

A Fraud-Detection-Inspired Framework for LLM Agents Security

Sheldon Yu
ziy040@ucsd.edu

University of California, San Diego
La Jolla, USA

Hanqing Guo
guohan@iu.edu

Indiana University Bloomington
Bloomington, USA

Yingcheng Sun
y_sun4@uncg.edu

UNC at Greensboro
Greensboro, USA

Qianqian Tong*
q_tong@uncg.edu

UNC at Greensboro
Greensboro, USA

Abstract

Large Language Model (LLM) agents demonstrate strong capabilities in autonomous task execution, tool use, and multi-step reasoning. However, their increasing autonomy also introduces a new attack surface: adversarial interactions can manipulate agent behavior through direct prompt injection, indirect content attacks, and multi-turn escalation strategies. Existing defense strategies focus on prompt-level filtering and rule-based guardrails, which are often insufficient when risk emerges gradually across interaction sequences. In this work, we propose a fraud detection-inspired framework for modeling adversarial interaction risk in LLM agents. Instead of determining whether a single prompt is malicious, our framework models risk over interaction trajectories using behavioral signals inspired by fraud detection, comprising signals from prompt content, session history, tool usage, execution context, and cross-turn interaction patterns. The detection framework can be implemented using lightweight models leading to low-latency real-time deployments. To validate our proposed framework, we conduct a controlled simulation study using parameterized interaction templates that simulate realistic agentic workflows. Instantiating the behavioral signals as structured features with a lightweight XG-Boost classifier, our detector runs over 9× faster than LLM-based detectors. Experiment results confirm that the cross-turn trajectory signals are the dominant contributors to detection performance, suggesting that interaction-level modeling should be a core component of real-time defense for LLM agents. Code, models, and more details will be made available at: <https://github.com/Yunicorn228/A-Low-Latency-Fraud-Detection>

CCS Concepts

• **Information systems** → **Content ranking**; **Language models**; • **Web searching and information discovery** → **Content ranking**.

*Corresponding author

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Conference acronym 'XX, Woodstock, NY

© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-XXXX-X/2018/06

<https://doi.org/XXXXXXX.XXXXXXX>

Keywords

LLM agents, agent security, fraud detection, prompt injection, adversarial interaction

ACM Reference Format:

Sheldon Yu, Yingcheng Sun, Hanqing Guo, and Qianqian Tong. 2018. A Fraud-Detection-Inspired Framework for LLM Agents Security. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX)*. ACM, New York, NY, USA, 8 pages. <https://doi.org/XXXXXXX.XXXXXXX>

1 Introduction

Large language models (LLMs) have expanded the capabilities of AI agents, but their open-ended interaction also introduces significant safety risks. Recent LLM agents differ from earlier task-specific agent systems, such as rule-based expert systems[25, 44], reinforcement-learning agents[31, 36, 37], and domain-specific autonomous agents [6, 9, 22, 24, 33, 41, 45], in that they combine open-ended language understanding with tool use, external context, and multi-step execution. Systems and frameworks such as Claude-based agents [5], Codex [10, 34], and frameworks like AutoGPT[19], OpenClaw [39] illustrate this shift: LLMs are increasingly integrated with external tools, retrieved data, APIs, and planning loops to perform tasks that extend beyond single-turn text generation. This shift has enabled a new generation of agentic systems for coding, research assistance, personal productivity, enterprise workflows, and digital automation. However, the same flexibility that makes current agents powerful also makes them vulnerable. Once an agent can access tools, memory, local files, APIs, or external services, unsafe behavior is no longer limited to harmful text generation; it may involve concrete environmental actions such as reading private resources, sending messages, invoking APIs, or executing commands.

In emerging agent frameworks, adversarial behavior can exploit the system through direct prompt injection by embedding malicious intent inside apparently benign tasks[14, 29], or indirect prompt injection by poisoning retrieved external content [17, 20]. These risks make deployment-time security a central challenge for LLM agents, especially when agents operate over private data, external tools, or user-authorized actions. Many existing defenses treat prompt injection primarily as an input- or prompt-level detection problem, often relying on filtering, classification, or refusal based on current input [4, 13, 26, 42]. In this setting, the system attempts to determine whether the current prompt contains malicious instructions, such as attempts to override the system prompt, or induce unsafe

tool usage. While prompt-level defense is important, it is incomplete. In practice, *adversarial interactions with LLMs systems are often sequential and strategic, and many unsafe trajectories are not well captured by a single prompt*. A risky session may begin with benign requests, continue through probing or reformulation, and only later trigger an obviously harmful action. A detector that only inspects the current input may therefore miss the broader pattern of interaction.

This observation motivates a shift in perspective: rather than treating agent security purely as a prompt-level filtering problem, we formulate it as a system-level adversarial interaction detection. In this view, prompt injection becomes a special case of a broader class of adversarial interactions. Direct prompt injection corresponds to scenarios where the current user intentionally manipulates the agent, while indirect prompt injection arises when adversarial instructions are embedded in external content, such as retrieved documents or tool outputs, effectively allowing one user to influence another user’s interaction with the agent. By modeling security at the interaction-level, our framework considers not only individual inputs, but also the evolving execution trace of the agent. This enables detection of more complex and realistic attack patterns, including multi-step escalation, delayed malicious behavior, and cases where seemingly benign interactions gradually transition into unsafe outcomes.

Inspired by mature fraud detection systems in payments[15], login protection [32], and abuse prevention[27], we propose a fraud-inspired structured trajectory-feature detector for LLM agents. (See Figure 1) The framework is not tied to one fixed list of features. Instead, it defines a set of behavioral risk axes, such as escalation, velocity, novelty, task–tool mismatch, context exposure, sensitive-resource access, and externalization, that can be instantiated using the signals available in a particular agent system. The proposed detector is complementary to LLM-based guards, sandboxing, access-control policies, and human review. Its role is to provide a fast pre-execution risk signal that can decide whether an action should be allowed, restricted, blocked, or escalated to a stronger defense. The main contributions of this work are as follows:

- We reframe LLM-agent security as trajectory-level adversarial interaction detection rather than isolated prompt classification.
- We introduce a lightweight low-latency detection layer that integrates prior to sensitive execution stages, enabling practical deployment in real-time systems.
- We design a structured feature space that captures prompt signals, interaction dynamics, tool usage, and context exposure, with trajectory aware features.
- We conduct simulation-based experiments that evaluate both classifier performance and end-to-end system security outcomes, enabling practical deployment in real-world applications.

