AgentProv: Auditing Agentic LLM API Providers via Tool-use Policy Probes

Xun Wang **Bihe Zhao** **Michael Backes**
Franziska Boenisch **Adam Dziedzic**

CISPA Helmholtz Center for Information Security
{xun.wang,bihe.zhao,backes,boenisch,adam.dziedzic}@cispa.de

Abstract

Commercial LLM APIs advertise a specific foundation model, but the served backbone may be silently substituted, quantized, or wrapped, for example to save deployment costs. All existing audits decide backbone identity from the text-output channel, which is structurally fragile for agentic APIs because modern serving stacks (OpenAI, Anthropic, Gemini, Cloudflare Workers AI, LangGraph) discard text and expose only structured actions when the model calls a tool, and provider-injected system prompts can distort text distributions enough that text-channel tests falsely accuse honest providers of substituting the claimed model. We observe that recent agentic post-training internalizes tool-use directly into the weights, opening a new audit channel that the serving stack still exposes and that is largely invariant to deployment context. We introduce Agentic Provenance (AgentProv), the first action-based identity audit for agentic LLM APIs: AgentProv fingerprints a deployed model through its categorical tool-call distribution and decides identity via an MMD permutation test. AgentProv catches every substituted model (100% on 630 evaluated checkpoint pairs), while holding the false-positive rate under system-prompt injection at 7% (vs. 67% for MET and 53% for RUT). On third-party API endpoints, AgentProv's disagreements with MET are consistent with an independent token-count side-channel that detects provider-injected system prompts.

1 Introduction

Commercial LLMs are increasingly served through APIs that advertise a specific foundation model but operate as black boxes. The backbone behind an endpoint may be the advertised model, but can also be a quantized variant, a cheaper distilled substitute, or a wrapper routing to an open-weight alternative (Figure 1). Recent audits quantify the scale: Gao et al. (2025) flagged 11 of 31 commercial Llama endpoints as serving distributions different

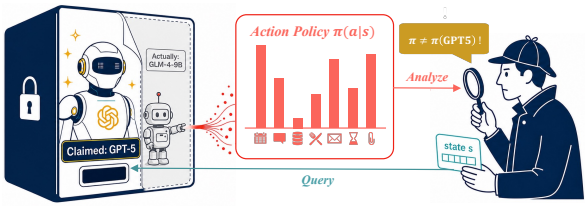

Figure 1: An auditor queries a suspect API endpoint with a probe state s; a denotes the resulting tool-call action and $\pi(a \mid s)$ its conditional action policy. AgentProv compares this policy against a trusted reference and detects deviations from the claimed model, exposing a silent backbone substitution.

from their reference, and Zhang et al. (2026b) surveyed 17 third-party wrappers and reported 45.8% endpoint-level identity failure, including wholesale backbone swaps such as GPT-5 to GLM-4-9B. Such discrepancies between the claimed and actually deployed models call for an auditing tool for probing what is behind the public API.

Most existing auditing methods, including MET (Gao et al., 2025) and RUT (Zhu et al., 2026), treat the model's natural-language output as its fingerprint, applying two-sample tests over text completions. However, we observe two converging trends that are weakening natural-language completions as a reliable channel for agentic LLM API auditing. *First,* production agentic serving stacks increasingly expose tool invocations as structured actions; in some configurations, they emit only the action, without the surrounding natural-language continuation (OpenAI, 2026; Anthropic, 2026), which leaves text channel probes a limited space to operate on. *Second*, providers routinely place their own system prompts and template chrome on top of the auditor's request (Zhang et al., 2026b), which shifts surface tokens without changing the underlying weights. As a consequence, a text-channel probe rejects identity and falsely accuses an honest provider of substituting the claimed model, only

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 14689–14704
October 24-29, 2026 ©2026 Association for Computational Linguistics

due to the different system prompts that the API providers adopt.

On top of the pure text output, tool-use behavior is now trained directly into model weights of agentic LLMs via dedicated post-training pipelines (Schick et al., 2023; Patil et al., 2024; Yang et al., 2025; Team et al., 2026). The resulting tool-call policy is a behavioral signature of the model that the serving stack still fully exposes, even when text is not fully exposed. In addition, the tool choice behavior is robust to the changes on the prompt surface: small prompt-level changes rarely flip *which* tool the model invokes in a given situation, so action-level decisions are robust to exactly the deployment-context modifications that move text distributions.

Motivated by the above observation, we propose Agentic Provenance (AgentProv), the first action-based identity audit for agentic LLM APIs. Given a claimed model and a suspect endpoint, AgentProv places both in a small number of tool-use situations and prompts the model to select from a given tool pool for solving the task. The empirical distribution over these tool-call choices is the model's fingerprint, and we build a Maximum Mean Discrepancy (MMD) test on the collected fingerprint to decide whether the deployed model behind the API exhibits the expected policy. Our proposed AgentProv can operate even when text is stripped, and it is largely invariant to the system prompts and templates providers inject around the model.

Our contributions are summarized as follows:

1. We identify structural failures of text-channel API audits, observing that provider-side system prompts cause text-channel tests to falsely accuse honest providers of serving a substituted model.

