (GPU-Accelerated Victim)



Restoring Prompt (CPU Victim)



Restoring Prompt (GPU-Accelerated Victim)

Attack Example

Observations:

- Recovered **unique n-grams**: "freddy krueger" and "e5"
- Potential to leak **PII** (Personally Identifiable Information)

Attacks on Prompts

ϕ : 100% R1: 100% LS: 100%

who played freddy krueger in the 2010 nightmare on elm street?

who played freddy krueger in the 2010 nightmare on elm street?

ϕ : 98% R1: 96% LS: 87%

How can I manage my weight and **avoid gaining** excess body fat?

How can I manage my weight and **avoid** excess body fat?

ϕ : 87% R1: 78% LS: 28%

what rank is an e5 in the air force?

an e5 in the air force is **what rank**?

Mitigation and Future Work

Hardware Mitigations

- Cache partitioning (Intel CAT)
  - However, CAT is typically unavailable on consumer-grade CPUs

Attack Limitations

- Cache side-channel is noisy
- Requires shared memory

Software Mitigations

- Disable zero-copy loading
  - However, it incurs memory overhead
- Role-based access control
  - Requires OS support in practice

Future Work

- Explore additional CPU side channels (e.g., Prime+Probe)
- Extend the attack to GPU side channels (e.g., Invalidate+Reload) targeting GPU token embedding

Key Takeaways

- We present the first cache side-channel attack framework capable of successfully recovering LLM prompts and responses
- Our study demonstrates tangible threats to on-device LLM privacy
- The mitigation calls for coordinated hardware/software methodologies
- Privacy assurances should span the full system stack

Thank you for your attention!

Check our website for more details!