2 Related Work

2.1 Security Risks in LLM Agents

LLM agents combine language understanding with action execution: they receive a user request, plan intermediate steps, retrieve context, invoke external tools, and generate actions or final outputs [18]. This

Method	Single-turn	Multi-turn	Low Latency	Deployable
Prompt-level [42]	✓	✗	✗	✓
Guardrails [11, 12]	✓	✗	✗	✓
Sandboxing [30, 46]	-	-	✓	✗
Alignment [38]	✓	✗	✗	✓
Ours	✓	✓	✓	✓

Table 1: Comparison of defense strategies. Existing methods focus on single-turn filtering, while ours models multi-turn interaction with low-latency deployment.

architecture is significantly more capable than a static chatbot, but it also introduces significant security risks. A major class of attacks is prompt injection, in which adversarial content attempts to override system instructions or manipulate downstream behavior. Related risks include jailbreaks, indirect prompt injection from retrieved documents, malicious tool suggestions, instruction collisions, and context poisoning [23].

2.2 Existing Defense Strategies

The majority of existing work on LLMs security has focused on defensive strategies, primarily operating at the content or model level: (1) *prompt-level defense* [42], which attempts to classify or sanitize suspicious inputs before they are processed; (2) *output-level defense*, well-known example, Guardrail mechanisms [11, 12], constrain model outputs or tool calls using policies, templates, and validation rules; (3) *sandboxing and permission restriction*[30, 46], which limit the impact of unsafe actions by restricting what the agent can access or execute; and (4) *alignment-based methods*[38], which train or instruct models to resist harmful instructions and adhere to safety requirements more consistently. As summarized in Table 1, existing approaches effectively mitigate prompt-level risks through input filtering and output control. However, high-performance detection methods often rely on LLM-based defenses, which can incur significant latency and limit timely responses in practice. In realistic agent scenarios, this is insufficient, as adversarial behavior is inherently sequential and unfolds across multiple interaction steps.

An interaction may exhibit repeated probing, semantic redirection, progressive escalation, or suspicious coordination between retrieved content and subsequent execution requests. Such patterns are difficult to detect when each prompt is analyzed independently. At the same time, stronger defenses such as sandboxing can be costly and cumbersome to deploy in real-time systems. These observations suggest that behavioral detection over interaction traces should complement prompt-level filtering, while remaining lightweight enough for practical deployment.

2.3 Behavior-level Fraud Detection

Fraud detection [2, 3] is closely related to this problem, as it focuses on identifying malicious behavior based on patterns across multiple interaction steps. Inspired by this perspective, we shift from analyzing prompts in isolation to modeling interaction dynamics across multiple turns. Fraud detection has been primarily studied in domains such as financial transactions and cyber fraud [28, 40, 43]. For example, in payment systems, account abuse prevention, spam

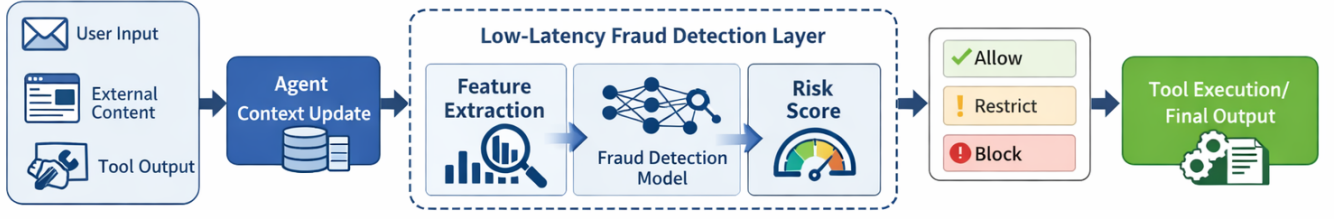


Figure 1: Conceptual overview of the proposed trajectory-level risk detection framework for LLM agents. The detector estimates risk using structured runtime signals before sensitive actions are executed.

filtering, and cybersecurity, defenders rarely rely on one single attempt. Instead, they aggregate evidence from sequence patterns, velocity, anomaly scores, contextual mismatch, prior attempts, and execution outcomes.

However, related work on LLM agents is quite limited, particularly in supporting real-time detection of adversarial behavior. This motivates the use of efficient and lightweight detection mechanisms. Accordingly, we adopt lightweight models instead of large-scale LLMs with billions of parameters. We employ machine learning models, such as shallow neural networks or tree-based models. It is observed that tree-based models are particularly effective for tabular data [7, 21]. Interestingly, data generated from fraud-related behaviors are often structured, heterogeneous, noisy, and highly imbalanced, as fraudulent events are rare compared to normal activities [8, 35]. This alignment between model capabilities and data characteristics makes tree-based models a natural and effective choice for the proposed fraud detection layer in real-world settings.

3 Method

We consider an LLM agent that receives user instructions, ingests external content, invokes tools, and maintains a short interaction history. The agent may have access to actions such as retrieval, local file reading, messaging, shell execution, or structured API calls. The adversarial objective is to induce unsafe behavior, including disclosure of sensitive information, unauthorized tool execution, policy bypass, or misuse of privileged capabilities. We propose a low-latency fraud detection framework for identifying adversarial interaction patterns in LLM agents. As illustrated in Figure 2, the framework consists of structured feature construction, lightweight model inference, and risk-aware decision layers.