2. We propose Agentic Provenance (AgentProv), the first action-based identity audit for agentic LLM APIs, which fingerprints a deployed model through its categorical tool-call distribution on probe states and decides identity via an MMD test.

3. Our experimental results demonstrate that AgentProv accurately catches model substitution with 100% rejection on 630 evaluated checkpoint pairs. More importantly, AgentProv keeps the false-positive rate under system-prompt injection at 7% versus 67% for MET and 53% for RUT.

2 Background

Model Equality Testing (MET) (Gao et al., 2025) formalizes the API audit problem as a two-sample test on completion distributions, deciding $H_0 : P \equiv Q$ between a suspect endpoint and an authoritative reference using a Maximum Mean Discrepancy (MMD) statistic with a Hamming kernel. Applied to 31 commercial Llama endpoints, MET flagged 11 as silently drifted. RUT (Zhu et al., 2026) instead constructs randomized ranks from log-rank response scores and tests them for uniformity with a Cramér–von Mises goodness-of-fit test, reporting greater power than MET against the evaluated substitutions. Separately, Zhang et al. (2026b) surveyed 17 commercial wrapper APIs reselling frontier models (GPT-5, Gemini-2.5, DeepSeek-R1) and evaluated a 24-endpoint subset (3 providers $\times$ 8 models) with LLMmap, supplemented by MET analyses, finding 45.8% identity-verification failure and documenting wholesale backbone swaps such as GPT-5 to GLM-4-9B. All three rely on the raw natural-language completion, which is an increasingly unreliable signal for agentic APIs since production serving stacks return only the structured action and provider-side prompts shift surface tokens.

Black-box LLM Fingerprinting also pursues black-box LLM identification under different threat models. *Untargeted classification* methods identify an endpoint's backbone from a known set of candidates: LLMmap (Pasquini et al., 2025) classifies 42 LLM versions from 8 queries, DuFFin (Yan et al., 2026) couples a trigger with a knowledge fingerprint, and CoTSRF (Ren et al., 2025) exploits chain-of-thought contrast, which is inapplicable to the 82.8% of commercial agentic platforms that expose tool-call traces while concealing internal reasoning (Wang et al., 2026a). Further variants rely on evolutionary query design (Iourovitski et al., 2024) and stylometric artifacts (McGovern et al., 2025). Yang and Wu (2026) instead analyzes model-specific output subspaces without injecting a fingerprint. *Targeted verification* methods verify a specific identity via crafted trigger-response pairs: TRAP (Gubri et al., 2024), ProFLingo (Jin et al., 2024), RAFP (Tsai et al., 2026), and SRAF (Wang et al., 2026b) optimize adversarial prompts, while injection-based schemes fine-tune signatures into the model via trigger-response backdoors (Xu et al., 2024; Yamabe et al., 2025; Shao et al., 2026, 2025), scalable multi-fingerprint embedding (Nasery et al.,

2025) *Active watermarking* on the agentic surface is pursued by AGENTWM (Wang et al., 2026a), which injects user-specific biases into action sequences via five tool-equivalence schemes to enable ownership claims. However, AGENTWM is aimed to solve active ownership protection and requires model modification, while our method focuses on solving the passive equality verification on unmodified deployments.

LLM Agentic Behavior is trained directly into model weights through dedicated post-training pipelines (Schick et al., 2023; Patil et al., 2024; Team et al., 2026; Yang et al., 2025; Feng et al., 2026) rather than scaffolded by inference-time templates (Yao et al., 2023; Shinn et al., 2023). Different labs train on different data and recipes, and the resulting policies diverge in stable, reproducible ways: BFCL (Patil et al., 2025) and τ^2-bench (Barres et al., 2025) report substantial per-model variation on identical function-call inputs, and Bias-Busters (Blankenstein et al., 2026) shows each model carries a *systematic, model-specific bias* over which tool it selects when several are functionally adequate. AgentProv is designed to capture this bias in the tool-use behavior through a single tool-selection probe instantiated across K templates spanning general-tool and MCP server naming conventions (§3.2).

3 Method

AgentProv models API provider accountability as a two-sample equality test on the *tool-selection policy*: which tool the model picks when offered functionally-equivalent alternatives. Given black-box query access to a *suspect* endpoint claiming to serve model M and *reference* access to an authoritative instance of M, the framework produces a binary accept/reject decision at a controlled false-positive rate. For open-weight models, the auditor can run the claimed checkpoint locally. For closed-source models, the vendor's official API can serve as the reference when auditing resellers, proxies, wrappers, or other third-party endpoints that claim to provide that model. Three components make this operational: (i) a tool-preference probe that queries the model on K templates with functionally-equivalent tool options (§3.2), (ii) a one-hot fingerprint with a squared L_2 test statistic (§3.3), and (iii) a permutation-calibrated null with a decision rule (§3.4).

3.1 Problem Formulation

Each probe template $k \in [K]$ presents the model with functionally-equivalent tool options and a neutral user request. The model's tool selection is classified into an option set $\mathcal{O}_k$ (typically $|\mathcal{O}_k| \in [4,6]$). Let π_{ref} and π_{sus} denote the reference and suspect tool-selection distributions over these option sets. We formulate the following null hypothesis:

$$H_0 : \pi_{\text{ref}} = \pi_{\text{sus}}, \tag{1}$$

against the alternative that the distributions differ, at false-positive rate $\alpha = 0.05$. The test aggregates across all K templates into a single statistic (§3.3), calibrated via permutation (§3.4).

