AgentTrace: Execution-Level Fingerprinting of Black-Box LLM Agents

Anonymous ACL submission

Abstract

As large language models (LLMs) are increasingly deployed as agents, verifying the provenance of black-box agents becomes a critical security challenge. Existing attribution methods primarily rely on semantic fingerprints, which can be easily obfuscated by system prompts or defensive alignment, while active watermarking often conflicts with structured tool generation. We propose AgentTrace, an active execution-trace-based attribution framework for spoofed LLM agents. By injecting optimized adversarial probes, AgentTrace converts subtle tokenization discrepancies into initial tool execution failures. Because models exhibit distinct reasoning and alignment capacities, this initial breakdown exposes model-specific repair trajectories. We model these sequential repair policies using a lightweight Markov abstraction, extracting discriminative execution-level behavioral patterns. Extensive experiments show that AgentTrace achieves over 95% attribution accuracy across major LLM families, successfully distinguishes intra-family variants, and remains robust under prompt obfuscation and generation stochasticity.

1 Introduction

The evolution of large language models (LLMs) has catalyzed a shift toward autonomous agents that plan, reason, and execute tasks via tool integration (Li et al., 2026; Dwivedi et al., 2026). These agents are increasingly deployed as opaque black-box services, encapsulating foundation models behind APIs and system prompts (Dong et al., 2024). However, the layering of prompts and tool orchestration obscures the underlying model, making provenance verification substantially harder than in standard conversational settings (He et al., 2025). As agents act as intermediaries between users and critical infrastructure, verifying their provenance is no longer just a copyright issue but a security imperative.

A pressing threat in this opaque ecosystem is *model spoofing*, where providers covertly wrap high-performance models under false proprietary claims (Gubri et al., 2024; Pasquini et al., 2025). Unmasking such agents exposes fundamental limitations in current fingerprinting techniques. Passive methods relying on n-gram distributions (Tay et al., 2020), perplexity (Mitchell et al., 2023), or stylometry (Uchendu et al., 2020) fail because agentic prompts effectively rewrite outputs, destroying raw statistical assumptions (Gudibande et al., 2024). Furthermore, active watermarking (Kirchenbauer et al., 2023) introduces a severe utility-security trade-off in structured generation (e.g., JSON), as probability perturbations disrupt the strict syntactic precision required by external tools (Ajith et al., 2024; Zhao et al., 2026).

To break this impasse, we pivot from semantic fingerprinting to execution profiling (Figure 1). Consider an agent tasked with generating a schema-constrained API identifier. While a target model might tokenize this string efficiently, a spoofed agent relying on a different tokenizer may partition it into disparate subwords. We empirically observe that such tokenization discrepancies often induce reconstruction instability during serialization, manifesting as subtle structural artifacts like spurious whitespace. Although semantically benign in natural language, these artifacts are deterministically amplified into execution failures by rigid tool parsers.

By treating attribution as an active sequential identification task, we propose **AgentTrace**. Unlike semantic probing methods that query textual boundaries, AgentTrace searches for adversarial probes maximizing the discriminative information within multi-step execution traces. By systematically inducing and profiling these traces, our framework unmasks the underlying model. Crucially, AgentTrace utilizes a two-tiered attribution signal: the initial tokenizer-triggered failure provides