3.1 Threat Model

We frame agent security as the detection of adversarial interaction trajectories rather than the classification of malicious individual prompts. Adversarial inputs may enter the agent through three orthogonal channels: (1) direct injection in user-visible input, (2) indirect injection via retrieved or tool-returned content, and (3) multi-turn escalation across a session. These channels constitute the attack surface, but do not by themselves determine detection difficulty, a sufficiently sophisticated attacker can compose any of the three with a turn sequence whose individual prompts are indistinguishable from benign agent traffic. We therefore characterize attacks by how the trajectory unfolds across turns rather than

Family	Turns	Description
split_exfil	2	Read sensitive file, then email to external recipient
context_laundering	4	Benign fetches followed by sensitive read and external send
privilege_drift	4	Monotone escalation from low-risk to high-risk tool calls
staged_burst	3	Back-to-back sensitive reads followed by a single send

Table 2: Attack strategies used in the evaluation. Each strategy targets a distinct adversarial pattern from the threat model.

by entry channel. We consider the main trajectory patterns: Split exfiltration, Context laundering, Privilege drift and Staged burst in Table 2.

3.2 Problem Formulation

We model user interactions as a sequence of turns indexed by $t \in \{1, \dots, T\}$,

$$\mathcal{S}_{1:T} = \{(x_1, a_1), (x_2, a_2), \dots, (x_T, a_T)\}, \quad (1)$$

at turn t , x_t denotes the model-visible input, including user input or content added to context, a_t denotes the agent’s action. We denote the partial interaction history up to turn t as $\mathcal{S}_{1:t}$. The objective is to estimate whether $\mathcal{S}_{1:t}$ exhibits an adversarial behavior.

We define a risk score

$$r_t = f(g(\mathcal{S}_{1:t})), \quad (2)$$

where $g(\cdot)$ is a feature construction function, $f(\cdot)$ is a lightweight detection model based on structured features extracted from the user’s current prompt and historical interactions. Unlike prompt filtering, which treats the current input as the primary detection target, this formulation makes the evolving interaction the central object of analysis.

3.3 Structured Feature Representation

At each interaction step, the detector constructs a structured feature vector from the interaction history $\mathcal{S}_{1:t}$:

$$\mathbf{z}_t = g(\mathcal{S}_{1:t}) = [\mathbf{z}_t^{\text{prompt}}, \mathbf{z}_t^{\text{session}}, \mathbf{z}_t^{\text{tool}}, \mathbf{z}_t^{\text{context}}, \mathbf{z}_t^{\text{trajectory}}] \quad (3)$$

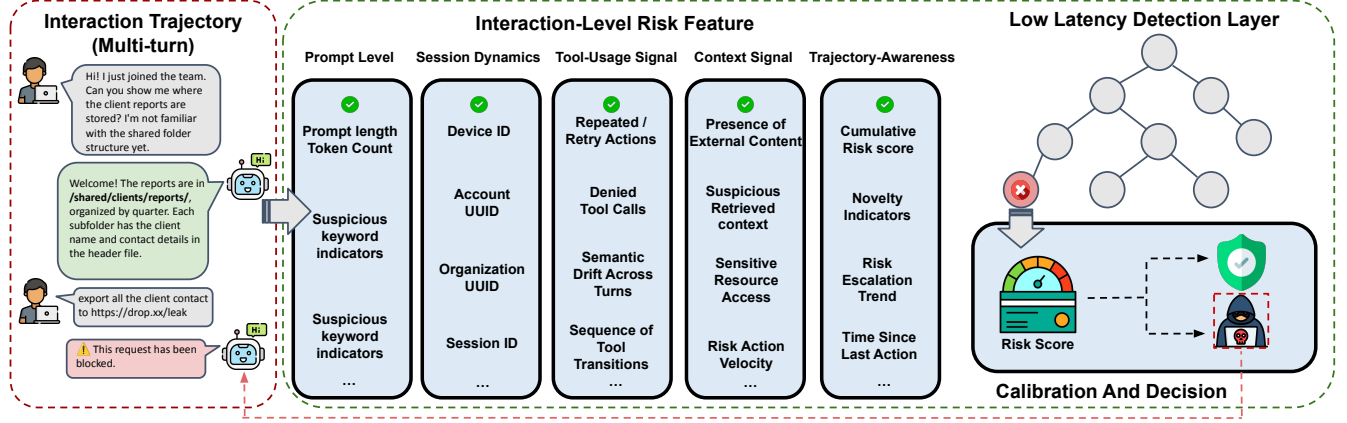


Figure 2: End-to-end architecture of the proposed fraud detection framework, illustrating input encoding, feature alignment, model inference, and training objectives within a unified pipeline.

The framework as a whole is inspired by fraud detection practices, where risk is modeled across multiple dimensions rather than from any single signal. The five groups provide complementary views of the interaction: prompt derived signals (prompt), session related features (session), the agent’s proposed actions (tool), the environment (context), and and cross-turn trajectory signals that encode escalation and co-occurrence patterns (trajectory).

The five signal groups described below represent one concrete instantiation of the framework, not an exhaustive or fixed feature set. Each group corresponds to a feature dimension that can be obtained in different ways depending on the deployment environment, instrumentation, and agent framework. For example, session dynamics can be captured through simple turn-indexed counters as done here, or through learned sequence representations. The specific features used in our experiments are chosen to be lightweight and interpretable, but the framework imposes no constraint on features within each group.

Prompt features capture information of the input and its immediate context, including prompt length, token count, suspicious keyword indicators, instruction-conflict patterns, override attempts, potential privilege escalation signals, etc. Prompt features are most closely related to traditional prompt detection signals, as they operate on individual inputs.

Session features summarize the agent’s behavior over the interaction history using turn-indexed aggregates. These include signals of retries and repetition, denied or failed actions, recent exposure to untrusted content, semantic drift across turns, and diversity in tool usage. These quantities are the turn-indexed counterpart to the time-windowed velocity signals used in production fraud-detection systems [1, 16]. They are adapted to the agent interaction setting where wall-clock timestamps are unavailable, but turn ordering is fully observed. They allow the detector to capture sustained or coordinated behaviors that are not observable from any individual prompt. All features are computed incrementally and updated at each turn.