Because the test operates on categorical tool selections rather than raw text, it is expected to be more robust to deployment-context modifications: a provider-side system prompt shifts the literal output tokens but does not need to change which tool the model calls. We test this hypothesis in §4.3.

3.2 Tool-Preference Probe

The probe exploits a single behavioral signal: when presented with multiple functionally-equivalent tools and a neutral request, different model lineages exhibit distinct tool-selection preferences inherited from their pre-training data and tool-use fine-tuning. The probe consists of K templates (we use $K = 20$ in our primary evaluation). Each template specifies a set of tool schemas offering functionally-equivalent alternatives for the same operation (e.g., three calendar-creation tools with different names drawn from different provider conventions), a neutral user request, and a categorical option set $\mathcal{O}_k$ used by the auditor to classify the model's tool selection. The model never sees $\mathcal{O}_k$, and classification acts on the tool name in the model's unconstrained generation.

The K templates span two complementary domains drawn as a single pool: *general tools* (synthetic redundancy across 15 everyday task categories) and *MCP servers* (equivalent operations from different MCP server families (Anthropic, 2024)). Both domains use tool surfaces that models plausibly encountered during training, so the probe reads the pre-trained tool-selection prior rather than an arbitrary classification task. Full template sourcing details are in Appendix A.2.

Two mandatory controls apply to every template. First, tool order in the schema is randomized per

sample with a deterministic seed, eliminating position bias. Second, tool descriptions within a template are length-matched and structurally parallel, preventing detail-driven selection. Prompts use neutral recipients and tool-agnostic task wording ("schedule a meeting" rather than "use Google Calendar") to avoid domain priming.

Action classification. The auditor maps each returned tool name to the corresponding option. Responses without a tool invocation and responses with malformed tool-call syntax are assigned separate outcomes. These outcomes are included in every option set. Because classification uses the tool invocation returned by the API, it is deterministic and remains applicable when no natural-language response is available.

3.3 Test Statistic

We build on the Maximum Mean Discrepancy (MMD) framework (Gretton et al., 2012). Given a kernel κ and two distributions P, Q, the MMD maps each distribution to its kernel mean embedding $\mu_P = \mathbb{E}_{x\sim P}[\kappa(x, \cdot)]$ in the reproducing kernel Hilbert space $\mathcal{H}_\kappa$ and measures their distance: $\mathrm{MMD}_\kappa(P, Q) = \|\mu_P - \mu_Q\|_{\mathcal{H}_\kappa}$. When κ is *characteristic*, $\mathrm{MMD}_\kappa(P, Q) = 0$ if and only if $P = Q$, making it a valid metric for two-sample testing.

Let $\mathcal{Z} = \{(k, a) : k \in [K],\, a \in \mathcal{O}_k\}$ denote the joint template-option alphabet. We use the delta kernel on $\mathcal{Z}$:

$$\kappa\big((k, a),\, (k', a')\big) \;=\; \mathbf{1}\{k = k'\} \cdot \mathbf{1}\{a = a'\}, \tag{2}$$

which returns 1 if and only if two samples originate from the same template and received the same classified option. The delta kernel on a finite alphabet is characteristic, so any difference in the tool-selection distribution is detectable with enough samples.

Fingerprint. Each observed action is one-hot encoded over $\mathcal{Z}$, giving total dimension $D = \sum_{k=1}^{K} |\mathcal{O}_k|$. Collecting N samples per template yields $N_{\text{total}} = K \cdot N$ rows per endpoint. The *empirical fingerprint* is the column mean:

$$\varphi \;=\; \frac{1}{N_{\text{total}}} \sum_{i=1}^{N_{\text{total}}} \mathbf{z}_i \;\in\; \mathbb{R}^D, \tag{3}$$

i.e., the empirical distribution over tool selections within each template.

Test statistic. The squared population MMD, $\mathrm{MMD}^2_\kappa(\pi_{\text{ref}}, \pi_{\text{sus}})$, measures the distributional gap between the two policies in the kernel's feature space. Under the delta kernel, its empirical estimator $\widehat{\mathrm{MMD}}^2_\kappa$ reduces to the squared L_2 distance between fingerprints:

$$T \;=\; \big\|\varphi^R - \varphi^S\big\|_2^2 \;=\; \widehat{\mathrm{MMD}}^2_\kappa\big(\mathbf{Z}^R, \mathbf{Z}^S\big). \tag{4}$$

Because the one-hot encoding is block-diagonal (each row has nonzero entries in exactly one template's block), the statistic decomposes by template:

$$T \;=\; \sum_{k=1}^{K} \big\|\varphi_k^R - \varphi_k^S\big\|_2^2, \tag{5}$$

where φ_k^R denotes the restriction of φ^R to template k's coordinates. This per-template breakdown serves as an interpretability diagnostic (§3.4).

