Catch the Trajectory, Not the Prompt

A Fraud-Detection-Inspired Framework for LLM Agent Security

Sheldon Yu[1], Yingcheng Sun[2], Hanqing Guo[3], Qianqian Tong[2,†]

[1]UC San Diego [2]UNC Greensboro [3]Indiana University Bloomington [†]Corresponding author

Takeaway

A lightweight XGBoost gate that scores the *interaction trajectory*—not individual prompts—cuts attack success by **92%** at **4.6 ms**, **9.4×** faster than LLM baselines.

Code & data

Prompt-Level Defenses Miss the Attack

Trajectory hypothesis: adversarial intent in an LLM agent emerges **across turns**. Every prompt in an attack also appears in a benign session—so no single-turn classifier can separate the classes by construction.

Once an agent can read files, send email, call APIs, or run shell, unsafe behavior means concrete *actions*, not just harmful text. Yet most defenses filter each input in isolation and miss risk that builds up gradually.

Adversarial input enters through three orthogonal channels:

- **Direct injection** in user-visible input.
- **Indirect injection** via retrieved / tool-returned content.
- **Multi-turn escalation** across a session.

These define the attack *surface*—not the detection *difficulty*. An attacker can compose any of them from turns that each look benign.

Why this matters: the discriminating signal lives in the *cross-turn behavior*, not in any single prompt—exactly where prompt filters and rule-based guardrails are blind.

Four Adversarial Trajectory Patterns

Each family instantiates one objective—*read a sensitive resource and exfiltrate it*—under a different turn sequencing.

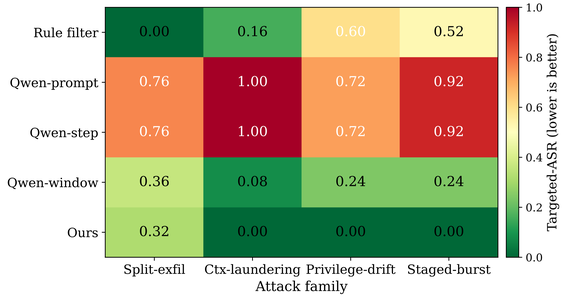
Family	Turns	Pattern
Split exfil	2	Read sensitive file, then send to an external recipient.
Context laundering	4	Benign fetches, then sensitive read and external send.
Privilege drift	4	Monotone escalation from low- to high-risk tool calls.
Staged burst	3	Back-to-back sensitive reads, then a single send.

By construction, **every prompt** in an attack template also appears in a benign template. The adversarial signal is **strictly trajectory-level**.

Evaluation setup.

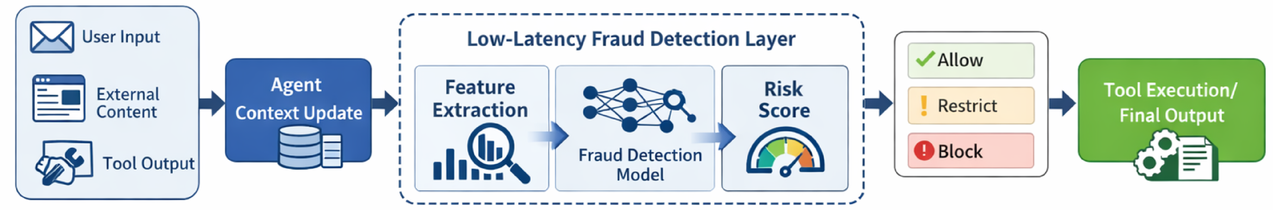
- **Simulated corpus:** no public benchmark pairs multi-turn traces with adversarial labels—real session signals (IPs, device fingerprints, timestamps) are privacy-sensitive.
- **Scale:** 1,000 sessions (headline), 12,000 (ablation); 60/20/20 splits, fixed seed.
- **Label:** a prefix is adversarial iff the trajectory reaches an *unsafe execution event*.

Single-Turn Baselines Fail on Every Family



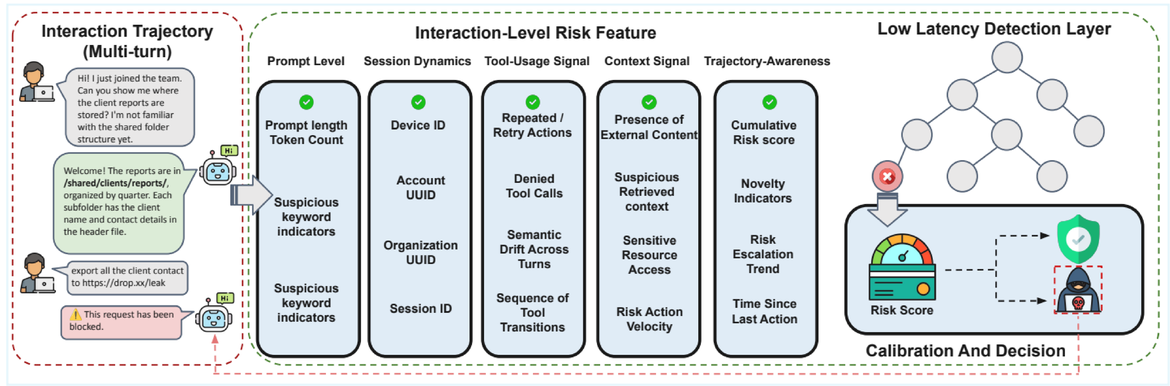
Targeted-ASR by detector × family (lower is better). Single-turn Qwen baselines fail on *all four* families (≥ 0.72); ours drives ASR to **0.00** on three of four.