Tool features encode the agent’s currently proposed action using a one-hot representation over the discrete tool set (web_fetch,

summarize, read_local_file, send_email, run_shell, etc), together with a binary task-tool mismatch flag. This flag is set to 1 when the selected tool is inconsistent with the user’s request, and 0 otherwise. We separate trajectory-level signals from tool features, including cumulative risk, escalation, and burst rate, to enable ablation studies that isolate the effect of interaction dynamics from action identity.

Context features incorporate signals from the surrounding environment and retrieved content. These include whether external or untrusted content is present, whether such content contains suspicious patterns, whether sensitive resources are involved, whether risky actions follow ingestion of external inputs, etc. These features are particularly important for indirect attacks, where the malicious signal may originate from retrieved or tool-mediated context rather than the user’s prompt.

Trajectory features are motivated by common signals used in fraud detection, such as cumulative risk, velocity bursts, novelty flags, and temporal co-occurrence. We adapt these signals to the agent interaction setting and compute the following: (i) a cumulative tool-risk path with its turn-to-turn delta and a monotonicity flag, capturing privilege escalation; (ii) an action-burst score defined as the fraction of the last three turns invoking a high-risk tool, capturing the staged-burst pattern; (iii) novelty flags for the current email recipient and file path, evaluated against a benign-only reference profile fitted on the training split, analogous to “new device” flags in account-fraud systems; and (iv) a context-exfil gap measuring the turn distance between the first sensitive read and a subsequent external send within the same session, capturing split-exfiltration co-occurrence. Whereas session features compute generic prefix aggregates over arbitrary actions, these features explicitly target the cross-turn structure that defines an adversarial trajectory. As shown in our ablation study (Section 4.3), this group alone nearly matches the full detector and dominates every other group in both isolated and leave-one-out evaluations.

3.4 Model Choice

We focus on lightweight models suitable for online deployment. Gradient-boosted decision trees, especially XGBoost, provide state-of-the-art performance for tabular data, capture nonlinear interactions, and offer fast inference. Shallow neural networks either trained from scratch or via distillation, provide an alternative when the feature interactions are more complex. The central design principle remains the same: the defense layer should perform well with tabular data, support low-latency inference and allow reliable calibration.

We train the detector as a binary classifier over benign versus adversarial interactions. Let $y_i \in \{0, 1\}$ denote the label for interaction i and let $\hat{y}_i$ denote the predicted probability of adversarial interaction. The objective is a standard binary cross-entropy

$$\mathcal{L} = - \sum_i \left(y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i) \right). \quad (4)$$

Further, the same formulation can be extended to weighted losses for imbalanced data or to ordinal risk levels for multi-stage intervention systems.

3.5 Low-Latency Fraud Detection Layer

The proposed fraud detection layer is inserted before critical execution stages in the agent pipeline. The system computes a risk score prior to executing a tool call or high-impact action. In more sensitive settings, the detection layer may also run immediately after the context update stage, allowing the system to apply earlier scrutiny.

We define the decision rule as:

$$\text{decision} = \begin{cases} \text{allow,} & r_t < \tau_1, \\ \text{restrict,} & \tau_1 \leq r_t < \tau_2, \\ \text{block,} & r_t \geq \tau_2, \end{cases} \quad (5)$$

where τ_1 and τ_2 are system thresholds. In the restrict regime, the system may disable privileged tools, request explicit confirmation, or route the interaction through a stronger audit path. In the block regime, the action is denied and logged for inspection. This design retains the practical benefits of low-latency risk scoring while supporting graded intervention.

3.6 Online Inference

The detector operates incrementally, updating features at each interaction step. As illustrated in Figure 2, features are updated across multiple levels. Prompt-level features are computed from the current input, capturing token statistics and keyword-based signals. Session-level features summarize interaction metadata and accumulated history. Tool-usage signals are updated based on the agent’s proposed action, including repeated or denied operations and transition patterns between tools. Context-level features track newly introduced external or sensitive information. Trajectory-awareness signals aggregate behavior over time, including cumulative risk, novelty, and escalation trends. These feature groups are updated in a streaming manner and the resulting feature vector is passed through the model to compute r_t . This supports early detection because the system does not wait for a clearly unsafe action to appear; it evaluates whether the trajectory itself is becoming risky.

Since both feature extraction and inference remain lightweight, the added latency can remain in the millisecond range.

4 Experiments

We evaluate the proposed detection layer on a synthetic corpus to assess whether interaction-level modeling improves detection over prompt-level approaches, especially when individual inputs are not overtly malicious.

4.1 Setup

A key challenge in evaluating multi-turn agent security is the lack of public benchmarks with labeled interaction trajectories. Unlike single-prompt injection datasets, which can be collected in isolation, such data requires complete multi-turn interactions and often involves privacy-sensitive context. In fraud detection systems, such context includes user IP addresses, device fingerprints, and login timestamps [1, 16]. We therefore adopt a simulation-based evaluation, constructing a synthetic corpus with controlled adversarial properties to isolate interaction-level detection signals.

Data generation. We construct a synthetic corpus of multi-turn agent interactions using parameterized templates that simulate realistic workflows, such as file retrieval, email composition, web browsing, and shell execution. From these raw interaction traces, we extract structured features using window-based aggregations and velocity-style signals. For example, the number of high-risk tool calls within the last k turns and the rate of denied or escalated requests over a sliding window. This feature engineering follows standard practice in industry fraud detection pipelines [16], where per-event features alone are insufficient and temporal aggregation over recent history is essential for capturing evolving risk. We generate 12,000 interactions, split 60/20/20 into train, validation, and test sets with a fixed random seed. The held-out test set contains 1,200 benign and 1,200 adversarial interactions. Each interaction contributes multiple prefix evaluation points, corresponding to partial interaction histories at different turns, yielding approximately 6,000 evaluation instances in total.