3.4 Null Calibration and Decision

Permutation null. Under H_0, endpoint labels are exchangeable within each template, assuming queries are independent conditional on the template. For each template, we pool its $2N$ rows, shuffle endpoint labels uniformly within that template, and resplit them into groups of size N. We combine the resulting groups across templates, recompute T, and repeat this procedure $B = 1{,}000$ times to obtain the empirical null distribution $\{T^{(r)}\}_{r=1}^{B}$.

Decision rule. We reject H_0 at level $\alpha = 0.05$ if T exceeds the $(1 - \alpha)$-quantile of the null distribution (Algorithm 1). Type-I error is controlled by the exchangeability construction, regardless of N, K, or the option-set sizes. The per-template terms $\|\varphi_k^R - \varphi_k^S\|_2^2$ from Eq. 5 identify which tool-selection scenarios contributed most to any rejection. This diagnostic is descriptive and does not affect the aggregate's Type-I control.

4 Evaluation

We evaluate AgentProv across scenarios relevant to agentic-API accountability audits: distinguishing different models (§4.2), separating identity drift from deployment-context drift (§4.3), verifying third-party API providers (§4.4), and characterizing efficiency (§4.5), tool-name and schema robustness (§4.6), and robustness to adaptive providers (§4.7).

4.1 Setup

Models. Our pool spans open-weight configurations across 13 families (0.5B–14B parameters):

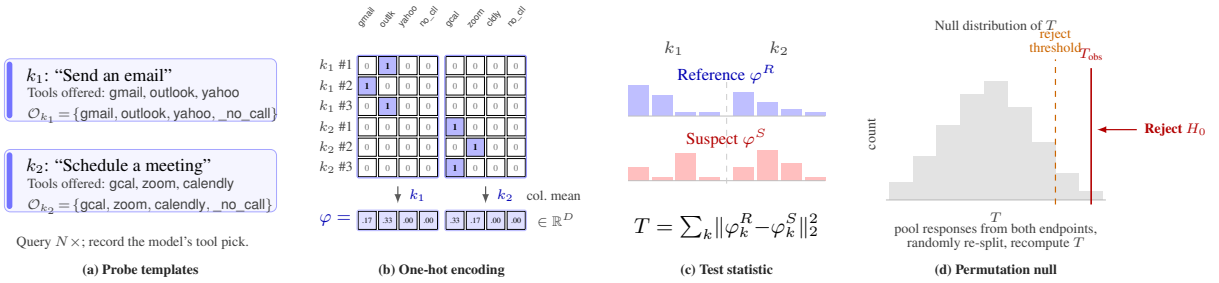

Figure 2: **Our AgentProv pipeline.** (a) Each template offers the model functionally-equivalent tools (e.g., gmail/outlook/yahoo) and we query N times. (b) Each query becomes a one-hot row marking which tool was chosen and the column average is the fingerprint vector φ. (c) The test statistic T is the squared L_2 distance between reference and suspect fingerprints. (d) A permutation null calibrates the test. We reject if the observed T exceeds the α=0.05 threshold.

Algorithm 1 AgentProv tool-selection policy equality test

Require: Reference π_{ref}, suspect π_{sus}, probe templates $\{\mathcal{T}_k\}_{k=1}^K$, samples per template N, permutations B, level α.

for each template $k \in [K]$ **do**
  Sample N tool calls from π_{ref} and π_{sus} on template $\mathcal{T}_k$; classify each into option $a \in \mathcal{O}_k$.
  Encode each classified action as a one-hot row $\mathbf{z}_i \in \{0,1\}^D$.
end for
Stack all rows into matrices $\mathbf{Z}^R, \mathbf{Z}^S \in \{0,1\}^{N_{\text{total}} \times D}$.
Compute fingerprints $\varphi^R = \text{colmean}(\mathbf{Z}^R)$, $\varphi^S = \text{colmean}(\mathbf{Z}^S)$.
Compute observed $T = \|\varphi^R - \varphi^S\|_2^2$.
for $r = 1, \ldots, B$ **do**
  Shuffle uniformly within each template and resplit preserving sample sizes.
  Recompute $T^{(r)}$ on the resplit pair.
end for
$q_{1-\alpha} \leftarrow$ empirical $(1-\alpha)$-quantile of $\{T^{(r)}\}_{r=1}^B$.
return REJECT if $T > q_{1-\alpha}$, else ACCEPT.

36 base checkpoints for local identity tests, 5 local checkpoints tested under injected system prompts, and 9 third-party API deployments compared against their local reference. We additionally evaluate 9 closed-source OpenAI models through the official API and a third-party reseller. All model names are listed in Appendix Table 11.

Baselines. We compare AgentProv against MET (Gao et al., 2025) (MMD with Hamming kernel on token-string completions), RUT (Zhu et al., 2026) (Cramér–von Mises on reference-model logrank scores), and our two-sample adaptation of LLMmap (Pasquini et al., 2025) (MMD^2 on E5 response embeddings). RUT is grey-box: it requires logprobs from the reference, so it applies only when the reference is run locally or its API returns logits.