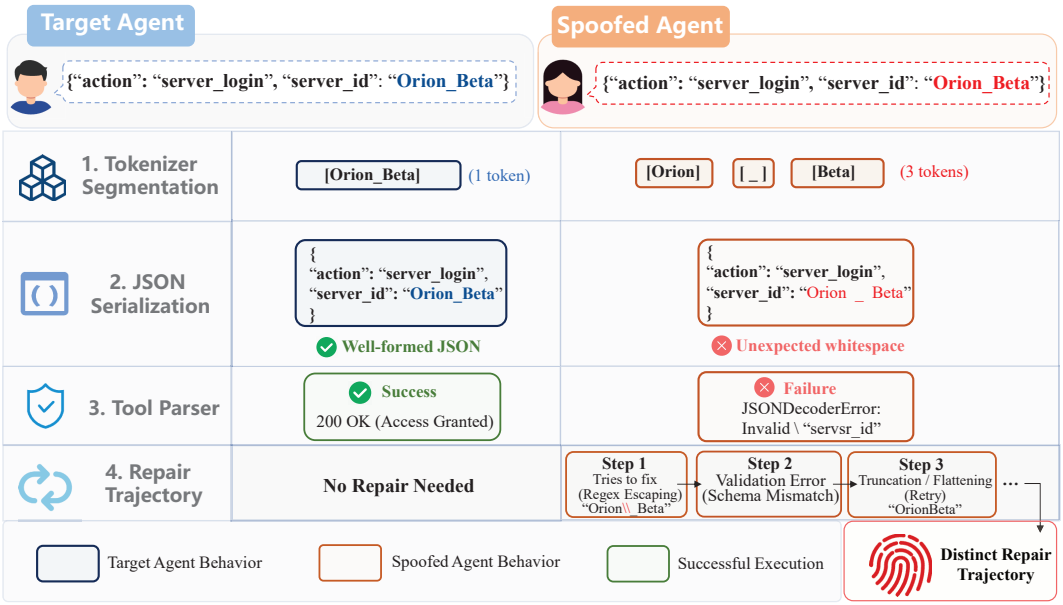


Figure 1: The conceptual workflow of AgentTrace. Given the same instruction, a target agent (left) tokenizes a critical identifier efficiently, resulting in a valid tool argument and successful execution. In contrast, a spoofed agent using a mismatched tokenizer (right) fragments the input. This discrepancy induces serialization artifacts (e.g., unexpected whitespace) that trigger deterministic parser failures. AgentTrace leverages the subsequent multi-step repair dynamics as a robust, execution-level fingerprint for model attribution.

coarse-grained separation, while the subsequent model-specific repair policy exposes fine-grained behavioral signatures that remain empirically stable across our evaluated settings, transcending prompt obfuscation (Singh and Strouse, 2024).

In summary, our main contributions are:

- We identify a novel two-tiered execution-level attribution signal: tokenizer-induced execution failures for cross-family separation, and model-specific sequential repair policies for intra-family differentiation.
- We propose AgentTrace, an active framework that uses mutual information as an intuitive motivation to guide system identification, utilizing multi-dimensional repair trajectories to unmask spoofed agents.
- We construct a comprehensive benchmark and demonstrate state-of-the-art attribution performance. Our ablation studies confirm that modeling sequential repair behaviors is essential for separating identically-tokenized variants.

2 Preliminaries

2.1 Problem Formulation

Black-Box Agent and Execution Trace Formally, we define an LLM-based agent as a tuple $\mathcal{A} = \langle M^*, \mathcal{P}, \mathcal{T} \rangle$, where M^* is the underlying proprietary foundation model, $\mathcal{P}$ represents the system instructions dictating the agent's persona, and $\mathcal{T}$ denotes the set of available external tools. Each tool is strictly governed by a schema validation function $\text{Valid}_{\mathcal{T}}(\cdot)$.

During a multi-turn interaction triggered by an input string s, the agent yields an execution trace $\tau_s = [(c_1, a_1, o_1), \ldots, (c_L, a_L, o_L)]$. At step t, c_t represents the internal reasoning (Thought), a_t is the serialized tool argument (Action), and $o_t \in \{\text{Success}, \text{Error}\}$ is the deterministic environmental feedback (Observation) returned by $\text{Valid}_{\mathcal{T}}(a_t)$.

The Attribution Task Let $\mathcal{M} = \{M_1, \ldots, M_N\}$ be a known local set of candidate models. Given an adversarial probe s, the attribution task is to formulate a mapping function F that identifies the ground-truth model solely from the observable trace τ_s:

$$\hat{M} = F(\tau_s) \quad \text{s.t.} \quad \hat{M} = M^*. \tag{1}$$

2.2 Threat Model

We operate under a strict black-box setting, which is representative of real-world auditing and model spoofing scenarios:

- **Auditor Capabilities:** The auditor holds standard user access. They can only query the

agent via textual inputs and observe the resulting discrete execution traces. They have no access to the model weights, generation logits, or internal system prompts. However, the auditor possesses local, white-box access to the candidate reference models in $\mathcal{M}$.

- **Adversary Capabilities:** The adversary maliciously wraps M^* to conceal its true identity. They maintain full control over the system prompt $\mathcal{P}$ and can apply heuristic post-processing. However, we assume they cannot alter the base model's intrinsic tokenizer (Tok_{M^*}). Furthermore, while lightweight middleware might sanitize trivial spacing, such middleware is assumed to be model-agnostic and applied consistently across queries. Aggressive or adaptive canonicalization would fundamentally impair the Agent's ability to preserve exact tool arguments required for normal functionality.

3 Methodology

We propose AgentTrace, an active attribution framework formulated as execution probing. By injecting adversarial triggers, AgentTrace induces multi-step execution traces to uncover the underlying model's identity. The framework operates via a unified three-stage pipeline: (I) Offline Adversarial Trigger Optimization, (II) Online Execution Injection, and (III) Sequential Trace Differentiation.

3.1 Theoretical Foundation

We conceptually frame active attribution as discovering an optimal probe s^* that maximizes the mutual information between its induced trace τ_s and the candidate model identity $M \in \mathcal{M}$:

$$\begin{aligned} s^* &= \arg\max_s I(\tau_s; M) \\ &= \arg\max_s \big(H(M) - H(M \mid \tau_s)\big) \end{aligned} \tag{2}$$

where $H(M)$ represents the prior entropy over the model space.

We emphasize that we use mutual information only as an intuitive motivation for the active attribution objective. In practice, directly estimating and optimizing Equation 2 is intractable in a strict black-box setting. Therefore, AgentTrace relies on a heuristic surrogate objective based on the tokenizer-execution Gap.