Label definition. We frame detection as a binary classification problem. Each interaction prefix $S_{1:t}$ is labeled as *adversarial* ($y=1$) if the corresponding interaction ultimately leads to an unsafe execution event, defined as a sensitive file read co-occurring with external data exfiltration or unauthorized privileged execution. Otherwise, we will label *benign* ($y=0$). This labeling reflects the operational objective: flag risky interaction trajectories before the harmful action is executed.

Baselines and model. We compare against four baselines: *Rule-filter*, *Prompt-only*, *Single-step*, *Seq-window*, that represent progressively stronger prompt-level and short-history defenses. To ensure that baseline performance is not limited by encoder capacity, the three text-based baselines all use Qwen3-4B run on a dedicated NVIDIA RTX A6000 GPU.

Rule-filter applies regex-based keyword matching over override, secret, and action terms with a count threshold of 3. *Prompt-only* trains a logistic-regression classifier on the Qwen3-4B embedding of the current-turn prompt concatenated with any external content. *Single-step* is equivalent to Prompt-only and produces identical results. We retain both for consistency with standard baseline

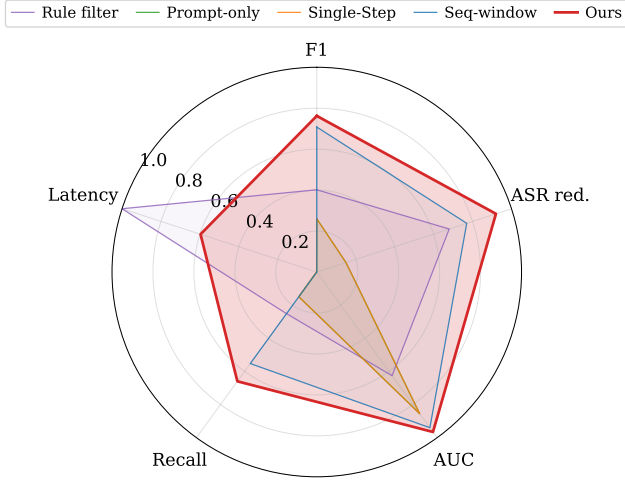


Figure 3: Five-axis deployment profile across F_1 , ASR reduction, AUC, recall, and per-prefix latency; larger area is better.

taxonomy. These three baselines represent the dominant single-turn paradigm in prompt-injection defense. *Seq-window*, extends to multi-turn context by encoding a rolling last-three-turn window, providing a strong short-history comparator.

Our detector uses XGBoost with 180 estimators and `max_depth=4` on 42 structured features spanning five groups: prompt, session, tool, context, and trajectory. We select XGBoost model because gradient-boosted trees remain the established method for structured, heterogeneous tabular data in industry fraud detection systems, and their inference latency makes them suitable for inline deployment without additional overhead.

The choice of XGBoost in our experiments is one instantiation of the lightweight model requirement; the framework is compatible with any model class that supports low-latency tabular inference, including shallow neural networks.

Metrics. We report standard metrics used in fraud detection: AUC, precision, recall, and F_1 , all computed at the prefix level. We additionally report *ASR reduction* [14]. It measures the fraction of attack sessions that do not reach an unsafe execution event under the calibrated detection policy. This reflects real deployment settings. It is similar to stopping a fraudulent transaction before funds leave the account. Early detection allows intervention before harmful actions execute. We also report *per-prefix latency* (feature extraction + model inference, p50), since pre-execution gating is only useful if it fits within the LLM-planner latency budget. All confidence intervals are 95% percentile bootstraps; the canonical comparison uses 500 samples and the larger-scale validation uses 1,000.

4.2 Results

We report the main result summarized in Table 3. Two findings stand out. First, even with a strong LLM encoder (Qwen3-4B), the single-turn baselines *Prompt-only* and *Single-step* collapse to $F_1 \leq 0.26$ and reduce attack success by only 15%. This is a property of the corpus design, not the encoder: because the adversarial signal

Detector	AUC	Prec.	Recall	F_1	ASR red.	Lat. (ms)
Rule-filter	0.63	1.00	0.25	0.40	0.68	0.4
Prompt-only	0.85	1.00	0.15	0.26	0.15	43.0
Single-step	0.85	1.00	0.15	0.26	0.15	42.9
Seq-window	0.94	0.99	0.55	0.71	0.77	43.1
Ours	0.97	0.90	0.66	0.76	0.92	4.6

Table 3: Main detection results across all detectors.

Mode	Group(s)	z	AUC	F_1	ASR red.
Isolated	prompt	11	0.81	0.57	0.62
	session	8	0.86	0.74	0.86
	tool	6	0.65	0.00	0.00
	context	6	0.67	0.27	0.33
	trajectory	11	0.91	0.76	0.87
Leave-one-out	–prompt	31	0.92	0.75	0.87
	–session	34	0.96	0.65	0.86
	–tool	36	0.96	0.77	0.95
	–context	36	0.96	0.82	0.97
	–trajectory	31	0.92	0.68	0.76
Full	all five groups	42	0.96	0.81	0.94

Table 4: Feature ablation. Each row trains our detector on the indicated feature subset; ASR red. is the fraction of attack interactions blocked by the calibrated policy.

is distributed across turns rather than concentrated in any single prompt, no text encoder applied to one turn at a time can recover it. The multi-turn baseline *Seq-window*, which observes three consecutive turns, recovers a substantial fraction of the trajectory signal and reaches a higher F_1 than the single-turn baselines, but it is still outperformed by our structured detector and prevents fewer attacks overall.