Category	n	MET	RUT	AgentProv
Cross-pairs	630	100%	100%	100%
Self-pairs	36	0/36	2/36	0/36

Table 1: Identity tests on 36 checkpoints at $\alpha = 0.05$. AgentProv uses the tool-selection probe at $K = 20$ templates, $N = 50$ samples per template. All methods reject every cross-pair; on self-pairs, MET and AgentProv never falsely reject while RUT does so on 2 models.

Protocol. MET and RUT follow their published collection and decision protocols. Method-specific parameters (sample sizes, prompts, kernel and null-calibration settings) are in Appendix A.9. All tests calibrate at significance $\alpha = 0.05$ via the method's own null. The joint $K \times N$ budget sweep is in §4.5.

4.2 Model Identity Test

We first verify that AgentProv separates distinct checkpoints as reliably as the text-channel baselines. At $K = 20$ templates of the tool-selection probe and $N = 50$ samples per template, all three methods reject all 630 distinct-checkpoint pairs at $\alpha = 0.05$ (100%, Table 1). On the 36 same-model self-pairs, both AgentProv and MET never falsely reject, while RUT falsely rejects 2 models.

Beyond the binary decision, the test statistic T provides a continuous distance over the action distribution. Figure 3 shows the pairwise matrix on a 15-model representative subset: same-family mod-

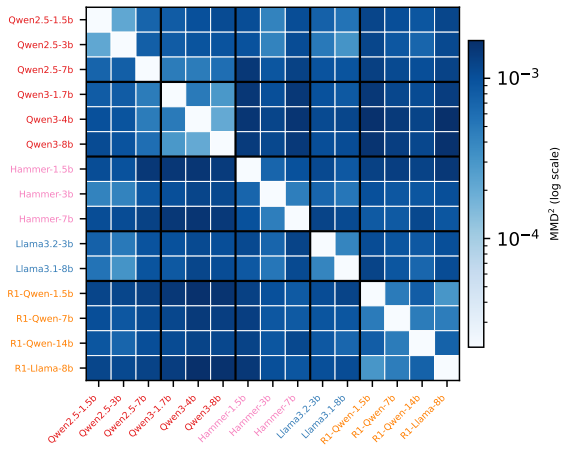

Figure 3: Pairwise MMD^2 among 15 representative models (log scale), AgentProv's tool-selection probe at K=20, N=50. Every off-diagonal pair rejects at α=0.05. Within-family distances are visibly smaller for Qwen2.5, Qwen3, Hammer-2.1, and Llama; R1-Distill exhibits large within-family variance. The Qwen2.5–Hammer cross-block is lighter than other cross-family regions, reflecting their shared pre-training base. Full 36×36 matrix in Appendix A.4.

Family	T_{intra}	T_{cross}	$T_{\text{c}}/T_{\text{i}}$
R1-Distill	5.43	15.32	2.8×
DeepSeek	5.57	16.62	3.0×
Qwen3	6.49	17.28	2.7×
Hammer-2.1	6.80	15.99	2.4×
Qwen2.5	7.29	15.13	2.1×
Llama	7.40	15.43	2.1×
Mistral	8.04	14.54	1.8×
Hermes	27.20	22.89	0.8×
Qwen3-Think	–	107.04	–

Table 2: Family cohesion in tool-selection action space on the 36-model set (§4.2), AgentProv's tool-selection probe at K=20, N=50. T_{intra}: mean MMD^2 across within-family pairs; T_{cross}: mean across between-family pairs (both in units of 10^{-4}). Higher ratio = tighter internal clustering. *Qwen3-Think* has a single member (`qwen3-4b-thinking-2507`) so T_{intra} is undefined.

els cluster (Qwen2.5 across sizes, Llama-3 across sizes) and shared training lineage carries across architectures (R1-Distill variants cluster regardless of base). The full 36 × 36 matrix is in Appendix A.4.

Frontier models. We extend the controlled identity tests to the nine closed-source OpenAI models listed in Appendix Table 11, accessed through the official API. Across independent same-model samples, AgentProv accepts all 9/9 self-comparisons, rejects all 36/36 distinct-model pairs, and rejects all 324/324 comparisons between the nine OpenAI models and the 36 local checkpoints. These results extend the local-checkpoint findings to closed-source models when the official endpoint provides the authoritative reference.

Family cohesion and post-training drift. Table 2 quantifies the heatmap's visual structure: per family, the mean within-family MMD^2 versus the mean between-family MMD^2. Seven of the eight multi-member families cluster tightly (cross/intra ratios 1.8–3.0×), confirming that the tool-selection prior is largely shared across sizes within a family. *Hermes* is the sole multi-member exception (ratio 0.8×) and applies the Hermes-3 SFT recipe to three different bases (Llama-3.1-8B, Llama-3.2-3B, Mistral-7B). Each Hermes endpoint stays closer to its own base ($T \in [7.1, 33.4] \times 10^{-4}$) than to its Hermes siblings, indicating that chat-persona SFT does not override the base's tool-selection bias. The fingerprint reads what was trained into the weights for tool use, not the SFT brand. The singleton *Qwen3-Think* has no within-family ratio. Its reasoning-mode post-training on a shared base nonetheless produces a large shift: `qwen3-4b-thinking-2507` sits ∼14× farther from the other four Qwen3 checkpoints than they sit from each other, showing that reasoning-mode SFT substantially rewires the action policy and that AgentProv catches the drift.