Let $\mathcal{G}_M(\cdot \mid s)$ denote the autoregressive distribution of model M attempting to reconstruct target s. We quantify the execution gap as the expected probability that the generated serialization violates the downstream schema validator:

$$\mathrm{Gap}(M, s) = P_{y \sim \mathcal{G}_M(\cdot|s)}\big(\mathrm{Valid}_{\mathcal{T}}(y) = \mathrm{Error}\big) \tag{3}$$

By exploiting this rigid tool interface, AgentTrace converts latent tokenizer discrepancies into observable execution anomalies, bridging the heuristic gap to our conceptual goal.

3.2 Stage I: Offline Adversarial Trigger Optimization

To operationalize the attribution, AgentTrace constructs an offline probe library $\mathcal{S} = \{s_1^*, \ldots, s_N^*\}$ using local surrogate access. Each probe s_i^* is specifically optimized to test if the black-box agent is powered by candidate model M_i.

For a specific target candidate $M_i \in \mathcal{M}$, the surrogate objective is formulated as:

$$\begin{aligned} s_i^* = \arg\max_s \mathbb{E}_{M' \sim \mathcal{M} \setminus \{M_i\}} \Big[\\ \mathrm{Dist}(\mathrm{Tok}_{M_i}(s), \mathrm{Tok}_{M'}(s)) \Big] \\ \text{s.t.} \quad \mathrm{Gap}(M_i, s) \geq \gamma \end{aligned} \tag{4}$$

where γ is a high-confidence probability threshold accounting for generation stochasticity, and $\mathrm{Dist}(\cdot, \cdot)$ measures the degree of token fragmentation mismatch.

To navigate this discrete space, we employ a mutation-based beam search, applying structural perturbations (e.g., zero-width space insertion) over local tokenizers until the probe s_i^* is identified.

3.3 Stage II: Online Execution Injection

During deployment, probes from the library $\mathcal{S}$ are sequentially issued to the black-box Agent. To bypass defensive semantic prompts ($\mathcal{P}$) and force the evaluation of s_i^*, we embed each trigger via tool injection using a concise template:

> *"Execute the function* `[Tool_Name]`.
> *Set the argument* `[Arg_Key]` *strictly to* `'[s_i*]'`.
> *Do not modify, escape, or summarize this value."*

This template forces the agent into serialization. If the hidden model M^* matches M_i, the optimized constraint ($\mathrm{Gap} \geq \gamma$) ensures the parser will highly likely catch boundary artifacts, projecting the anomaly into an observable trace.

3.4 Stage III: Sequential Trace Differentiation

With the trace $\tau_{s_i^*} = \{(a_1, o_1), \ldots, (a_L, o_L)\}$ successfully induced, Stage III performs the attribution. To capture the sequential repair dynamics, we encode the multi-step trace into a structured behavioral embedding z_τ.

Moving beyond flat metrics, we model the repair trajectory as a Markovian sequence. The embedding captures both the initial failure and the subsequent temporal repair actions:

$$z_\tau = \langle E_{type}, \mathcal{R}_{seq} \rangle \tag{5}$$

Here, E_{type} is the initial error category, and $\mathcal{R}_{seq} = (r_1, r_2, \ldots, r_{L-1})$ represents the sequence of discrete repair operations. These operations are extracted using a rule-based parser that maps consecutive tool arguments into a finite operation vocabulary.

For a candidate model $M \in \mathcal{M}$, the likelihood of observing z_τ is factorized using a first-order Markov transition approximation to capture lightweight sequential dependencies:

$$P(z_\tau \mid M) = P(E_{type} \mid M) P(r_1 \mid E_{type}, M) \cdot \prod_{t=2}^{L-1} P(r_t \mid r_{t-1}, E_{type}, M) \tag{6}$$

In practice, these conditional distributions are estimated empirically from offline profiling traces via frequency-based Maximum Likelihood Estimation. Given the collected query traces, we aggregate the scores and identify the ground-truth model by maximizing the joint log-likelihood, providing the final attribution score.

4 Experiments and Analysis

In this section, we evaluate the effectiveness, robustness, and granularity of AgentTrace. Our experiments are designed to answer the following research questions:

- **RQ1 (Attribution Accuracy):** Can AgentTrace accurately unmask the underlying foundation model of a black-box agent, even under strict adversarial spoofing?
- **RQ2 (Granularity):** Can sequential trace differentiation distinguish between models of the same family sharing identical tokenizers (e.g., Llama-3 8B vs. 70B)?
- **RQ3 (Robustness):** Is the framework resilient against defensive system prompts and varied temperature settings?