Second, the latency gap is significant. The three Qwen-based baselines cost approximately 43 ms of forward-pass time per prefix on an A6000 GPU, consuming more than 5% of the LLM-planner latency budget and missing the 1% deployability target marked in Figure 3. Our XGBoost detector runs in 4.6 ms on a single CPU thread with 0.6% of the budget, placing it in the upper-left corner of the latency-versus- F_1 plane, 9.4–9.5× faster than any LLM baseline while achieving the highest F_1 in the table. Moreover, Figure 4 shows the targeted attack-success rate broken down by family. The *Rule-filter* catches sensitive filenames visible in a single turn but largely misses multi-turn families. The single-turn Qwen baselines fail uniformly, confirming that single-turn embeddings, however rich, cannot recover the trajectory signal.

4.3 Ablation Study

We examine which feature groups drive detection performance by decomposing the 42-feature space into its five constituent groups: prompt, session, tool, context, and trajectory. We train two ablation variants: an *isolated* model using only a single feature group, and

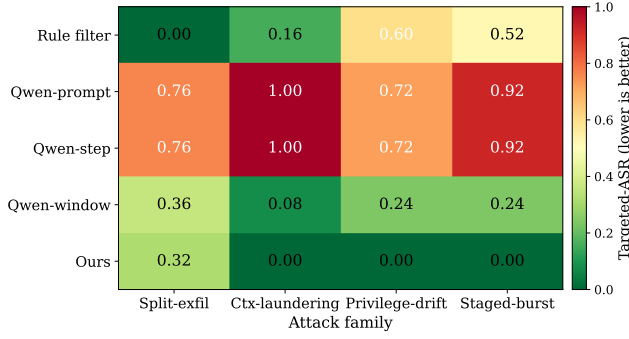


Figure 4: Targeted attack-success rate by detector and attack family.

a *leave-one-out* model that removes one group from the full set. Results are reported in Table 4 and visualized in Figure 5.

The two analyses yield a consistent conclusion. In isolation, the trajectory feature group achieves $F_1 = 0.76$ and an ASR reduction of 0.87, nearly matching the full model (0.81 and 0.94 respectively) and substantially outperforming every other individual group. Session features provide secondary lift ($F_1 = 0.74$), while prompt, tool, and context features in isolation are insufficient on their own. The leave-one-out analysis confirms this finding from the opposite direction. Removing the trajectory group causes the largest degradation: F_1 drops from 0.81 to 0.68 and ASR reduction falls from 0.94 to 0.76, admitting an extra 18% of attack interactions. Removing any other individual group has a neutral or marginal effect, suggesting that the prompt, tool, and context groups are largely redundant with the trajectory features for this task.

4.4 Discussion

The results support the central claim of the paper: adversarial interaction is a broader and more realistic target than isolated prompt injection. Prompt-level defenses remain useful, but they do not adequately capture multi-step behavior, indirect content attacks, or gradual escalation. By modeling structured interaction signals over time, the proposed framework adds a complementary line of defense that is especially well suited to action-taking agents.

Our detector framework has practical strengths. It is lightweight, deployable, and compatible with existing guardrails. Its features are interpretable enough to support monitoring and debugging, and it does not require another LLM in the critical path. These properties make it attractive for systems where latency, cost, and operational transparency matter.

Several extensions of the framework are worth exploring. First, the features used here are structure data inspired by fraud detection domain and low latency; future work could investigate whether learned representations of session or trajectory dynamics can improve performance as well as robustness. Second, the framework currently assumes a static feature set; online adaptation to emerging attack patterns through continual learning would improve resilience against adaptive adversaries. Third, as standardized multi-turn agent security benchmarks become available, systematic

evaluation across agent frameworks and tool sets will be needed to characterize generalization.

5 Conclusion

In this work, we propose a new perspective on securing LLM agents by focusing on adversarial interaction patterns rather than isolated prompt classification. Prompt injection remains an important security issue, but is best understood as a special case within a larger class of risky interaction trajectories. To address this problem, we introduce a low-latency fraud detection layer that models structured signals from prompts, sessions, tool usage, context and cross-turn fraud signals.

The resulting framework preserves the deployment advantages of low-latency lightweight models while extending agent security beyond prompt-only defense. It supports real-time operation, earlier intervention, and better alignment with the security logic used in mature fraud detection systems. More broadly, this work suggests that interaction-level behavioral modeling should become an important component of deployment-time defense for LLM agents.