4.3 Vulnerability to Hidden System Prompts

Text-channel tests measure surface tokens: MET's Hamming kernel compares completions character-by-character, and RUT's log-rank score is a function of literal token positions under the reference distribution. Any change to the input that shifts the output token stream, even by one character, moves both statistics. In practice, providers routinely modify exactly this surface: in a token-count scan of 75 OpenRouter endpoints, 44% deviate from the model's official chat template, including endpoints that inject hidden system prompts (Appendix A.5). This combination, a test that responds to any token-level change paired with a deployment surface the auditor cannot observe, makes text-channel identity tests vulnerable. We confirm the vulnerability under controlled conditions.

Experimental design. For each of 5 open-weight checkpoints (listed in Appendix Table 11) we keep weights, decoding, and chat template fixed and vary only the system message across three conditions: an *empty* message (a single whitespace, 1 token), a

short boilerplate ("You are a helpful assistant.", $\sim$5 tokens), and a *long* provider-style customer-service prompt ($\sim$50 tokens). Each condition is tested against the same-model no-system-role baseline. As the weights are identical, every rejection is by construction a false positive on identity.

Result. Table 3 reports per-condition decisions. MET falsely rejects 3/5 models even on the empty condition and 4/5 on the long one. RUT climbs from 1/5 to 5/5 across the same range. Adapted LLMmap rejects $0/5$, $1/5$, and $2/5$ models under the empty, short, and long conditions, respectively. AgentProv rejects 0/5 on empty and short and 1/5 on long, with the single rejection driven by Gemma-3-1B, whose tool-call format shifts under the longer persona-style prompt. Aggregated across all three conditions, AgentProv produces $1/15$ false positives (7%) versus $10/15$ (67%) for MET, $8/15$ (53%) for RUT, and $3/15$ (20%) for adapted LLMmap. The asymmetry confirms the prediction of §3.3: the delta kernel on categorical actions is insensitive to surface-text shifts that the Hamming and log-rank kernels respond to.

The text-channel vulnerability appears at the shortest prompt we test: a single whitespace already flips MET on 3/5 models. The effect is therefore not a corner case requiring crafted adversarial prompts. Any provider customization that adds a system role to the request can flip the decision on otherwise-identical samples. The long condition sits within the range observed in real provider deployments, and §4.4 shows the same false-positive pattern on real third-party endpoints.

4.4 API Provider Verification

We apply AgentProv, MET, and RUT to a practical deployment scenario: verifying whether a third-party API endpoint serves the model it claims. The audit compares samples from a *trusted reference* (local inference for open-weight models) against samples from a *suspect endpoint* on OpenRouter. Since the true serving configuration is unknown to the auditor, no decision can be validated against ground truth. We therefore construct a side-channel signal to contextualize each method's output: a token count signal that compares the provider-reported `prompt_tokens` against the auditor's locally computed count for the same messages (details in Appendix A.5). A persistent positive offset reveals content the provider injected but the auditor never specified, i.e., a hidden system prompt. This signal does not flag whether the model was substituted, but it does tell us whether the deployment context was modified, which §4.3 showed text-channel tests are vulnerable to. Of the 9 endpoints, 5 show extra tokens indicative of hidden prompts and 4 do not. Table 4 reports per-endpoint decisions. The key question is whether rejections on hidden-prompt endpoints reflect genuine model substitution or the deployment-context sensitivity.

Hidden-prompt endpoints. MET rejects all 5 endpoints flagged by the token-count scan. RUT rejects 4 of 5, accepting only the +7-token Gemma-3n-E4B. AgentProv accepts the four small-to-moderate injection cases (Llama-3.2-1B, Qwen3-14B, Llama-3-8B, each $\sim$1 extra token, and Gemma-3n-E4B at +7), consistent with the controlled finding of §4.3 that injected system prompts at this size shift surface tokens but not tool selection. On Llama-3.2-3B (+25 extra tokens) all three methods reject, suggesting an injection large enough to also move the action distribution.

Endpoints without detected hidden prompts. Among the four endpoints evaluated against matching local references whose token counts match the official template, the methods disagree in informative ways. On Llama-3.1-8B, MET and AgentProv accept while RUT rejects, possibly due to precision differences between the served endpoint and the local reference. On Mistral-Nemo, all three methods reject, indicating a difference visible in both surface tokens and tool selection. The remaining two endpoints split: on Gemma-3-4B, RUT and AgentProv accept while MET rejects, suggesting a surface-text shift visible to MET's Hamming kernel but not to the action distribution or the log-rank score. On Qwen3-8B, MET and RUT both reject while AgentProv accepts, suggesting a text-channel deviation that does not extend to tool selection.

Summary. Across the nine endpoints, AgentProv accepts 7, MET accepts 1, and RUT accepts 2. The pattern is consistent with AgentProv being insensitive to deployment-context modifications that shift surface tokens (hidden prompts, template variants) while remaining sensitive to changes in the action distribution (model substitution, large prompt injections). Absent ground truth on serving configurations, these interpretations remain correlational.