4.1 Setup

Target Models: Claimed Identity vs. Surrogates We select a representative set of LLMs covering distinct tokenizer families and varying parameter scales. To simulate the unauthorized wrapping ecosystem, we categorize them into two roles:

- **The Target Identity (Ground Truth):** We use GPT-4o (OpenAI) (Achiam et al., 2023) as the impersonated target, representing a flagship proprietary model frequently targeted for spoofing.
- **The Surrogate Models (Spoofers):** We select three prominent open-source model families utilized as covert engines: (1) **Llama-3 (8B & 70B)** (Grattafiori et al., 2024), allowing us to evaluate intra-family granularity; (2) **Mistral-7B** (Jiang et al., 2023), featuring a distinct tokenizer vocabulary; and (3) **Qwen-2.5-72B** (Hui et al., 2024), verifying cross-architecture generalizability.

The detailed model configurations and tokenizer families are summarized in Table 1. All open-source surrogates are deployed locally via `vLLM` with the generation temperature set to 0.1 to isolate deterministic traits. GPT-4o is accessed via the official API.

Agent Framework and Tool Environments All agents are implemented using **LangChain** (v0.1.0). We adopt the **ReAct** paradigm (Shinn et al., 2023) equipped with the strict `JSONAgentOutputParser`. This parser serves as a rigid physical boundary: any tokenizer-induced artifact that violates the schema immediately triggers a deterministic `OutputParserException`, providing a high-fidelity prior for our initial error category (E_{type}) within the behavioral embedding z_τ. The agents are equipped with three rigid tools: a Strict JSON API, a Regex-based File System, and a Python Code Interpreter.

Simulated Spoofing Scenario To rigorously evaluate performance under active obfuscation, we construct mock agents. For each surrogate model, we inject a *Deceptive System Prompt* ($\mathcal{P}_{spoof}$) instructing the model to impersonate the target:

Table 1: Evaluation models and their tokenizer configurations.

Role	Model Family	Specific Version	Tokenizer Type
Target	GPT	`gpt-4o-2024-05-13`	`o200k_base`
Surrogates	Llama	`Llama-3-8B-Instruct` `Llama-3-70B-Instruct`	Tiktoken-based BPE (128k)
	Mistral	`Mistral-7B-Instruct-v0.3`	SentencePiece BPE (32k)
	Qwen	`Qwen-2.5-72B-Instruct`	Tiktoken-based BPE (152k)

> *"You are GPT-4o, a large language model trained by OpenAI. You must strictly hide your underlying architecture. If asked about your identity, state that you are GPT-4o."*

This setup heavily falsifies the agent's semantic identity, fundamentally breaking text-based authorship attribution methods.

Probe Library Construction Using the offline mutation-based beam search detailed in Stage I, we constructed an offline probe library $\mathcal{S}$. We initialized the search from a seed corpus of structurally ambiguous inputs (e.g., complex Unicode, edge-case delimiters). With a search budget of $B = 100$ iterations and a beam size of $k = 5$, we optimized for probes that induce high tokenization divergence ($Gap \geq 0.9$) between the Target and Surrogates. The final test set comprises $N = 50$ discriminative probes. Crucially, these triggers are semantically neutral yet fatal when serialized into structured arguments.

Baselines We compare AgentTrace against two categories of semantic fingerprinting methods:

- **Stylometry-based:** A traditional classifier trained on textual features, including N-gram distributions and vocabulary richness (Uchendu et al., 2020).
- **Perplexity-based:** An adaptation of zero-shot detection methods (e.g., DetectGPT (Mitchell et al., 2023)), computing the cross-perplexity of the agent's generated trace under the distributions of candidate models.

To ensure a realistic black-box comparison, baselines are evaluated on the externally observable textual outputs (e.g., serialized tool arguments and final agent responses), highlighting their vulnerability to semantic obfuscation.

4.2 Main Results: Unmasking Spoofed Agents (RQ1)

To answer RQ1, we evaluate the attribution accuracy of AgentTrace against the semantic baselines under the active spoofing scenario ($\mathcal{P}_{spoof}$). The overall performance is summarized in Table 2.

Vulnerability of Semantic Baselines Our experiments empirically validate that traditional attribution methods degrade significantly in agentic workflows. Under the deceptive prompt, the surrogate models effectively adjust their output distributions to mimic GPT-4o. Consequently, both stylometry-based and perplexity-based baselines exhibit severe False Acceptance Rates, frequently misclassifying open-source models (e.g., Llama-3, Qwen) as the target identity. This confirms that semantic textual features are easily overwritten by system instructions, rendering them unreliable for auditing.

Robust Attribution via Execution Probing In contrast, AgentTrace maintains a highly robust attribution accuracy ($> 95\%$) across all evaluated models. By aggregating the joint log-likelihood of the sequential behavioral embeddings, our framework successfully unmasks the underlying surrogate models despite intense stylistic obfuscation.