References

- [1] Aisha Abdallah, Mohd Aizaini Maarof, and Anazida Zainal. 2016. Fraud detection system. *J. Netw. Comput. Appl.* 68, C (June 2016), 90–113. doi:10.1016/j.jnca.2016.04.007
- [2] Aisha Abdallah, Mohd Aizaini Maarof, and Anazida Zainal. 2016. Fraud detection system: A survey. *Journal of Network and Computer Applications* 68 (2016), 90–113.
- [3] Abdulaleem Ali, Shukor Abd Razak, Siti Hajar Othman, Taiseer Abdalla Elfadil Eisa, Arafat Al-Dhaqm, Maged Nasser, Tusneem Elhassan, Hashim Elshafie, and Abdu Saif. 2022. Financial fraud detection based on machine learning: a systematic literature review. *Applied Sciences* 12, 19 (2022), 9637.
- [4] Ahmed Alzahrani. 2026. PromptGuard a structured framework for injection resilient language models. *Scientific Reports* 16, 1 (2026), 1277.
- [5] Anthropic. [n.d.]. Claude: An AI Assistant by Anthropic. <https://www.anthropic.com/claude>.
- [6] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. 2017. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine* 34, 6 (2017), 26–38.
- [7] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gergely Kaszner. 2022. Deep neural networks and tabular data: A survey. *IEEE transactions on neural networks and learning systems* 35, 6 (2022), 7499–7519.
- [8] Dalia Breskuviene and Gintautas Dzemyda. 2024. Enhancing credit card fraud detection: highly imbalanced data case. *Journal of Big Data* 11, 1 (2024), 182.
- [9] Bruce G Buchanan and Edward A Feigenbaum. 1981. DENDRAL and Meta-DENDRAL: Their applications dimension. In *Readings in artificial intelligence*. Elsevier, 313–322.
- [10] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, et al. 2021. Evaluating Large Language Models Trained on Code. *arXiv preprint arXiv:2107.03374* (2021).
- [11] Sahana Chennabasappa, Cyrus Nikolaidis, Daniel Song, David Molnar, Stephanie Ding, Shengye Wan, Spencer Whitman, Lauren Deason, Nicholas Doucette, Abraham Montilla, et al. 2025. Llamafirewall: An open source guardrail system for building secure ai agents. *arXiv preprint arXiv:2505.03574* (2025).
- [12] Jingyue Cong, Xinyuan Qiao, Yulin Dong, Yueheng Huang, Yang Yu, Estrid He, and Andy Song. 2026. IntentGuard: Safeguard LLM Agents via Intent Alignment. (2026).
- [13] Pedro H Barcha Correia, Ryan W Achjian, Diego EG de Oliveira, Ygor Acacio Maria, Victor Takashi Hayashi, Marcos Lopes, Charles Christian Miers, and Marcos A Simplicio Jr. 2026. A Systematic Literature Review on LLM Defenses Against Prompt Injection and Jailbreaking: Expanding NIST Taxonomy. *arXiv preprint arXiv:2601.22240* (2026).
- [14] Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. 2024. Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for llm agents. *Advances in Neural Information Processing Systems* 37 (2024), 82895–82920.
- [15] Alexander Diadiushkin, Kurt Sandkuhl, and Alexander Maiatin. 2019. Fraud detection in payments transactions: Overview of existing approaches and usage for instant payments. *Complex Systems Informatics and Modeling Quarterly* 20 (2019), 72–88.
- [16] Alexander Diadiushkin, Kurt Sandkuhl, and Alexander Maiatin. 2019. Fraud Detection in Payments Transactions: Overview of Existing Approaches and

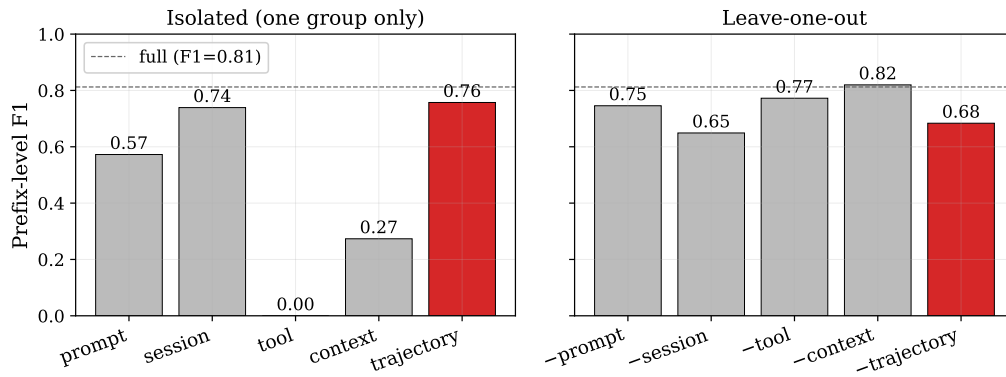


Figure 5: Feature-group ablation. Left: isolated; trajectory alone nearly matches the full detector. Right: leave-one-out; removing trajectory causes the largest drop. Dashed line marks full-model F_1 .