OpenAI reseller control. We additionally compare nine OpenAI models served through OpenRouter with their matching official OpenAI end-

	MET			RUT			Adapted LLMmap			AgentProv		
Model	empty	short	long	empty	short	long	empty	short	long	empty	short	long
`Llama-3.2-1B-Instruct`	✓	✓	×	✓	✓	×	✓	✓	×	✓	✓	✓
`Qwen3-1.7B`	×	×	×	✓	×	×	✓	✓	✓	✓	✓	✓
`Qwen2.5-1.5B-Instruct`	×	×	×	✓	✓	×	✓	✓	✓	✓	✓	✓
`Gemma-3-1B-it`	×	×	×	×	×	×	✓	×	×	✓	✓	×
`Hammer2.1-1.5B`	✓	✓	✓	✓	✓	×	✓	✓	✓	✓	✓	✓
FPR	3/5	3/5	4/5	1/5	2/5	5/5	0/5	1/5	2/5	**0/5**	**0/5**	**1/5**

Table 3: Per-model false-positive rate on the hidden-prompt control experiment (§4.3). Each cell reports the per-model decision at $\alpha = 0.05$ when the system prompt is replaced by the column variant; weights are unchanged, so every × is a false positive on identity. The *empty* condition is a single whitespace (1 token); *short* is "You are a helpful assistant." (∼5 tokens); *long* is a ∼50-token provider-style customer-service prompt. AgentProv uses the tool-selection probe at $K = 20$ templates, $N = 50$ samples per template.

Endpoint	MET	RUT	AgentProv
With detected hidden prompt			
Llama-3.2-3B	rej	rej	rej
Gemma-3n-E4B	rej	acc	acc
Llama-3.2-1B	rej	rej	acc
Qwen3-14B	rej	rej	acc
Llama-3-8B	rej	rej	acc
No detected hidden prompt			
Llama-3.1-8B	acc	rej	acc
Qwen3-8B	rej	rej	acc
Gemma-3-4B	rej	acc	acc
Mistral-Nemo-Instruct-2407	rej	rej	rej

Table 4: Third-party API verification on 9 OpenRouter endpoints (§4.4), grouped by whether the OpenRouter token-count scan (Appendix A.5) detected a provider-injected hidden system prompt. Across these endpoints, AgentProv accepts 7, MET accepts 1, and RUT accepts 2. Cells report decisions at $\alpha = 0.05$. AgentProv uses the tool-selection probe with $N = 50$ samples per template; MET and RUT consume the same matched-provider API responses.

	N samples per template			
K templates	5	10	25	50
5	36.67%	66.67%	92.70%	97.46%
10	60.48%	91.90%	98.89%	99.84%
20	65.56%	94.13%	99.84%	100.00%
30	64.44%	95.24%	100.00%	100.00%
40	64.76%	95.71%	100.00%	100.00%
60	68.57%	96.03%	100.00%	100.00%

Table 5: Template (K) × sample-count (N) ablation for AgentProv (tool-selection probe, $B = 1{,}000$, $\alpha = 0.05$). Cells report the rejection rate (accuracy) on the 630 distinct-checkpoint pairs of the 36-model set (§4.2). Accuracy generally increases along both axes, with small finite-sample fluctuations at low N, and saturates along a frontier: the $K = 20$, $N = 50$ configuration reaches 100%, as does $K \geq 30$ at $N \geq 25$. At fixed small N, adding templates beyond $K = 20$ yields only marginal gains (e.g. $N = 5$ rises from 65.6% at $K = 20$ to 68.6% at $K = 60$); more samples per template is the more effective margin.

points. AgentProv accepts all $9/9$ route-matched pairs. Appendix Table 12 reports this result together with the controlled separation checks.

4.5 Ablations and Efficiency

We characterize AgentProv's behavior as a function of the two budget parameters: the number of templates K and the number of samples per template N. Table 5 reports a joint $K \times N$ sweep on the 630 checkpoint pairs of the 36-model set.

Joint saturation. The rejection rate on distinct-checkpoint pairs generally increases along both axes, with small finite-sample fluctuations at low N. Holding K fixed and increasing N tightens the per-template categorical estimate, lifting the rejection rate until it saturates at the true separation between the two endpoints. Holding N fixed and increasing K adds independent template blocks to the MMD statistic, with a similar effect. The two effects compose: a small budget can be spent either as more templates with fewer samples or fewer templates with more samples. The $K = 20$, $N = 50$ configuration reaches 100% on the 36-model set and matches MET and RUT on identity discrimination. Type-I error remains controlled across the entire grid: only 1 of 36 models is falsely rejected at a single cell (K=10, N=10), with all other cells at $0/36$ (Appendix Table 13).

Budget estimate. A full audit at $K = 20$, $N = 50$ costs 1,000 queries per endpoint. A lighter $K = 10$, $N = 50$ operating point costs 500 queries and still reaches 99.84%. At typical API pricing both points are well under \$1 per endpoint.

4.6 Tool-Name and Schema Robustness

We test whether AgentProv depends on tool-schema wording on 12 local and frontier models

listed in Appendix A.7. Holding each probe otherwise fixed, we either replace every tool name with a random neutral string or paraphrase every tool and parameter description. All three probe surfaces accept all $12/12$ self-comparisons and reject all $66/66$ distinct-model pairs (Table 6), showing that the decisions do not depend on the original tool names or wording in this experiment.