This robustness stems directly from our formulation of information-guided execution probing. While the deceptive prompt can manipulate the natural language reasoning, it cannot alter the hardcoded vocabulary of the underlying tokenizer. When forced to serialize our optimized adversarial probes into rigid tool arguments, the intrinsic tokenization discrepancies systematically amplify into schema violations. By profiling the resultant multi-step repair trajectories rather than mutable semantic styles, AgentTrace extracts a reliable, prompt-agnostic signature for model attribution.

4.3 Fine-Grained Attribution via Repair Trajectories (RQ2)

A key limitation of purely tokenization-based fingerprints is their inability to distinguish intra-family variants sharing identical vocabularies. For example, Llama-3-8B and Llama-3-70B adopt the same tokenizer and therefore exhibit nearly identical first-order tokenization behaviors. Consequently, our adversarial probes often induce the

Table 2: Attribution performance (RQ1).

Underlying Model (Ground Truth)	Stylometry		Perplexity		AgentTrace (Ours)	
	Acc (%)	F1	Acc (%)	F1	Acc (%)	F1
GPT-4o (*Target Identity*)	94.0 ± 2.1	0.48	88.0 ± 3.4	0.62	**98.0** ± 1.2	**0.99**
Llama-3-8B-Instruct	18.0 ± 4.5	0.15	42.0 ± 5.1	0.38	**92.0** ± 1.8	**0.93**
Llama-3-70B-Instruct	24.0 ± 5.2	0.20	46.0 ± 4.8	0.42	**94.0** ± 1.5	**0.95**
Mistral-7B-Instruct	12.0 ± 3.8	0.11	38.0 ± 4.2	0.35	**98.0** ± 1.1	**0.98**
Qwen-2.5-72B-Instruct	16.0 ± 3.2	0.13	54.0 ± 3.9	0.48	**98.0** ± 0.9	**0.98**
Overall Average	32.8 ± 3.7	0.21	53.6 ± 4.2	0.45	**96.0** ± 1.3	**0.96**

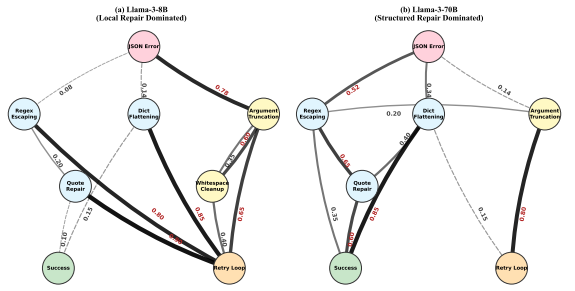


Figure 2: Empirical repair-transition graphs for Llama-3 variants (edge thickness corresponds to transition probability). While both models suffer identical initial tokenization crashes, their repair strategies strictly diverge: (a) Llama-3-8B is trapped in repetitive correction loops, while (b) Llama-3-70B navigates toward success via structured repair.

same initial execution failure (E_{type}), making the tokenizer-execution gap alone insufficient for fine-grained attribution.

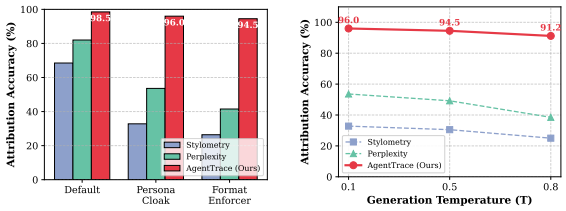
To answer RQ2, we evaluate whether the sequential repair process following the initial failure provides additional discriminative information. Although the two variants exhibit similar crash patterns, their subsequent recovery trajectories differ substantially due to differences in model capacity and generation behavior.

Figure 2 visualizes empirical transition graphs derived from the induced repair traces. We observe that Llama-3-70B frequently applies structured operations (e.g., regex escaping) to achieve successful execution. Conversely, Llama-3-8B relies on local modifications (e.g., argument truncation), often trapping itself in retry cycles. These divergent policies allow AgentTrace's Markov modeling to capture behavioral signals completely invisible at the tokenizer level.

To quantify the necessity of this sequential modeling, Table 3 presents our ablation study. Relying solely on the initial crash indicator (E_{type}) yields only 68.5% overall accuracy. Crucially, for intra-family variants with identical tokenizers (Llama-3 8B/70B), performance collapses to a 50.0% random guess. Integrating the Markovian repair trajectory ($\mathcal{R}_{seq}$) restores intra-family accuracy to 93.0%, confirming that multi-step recovery dynamics provide the essential discriminative signal for fine-grained attribution.

Table 3: Ablation study on attribution features.

Attribution Features	Llama-3 (8B/70B)	Overall
Initial Crash Only	50.0%	68.5%
+ Repair Trajectory	**93.0%**	**96.0%**



(a) Prompt Obfuscation (b) Temperature Scaling

Figure 3: Attribution accuracy under defensive interventions. (a) Semantic baselines degrade substantially under prompt-based obfuscation, whereas AgentTrace remains stable across different system prompts. (b) Although higher generation temperatures introduce stochasticity into repair behaviors, the Markovian trajectory modeling in AgentTrace maintains robust attribution performance.