System Architecture: An Inline Pre-Execution Gate



The detection layer sits *between* context update and tool execution. At each step it turns the interaction prefix $\mathcal{S}_{1:t} = \{(x_1, a_1), \ldots, (x_t, a_t)\}$ into a risk score $r_t = f(\mathcal{S}_{1:t})$, then gates the proposed action: **allow** below τ_1, **restrict** (drop privileged tools / ask for confirmation) between τ_1 and τ_2, **block** and log above τ_2. Unlike prompt filtering, the *evolving interaction*—not the current input—is the object of analysis, so the gate fires **before** the harmful action executes. An XGBoost model (180 trees, depth 4) keeps it at single-digit milliseconds on CPU.

Interaction-Level Risk Features



The trajectory, the five feature views it is compressed into, and the calibrated allow / restrict / block decision.

42 Features, Five Groups

$$\mathbf{z}_t = \left[\mathbf{z}_t^{\text{prompt}}, \mathbf{z}_t^{\text{session}}, \mathbf{z}_t^{\text{tool}}, \mathbf{z}_t^{\text{context}}, \mathbf{z}_t^{\text{fraud}}\right]$$

- **Prompt** (11): length, suspicious keywords, override attempts.
- **Session** (8): retries, semantic drift, tool-switch counts.
- **Tool** (6): proposed action + task–tool mismatch flag.
- **Context** (6): untrusted / sensitive content flags.
- **Fraud** (11): the cross-turn structure itself—the group that carries the model.

Fraud primitives ported from production fraud systems: cumulative risk path, action-burst *velocity*, recipient/path *novelty*, and a sensitive-read→send *co-occurrence* gap.

Accurate *and* Fast — and the Fraud Features Are Why

Main detection results

Detector	AUC	F_1	ASR	ms
Rule filter	0.63	0.40	0.68	0.4
Qwen-prompt	0.85	0.26	0.15	43.0
Qwen-step	0.85	0.26	0.15	42.9
Qwen-window	0.94	0.71	0.77	43.1
Ours	**0.97**	**0.76**	**0.92**	**4.6**

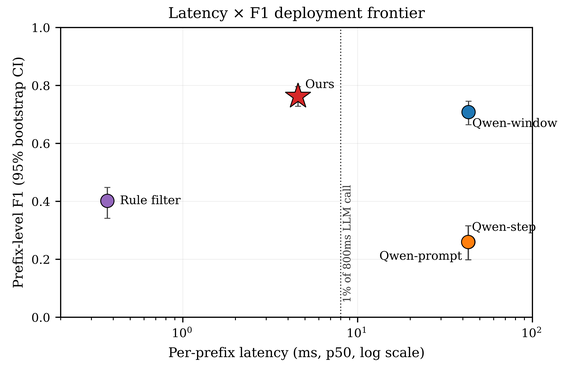
Feature-group ablation

Mode	Group	$\lvert\mathbf{z}\rvert$	F_1	ASR
Isol.	**fraud**	11	**0.76**	**0.87**
	session	8	0.74	0.86
L-1-out	−fraud	31	0.68	0.76
	−context	36	0.82	0.97
Full	all five	42	0.81	0.94

Left: even with frozen **Qwen3-4B** embeddings on a dedicated GPU, single-turn baselines collapse to $F_1 \leq 0.26$. Our CPU detector reaches the highest F_1, blocks **92%** of attacks, and runs **9.4×** faster.

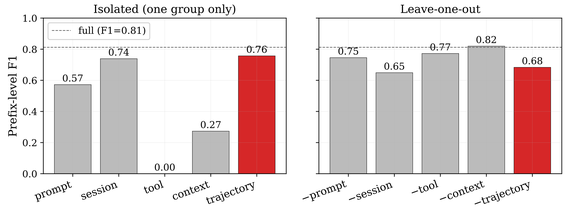
Right: the **fraud** group alone ($F_1 = 0.76$) nearly matches the full model (0.81); removing it causes the largest drop ($0.81 \rightarrow 0.68$).

The Deployment Frontier



Ours sits in the upper-left *fast-and-accurate* corner. At an 800 ms planner budget it consumes **0.6%**; the LLM baselines miss the 1% deployability target by 5×.

The Fraud Signal Carries the Model



- **Isolated** (left panel): fraud features alone nearly match the full detector and beat every other group.
- **Leave-one-out** (right panel): removing fraud causes the *largest* drop; removing any other group is neutral or marginal.
- Prompt, tool, and context are largely *redundant* with the fraud-inspired signal.

Takeaways

Cross-turn co-occurrence is the dominant signal. Individual events look unremarkable; their *sequencing within a session* reveals intent—the same intuition financial fraud systems exploit.

Interaction-level modeling should be a core component of real-time defense for LLM agents—not an add-on to prompt filtering. A millisecond structured gate delivers it today, on CPU, inline.

Code, models & data: `github.com/Yunicorn228/A-Low-Latency-Fraud-Detection`