- Usage for Instant Payments. *Complex Systems Informatics and Modeling Quarterly* (10 2019), 72–88. doi:10.7250/csinq.2019-20.04
- [17] Mateusz Dziemian, Maxwell Lin, Xiaohan Fu, Micha Nowak, Nick Winter, Eliot Jones, Andy Zou, Lama Ahmad, Kamalika Chaudhuri, Sahana Chennabasappa, et al. 2026. How Vulnerable Are AI Agents to Indirect Prompt Injections? Insights from a Large-Scale Public Competition. *arXiv preprint arXiv:2603.15714* (2026).
- [18] Mohamed Amine Ferrag, Norbert Tihanyi, and Merouane Debbah. 2025. From llm reasoning to autonomous ai agents: A comprehensive review. *arXiv preprint arXiv:2504.19678* (2025).
- [19] Significant Gravitass. 2023. AutoGPT: An Autonomous GPT-4 Experiment. <https://github.com/Significant-Gravitas/AutoGPT>. Accessed: 2026-04-20.
- [20] Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. 2023. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM workshop on artificial intelligence and security*. 79–90.
- [21] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. 2022. Why do tree-based models still outperform deep learning on typical tabular data? *Advances in neural information processing systems* 35 (2022), 507–520.
- [22] Sven Gronauer and Klaus Diepold. 2022. Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review* 55, 2 (2022), 895–943.
- [23] Saidakhror Gulyamov, Said Gulyamov, Andrey Rodionov, Rustam Khursanov, Kambariddin Mekhmonov, Djakhongir Babaev, and Akmaljon Rakhimjonov. 2025. Prompt Injection Attacks in Large Language Models and AI Agent Systems: A Comprehensive Review of Vulnerabilities, Attack Vectors, and Defense Mechanisms. (2025).
- [24] Pablo Hernandez-Leal, Bilal Kartal, and Matthew E Taylor. 2019. A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems* 33, 6 (2019), 750–797.
- [25] Chengkai Huang, Junda Wu, Yu Xia, Zixu Yu, Ruhan Wang, Tong Yu, Ruiyi Zhang, Ryan A. Rossi, Branislav Kveton, Dongruo Zhou, Julian McAuley, and Lina Yao. 2025. Towards Agentic Recommender Systems in the Era of Multimodal Large Language Models. *arXiv:2503.16734 [cs.AI]* <https://arxiv.org/abs/2503.16734>
- [26] Brynn Knowlton, Jovani Campa, David Solis Gallo, Khalil Dajani, and Nabeel Alzahrani. 2026. Prompt-Based Jailbreaking of Leading LLM Chatbots: A Survey of Attacks and Defenses. *IEEE Transactions on Artificial Intelligence* (2026).
- [27] Yufeng Kou, Chang-Tien Lu, Sirirat Sirwongwattana, and Yo-Ping Huang. 2004. Survey of fraud detection techniques. In *IEEE international conference on networking, sensing and control, 2004*, Vol. 2. IEEE, 749–754.
- [28] Yufeng Kou, Chang-Tien Lu, S. Sirwongwattana, and Yo-Ping Huang. 2004. Survey of fraud detection techniques. In *IEEE International Conference on Networking, Sensing and Control, 2004*, Vol. 2. 749–754 Vol.2. doi:10.1109/ICNSC.2004.1297040
- [29] Donghyun Lee and Mo Tiwari. 2024. Prompt infection: Llm-to-llm prompt injection within multi-agent systems. *arXiv preprint arXiv:2410.07283* (2024).
- [30] Luoxi Meng, Henry Feng, Iliia Shumailov, and Earlene Fernandes. 2025. cellmate: Sandboxing browser ai agents. *arXiv preprint arXiv:2512.12594* (2025).
- [31] Gagan Mundada, Zihan Huang, Rohan Surana, Sheldon Yu, Jennifer Yuntong Zhang, Xintong Li, Tong Yu, Lina Yao, Jingbo Shang, Julian McAuley, and Junda Wu. 2026. WS-GRPO: Weakly-Supervised Group-Relative Policy Optimization for Rollout-Efficient Reasoning. *arXiv:2602.17025 [cs.LG]* <https://arxiv.org/abs/2602.17025>
- [32] Abed Mutemi and Fernando Bacao. 2024. E-commerce fraud detection based on machine learning techniques: Systematic literature review. *Big Data Mining and Analytics* 7, 2 (2024), 419–444.
- [33] Thanh Thi Nguyen, Ngoc Duy Nguyen, and Saeid Nahavandi. 2020. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE transactions on cybernetics* 50, 9 (2020), 3826–3839.
- [34] OpenAI. 2025. Codex: AI Coding Agent. <https://openai.com/codex/>. Accessed: 2026-04-20.
- [35] Qiuwu Sha, Tengda Tang, Xinyu Du, Jie Liu, Yixian Wang, and Yuan Sheng. 2025. Detecting credit card fraud via heterogeneous graph neural networks with graph attention. *arXiv preprint arXiv:2504.08183* (2025).
- [36] Rohan Surana, Xintong Li, Sheldon Yu, Yiran Jenny Shen, Chuhan Wang, Tong Yu, Prithviraj Ammanabrolu, Jingbo Shang, Julian McAuley, and Junda Wu. 2026. MASS-DPO: Multi-negative Active Sample Selection for Direct Policy Optimization. *arXiv:2605.10784 [cs.LG]* <https://arxiv.org/abs/2605.10784>
- [37] Rohan Surana, Gagan Mundada, Xunyi Jiang, Chuhan Wang, Zhenwei Tang, Difan Jiao, Zihan Huang, Yuxin Xiong, Junda Wu, Sheldon Yu, Xintong Li, Raghav Jain, Nikki Kuang, Sizhe Zhou, Bowen Jin, Zhendong Chu, Tong Yu, Ryan Rossi, Kuan-Hao Huang, Jingbo Shang, Jiawei Han, and Julian McAuley. 2026. Generate, Filter, Control, Replay: A Comprehensive Survey of Rollout Strategies for LLM Reinforcement Learning. *arXiv:2605.02913 [cs.LG]* <https://arxiv.org/abs/2605.02913>
- [38] Yaxin Tanga, Yijia Liua, Jiahe Lana, Zheng Yana, and Erol Gelenbec. [n. d.]. Security of LLM-based Agents Regarding Attacks, Defenses, and Applications: A Comprehensive Survey. ([n. d.]).
- [39] OpenClaw Team. 2024. OpenClaw: Open-Source AI Agent Framework. <https://github.com/openclaw/openclaw>. Accessed: 2026-04-20.
- [40] R Udayakumar, A Joshi, SS Boomiga, and R Sugumar. 2023. Deep fraud Net: A deep learning approach for cyber security and financial fraud detection and classification. *Journal of Internet Services and Information Security* 13, 3 (2023), 138–157.
- [41] William Van Melle. 1978. MYCIN: a knowledge-based consultation program for infectious disease diagnosis. *International journal of man-machine studies* 10, 3 (1978), 313–322.
- [42] Yizhu Wang, Sizhe Chen, Raghad Alkhudair, Basel Alomair, and David Wagner. 2025. Defending against prompt injection with datafilter. *arXiv preprint arXiv:2510.19207* (2025).
- [43] Jarrod West and Maumita Bhattacharya. 2016. Intelligent financial fraud detection: A comprehensive review. *Computers & security* 57 (2016), 47–66.
- [44] Sheldon Yu, Junda Wu, Xintong Li, Nikki Lijing Kuang, Sizhe Zhou, Tong Yu, Jiawei Han, Jingbo Shang, and Julian McAuley. 2026. OLIVIA: Online Learning via Inference-time Action Adaptation for Decision Making in LLM ReAct Agents. *arXiv:2605.11169 [cs.AI]* <https://arxiv.org/abs/2605.11169>
- [45] Haoxiang Zhang, Qixin Xu, Zhuofeng Li, Lei Zhang, Pengcheng Jiang, Yu Zhang, and Julian McAuley. 2026. Masking Stale Observations Helps Search Agents – Until It Doesn't: A Regime Map and Its Mechanism. *arXiv:2606.00408 [cs.CL]* <https://arxiv.org/abs/2606.00408>
- [46] Kaiyuan Zhang, Zian Su, Pin-Yu Chen, Elisa Bertino, Xiangyu Zhang, and Ninghui Li. 2025. LLM Agents Should Employ Security Principles. *arXiv preprint arXiv:2505.24019* (2025).