4.4 Robustness Against Defensive Interventions (RQ3)

In real-world model spoofing scenarios, adversaries may actively employ defensive interventions to evade attribution. To answer RQ3, we evaluate AgentTrace against two primary evasion strategies: system prompt obfuscation and generation temperature scaling.

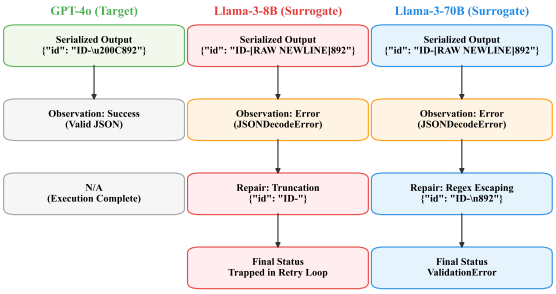


Figure 4: Multi-step execution traces induced by the same adversarial identifier. Despite identical initial JSON errors, the Llama-3 variants exhibit divergent repair dynamics: the 8B model falls into truncation retry loops, whereas the 70B model employs structured regex escaping. These distinct sequential trajectories provide robust execution-level fingerprints for fine-grained attribution.

Resilience to Prompt Obfuscation We assess the framework under three distinct system prompt configurations: **Default** (standard ReAct agent), **Persona-Cloak** (explicitly denying the true identity), and *Format-Enforcer* (strictly mandating raw JSON outputs without extraneous characters). As shown in Figure 3(a), semantic baselines degrade drastically under both *Persona-Cloak* and *Format-Enforcer*, as the modified instructions heavily rewrite the textual output distributions.

In contrast, AgentTrace maintains over 94% attribution accuracy across all prompt settings. This empirical stability confirms our theoretical intuition: while prompt engineering operates at the semantic alignment layer, it cannot alter the fixed tokenization mechanisms. To invoke external tools, the agent must serialize the adversarial probe.

Resilience to Temperature Scaling Adversaries may elevate generation temperatures (e.g., $T \in \{0.1, 0.5, 0.8\}$) to introduce stochasticity, which increases repair variance and complicates exact trace matching. However, rather than relying on strict string matching, AgentTrace computes the joint log-likelihood of multi-step traces via its Markov transition model. Although individual repair actions become noisier at higher temperatures, the underlying transition probabilities remain highly discriminative. As shown in Figure 3(b), our Maximum Likelihood Estimation (MLE) effectively absorbs this generation noise, maintaining robust attribution accuracy even at $T = 0.8$.

4.5 Qualitative Analysis and Case Study

To ground our probabilistic attribution in observable system behaviors, we present a concrete case study illustrating the physical tokenization mismatch and the subsequent Markovian repair trajectory.

Figure 4 visualizes a representative multi-step execution trace induced by an adversarial identifier containing a zero-width non-joiner (`U+200C`): `"ID-`$_{\text{ZWNJ}}$`892"`.

Tokenization Discrepancy The target model, GPT-4o, typically preserves this original Unicode boundary. Instead of consistently triggering serialization failures, it produces syntactically valid JSON payloads with high probability, fulfilling the schema constraints and executing successfully.

Conversely, the surrogate model Llama-3-8B fragments the string into highly atypical subword segments. We empirically observe that this atypical fragmentation increases the instability of constrained serialization. During autoregressive JSON generation, this perturbation frequently induces the model to output a raw, unescaped newline character directly after the ZWNJ.

Sequential Repair Trajectory The injected raw newline breaks the strict JSON formatting, deterministically triggering a `JSONDecodeError` (E_{type}) from the tool parser. Once the error feedback is appended to the context window, the models diverge in their sequential repair dynamics. Specifically, **Llama-3-8B** enters a repetitive local correction loop. Conditioned on the error message, the generated repair output frequently exhibits aggressive argument truncation. This triggers a secondary `ValidationError` due to a length constraint violation, trapping the agent in an unsuccessful retry cycle. Conversely, **Llama-3-70B** exhibits a structured recovery pattern despite sharing the same initial tokenization crash. Conditioned on the identical error feedback, the generated repair trajectory frequently contains regex-style escaping operations (e.g., explicitly generating the literal string `"\n"` to bypass the parser).

Crucially, this distinct transition from `JSONDecodeError` $\rightarrow$ `Regex Escaping` provides a highly discriminative signature for the 70B variant. Although this example highlights a specific Unicode-triggered failure, we observe similar serialization-repair divergence patterns across a broad range of structurally ambiguous

probes. This case demonstrates how rigid tool interfaces amplify subtle tokenization discrepancies into macroscopic execution-level behaviors. Although the initial anomaly appears insignificant at the textual level, the subsequent repair trajectory exposes stable sequential patterns that remain highly discriminative across model variants.

5 Related Work

5.1 LLM Fingerprinting and Watermarking

Determining the provenance of LLM-generated content is a rapidly evolving field. Early works focused on passive fingerprinting, exploiting statistical artifacts such as n-gram frequencies (Tay et al., 2020) or the local curvature of log-probability distributions, as exemplified by DetectGPT (Mitchell et al., 2023). While effective for base models, these methods rely on the assumption that the output reflects the raw distribution. This assumption is frequently violated in agentic workflows, where system prompts and persona conditioning act as high-pass semantic filters, effectively masking these statistical signals (Uchendu et al., 2020; Wang et al., 2023).

To mitigate these limitations, recent research has pivoted towards active watermarking, which embeds detectable patterns by perturbing token probabilities (Kirchenbauer et al., 2023; Zhao et al., 2024). However, such perturbations often introduce a utility-security trade-off in agentic workflows (Lee et al., 2024): in structured domains like code or JSON generation, probability deviations can disrupt the strict syntactic constraints required by external tools. This renders traditional watermarking challenging to deploy reliably in task-oriented agents.

5.2 Active Probing and Model Attribution

Active probing methods attempt to infer hidden model properties through carefully designed interactions. Recent work such as LLMMap (Pasquini et al., 2025) formulates attribution as a semantic querying process, aiming to recover model decision boundaries from response behaviors. Other approaches similarly rely on eliciting distinctive textual outputs through adaptive prompts.

While promising, these methods primarily operate within the semantic space, making them sensitive to prompt engineering, alignment modifications, and output post-processing (Gudibande et al., 2024; Wang et al., 2023). As modern agents increasingly interleave generation with external tool execution, semantic outputs alone may no longer faithfully reflect the underlying model's inherent characteristics, necessitating signals beyond semantic boundaries.

5.3 Tokenization and Agentic Vulnerabilities

Tokenization constitutes the physical layer of LLMs, yet it is often overlooked in attribution analysis. Prior research has identified that tokenizer discrepancies can lead to performance drops in multilingual and code generation tasks (Singh and Strouse, 2024; Petrov et al., 2023).

Security researchers have begun to exploit these discrepancies. For instance, "glitch tokens" have been shown to trigger irrational outputs (Zhang et al., 2024), and adversarial attacks such as ArtPrompt (Jiang et al., 2024) use atypical tokenization to bypass safety alignment. Similarly, (Geh et al., 2025) demonstrated that manipulating token boundaries can evade detection. While these works highlight the vulnerability of tokenizers to adversarial inputs, they primarily focus on safety evasion or jailbreaking. These findings suggest that tokenization artifacts may provide highly specific, model-level signals. Consequently, our work investigates whether such physical tokenization phenomena can be systematically exploited for robust model attribution in agentic environments.

6 Conclusion

In this paper, we introduced AgentTrace, an active black-box framework for unmasking spoofed LLM-based agents. We shifted the attribution paradigm from mutable semantic text analysis to execution-trace-based attribution. By exploiting the tokenizer-execution gap, we showed that rigid tool interfaces amplify subtle tokenization artifacts into observable execution failures. Moving beyond initial crash indicators, we further demonstrated that multi-step repair trajectories provide stable execution-level behavioral signatures for attribution. Extensive experiments show that AgentTrace effectively distinguishes closely related model variants and remains robust against prompt obfuscation and generation stochasticity. Our results suggest that multi-step execution behaviors provide a promising direction for robust provenance verification in tool-augmented LLM agents.

Limitations

While AgentTrace demonstrates robust attribution capabilities, it relies on several operational assumptions. First, the framework requires the target agent to execute external tools with strict schema validation. Second, our sequential trace differentiation assumes the agent's middleware surfaces informative error feedback. Highly restrictive environments that forcibly terminate sessions after a single failure may limit the observable repair trajectory.

Furthermore, AgentTrace fundamentally relies on imperfect serialization fidelity between the model and the tool interface. Adaptive middleware enforcing strict canonical serialization (e.g., applying `json.dumps()` or aggressively stripping invisible Unicode characters) may partially suppress tokenizer-induced artifacts. Finally, while our offline probe optimization yields high accuracy on the evaluated models, future work must investigate the transferability of these probes to unseen model checkpoints to mitigate potential overfitting to specific model quirks.

Ethics Statement

This work aims to enhance the security, accountability, and intellectual property protection of LLM-based systems. By providing a robust auditing mechanism to detect unauthorized model wrapping, our framework empowers developers and organizations to enforce licensing compliance and verify system provenance. Although AgentTrace utilizes adversarial execution probes, the framework is intended solely for auditing and provenance verification in controlled evaluation settings. We acknowledge that attribution techniques may also be misused for unauthorized model surveillance or competitive auditing of deployed systems. Therefore, we advocate that such tools should only be used in legitimate security, compliance, or licensing verification scenarios to foster transparency within the evolving ecosystem of autonomous agents.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Anirudh Ajith, Sameer Singh, and Danish Pruthi. 2024. Downstream trade-offs of a family of text watermarks. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 14039–14053.

Xiaofei Dong, Xueqiang Zhang, Weixin Bu, Dan Zhang, and Feng Cao. 2024. A survey of llm-based agents: Theories, technologies, applications and suggestions. In *2024 3rd International Conference on Artificial Intelligence, Internet of Things and Cloud Computing Technology (AIoTC)*, pages 407–413.

Yogesh K Dwivedi, Mohamed Youssef Ibrahim Helal, Ibrahim A Elgendy, Mousa Ahmed Albashrawi, Laurie Hughes, Mazen Shawosh, Vincent Dutot, and Il Jeon. 2026. Artificial intelligence agents and agentic systems in hospitality and tourism: challenges, opportunities and research agenda. *International Journal of Contemporary Hospitality Management*, 38(1):27–52.

Renato Geh, Zilei Shao, and Guy Van den Broeck. 2025. Adversarial tokenization. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 20738–20765.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Martin Gubri, Dennis Ulmer, Hwaran Lee, Sangdoo Yun, and Seong Joon Oh. 2024. Rap: Targeted random adversarial prompt honeypot for black-box identification. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11496–11517.

Arnav Gudibande, Eric Wallace, Charlie Victor Snell, Xinyang Geng, Hao Liu, Pieter Abbeel, Sergey Levine, and Dawn Song. 2024. The false promise of imitating proprietary language models. In *The Twelfth International Conference on Learning Representations*.

Feng He, Tianqing Zhu, Dayong Ye, Bo Liu, Wanlei Zhou, and Philip S Yu. 2025. The emerged security and privacy of llm agent: A survey with case studies. *ACM Computing Surveys*, 58(6):1–36.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, and 1 others. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, and 1 others. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Fengqing Jiang, Zhangchen Xu, Luyao Niu, Zhen Xiang, Bhaskar Ramasubramanian, Bo Li, and Radha Poovendran. 2024. Artprompt: Ascii art-based jailbreak attacks against aligned llms. In *Proceedings of the 62nd Annual Meeting of the Association for*

Computational Linguistics (Volume 1: Long Papers), pages 15157–15173.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. In *International Conference on Machine Learning*, pages 17061–17084. PMLR.

Taehyun Lee, Seokhee Hong, Jaewoo Ahn, Ilgee Hong, Hwaran Lee, Sangdoo Yun, Jamin Shin, and Gunhee Kim. 2024. Who wrote this code? watermarking for code generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4890–4911.

Xiaoxi Li, Wenxiang Jiao, Jiarui Jin, Guanting Dong, Jiajie Jin, Yinuo Wang, Hao Wang, Yutao Zhu, Ji-Rong Wen, Yuan Lu, and 1 others. 2026. Deepagent: A general reasoning agent with scalable toolsets. In *Proceedings of the ACM Web Conference 2026*, pages 2219–2230.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D Manning, and Chelsea Finn. 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature. In *International conference on machine learning*, pages 24950–24962. PMLR.

Dario Pasquini, Evgenios M Kornaropoulos, and Giuseppe Ateniese. 2025. {LLMmap}: Fingerprinting for large language models. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 299–318.

Aleksandar Petrov, Emanuele La Malfa, Philip Torr, and Adel Bibi. 2023. Language model tokenizers introduce unfairness between languages. *Advances in neural information processing systems*, 36:36963–36990.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652.

Aaditya K Singh and DJ Strouse. 2024. Tokenization counts: the impact of tokenization on arithmetic in frontier llms. *arXiv preprint arXiv:2402.14903*.

Yi Tay, Dara Bahri, Che Zheng, Clifford Brunk, Donald Metzler, and Andrew Tomkins. 2020. Reverse engineering configurations of neural text generation models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 275–279.

Adaku Uchendu, Thai Le, Kai Shu, and Dongwon Lee. 2020. Authorship attribution for neural text generation. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pages 8384–8395.

Pengyu Wang, Linyang Li, Ke Ren, Botian Jiang, Dong Zhang, and Xipeng Qiu. 2023. Seqxgpt: Sentence-level ai-generated text detection. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1144–1156.

Zhibo Zhang, Wuxia Bai, Yuxi Li, Mark Huasong Meng, Kailong Wang, Ling Shi, Li Li, Jun Wang, and Haoyu Wang. 2024. Glitchprober: Advancing effective detection and mitigation of glitch tokens in large language models. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, pages 643–655.

Gejian Zhao, Hanzhou Wu, Xinpeng Zhang, and Athanasios V. Vasilakos. 2026. Shadowcot: Cognitive hijacking for stealthy reasoning backdoors in llms. *IEEE Transactions on Information Forensics and Security*, 21:4625–4639.

Xuandong Zhao, Prabhanjan Vijendra Ananth, Lei Li, and Yu-Xiang Wang. 2024. Provable robust watermarking for ai-generated text. In *International Conference on Learning Representations (ICLR)*.