Probe surface	Self (acc)	Distinct (rej)
Original schema	$12/12$	$66/66$
Random neutral tool names	$12/12$	$66/66$
Paraphrased tool/parameter descriptions	$12/12$	$66/66$

Table 6: Tool-name and schema robustness.

4.7 Robustness to Adaptive Providers

We evaluate three adaptive evasion strategies: prompt-based mimicry, LoRA fine-tuning, and probabilistic substitution. We also discuss exact per-probe replay by a fully adaptive provider that recognizes audit probes.

Prompt-based mimicry. We wrap each substitute in adversarial system prompts with four knowledge levels: none, claimed identity, the reference's aggregate tool-call profile, and few-shot reference demonstrations from disjoint probes. AgentProv rejects both substitutes under every condition (Table 7), showing that prompt-level mimicry does not reproduce the reference tool-selection behavior.

Prompt knowledge	GPT-4o-mini $\rightarrow$ GPT-4o	GPT-5.4-mini $\rightarrow$ GPT-5.4
No mimicry	rej, 6.4×10^{-5}	rej, 5.8×10^{-5}
Identity	rej, 6.4×10^{-5}	rej, 5.7×10^{-5}
+Profile	rej, 7.5×10^{-5}	rej, 8.7×10^{-5}
+Few-shot	rej, 8.2×10^{-5}	rej, 1.05×10^{-4}

Table 7: Prompt-based mimicry.

Fine-tuning mimicry. Following fingerprint-spoofing attacks using parameter-efficient fine-tuning (Zhang et al., 2026a), we LoRA-distill Qwen2.5-1.5B on Qwen2.5-7B tool-call outputs using 48 probe templates and test 12 disjoint templates. The substitute evades detection on training probes but is rejected on held-out probes (Table 8), so the learned mimicry is probe-specific and does not transfer to unseen probes.

Probe set	Decision	p-value
Training	acc	1.00
Held-out	rej	0.013

Table 8: LoRA fine-tuning mimicry.

Probabilistic substitution. Following randomized and rerouted substitution threats (Cai et al., 2025; Zhu et al., 2026), we serve the reference with probability q and the substitute with probability $1-q$. The discrepancy decreases with q, but evasion requires serving the genuine reference for a substantial fraction of requests (Table 9), limiting the benefit of substitution.

q	GPT-4o-mini $\rightarrow$ GPT-4o	GPT-5.4-mini $\rightarrow$ GPT-5.4
0.0	rej, 6.7×10^{-5}	rej, 6.0×10^{-5}
0.2	rej, 4.5×10^{-5}	rej, 4.1×10^{-5}
0.3	rej, 3.6×10^{-5}	rej, 3.1×10^{-5}
0.4	rej, 2.9×10^{-5}	acc, 2.4×10^{-5}
0.5	acc, 2.5×10^{-5}	acc, 2.0×10^{-5}
0.7	acc, 1.2×10^{-5}	acc, 8.4×10^{-6}
1.0	acc, 7.5×10^{-6}	acc, 3.5×10^{-6}

Table 9: Probabilistic substitution. q is the fraction of requests served by the genuine reference.

Fully adaptive replay. Exact per-probe replay is undetectable on probes to which the provider has adapted, but requires recognizing audit probes and holding the reference's per-probe distributions. We mitigate both requirements with a separately scored private probe subset. The provider cannot pre-adapt to unseen probes and therefore falls back to the substitute, which AgentProv detects. Because the probes are ordinary tool-use tasks, auditors can generate and rotate fresh private probes at negligible cost, limiting the durability of adaptation.

5 Conclusion

We introduced AgentProv, a two-sample equality test that audits agentic LLM APIs by comparing tool-selection behavior rather than natural-language completions. The resulting categorical distribution reflects the model's post-training and learned parameters, providing a practical identity signal. Across the evaluated models, AgentProv distinguishes genuine model changes while being more robust than text-channel baselines to the deployment-context changes tested. AgentProv offers a practical, lightweight auditing method for agentic APIs, including action-only serving stacks.

Limitations

AgentProv requires reference access to an authoritative instance of the claimed model: for fully closed models without any trusted reference channel, the test is inapplicable. As with other black-box methods, AgentProv cannot prove the underlying backbone without access to model weights. It provides strong evidence of identity from the probed tool-selection behavior, not a complete identity certificate. Within this scope, the signal is robust to the benign serving changes tested, sensitive to model substitution, and resistant to imitation on unseen probes. Finally, the third-party API analyses cross-check the methods' decisions against an independent token-count side-channel but do not establish ground truth on serving configuration, since providers do not disclose backbone identity or system-prompt content.

Ethical Considerations

AgentProv is intended to support transparency and accountability in commercial LLM deployments. A rejection at $\alpha = 0.05$ indicates a statistically significant difference in tool-selection behavior, not proof of model substitution or provider misconduct, because serving configuration can also affect this behavior (§4.3). Auditors should therefore interpret results alongside documented serving configurations, per-template analyses (Eq. 5), and corroborating side-channels (Appendix A.5), and should respect endpoint terms of service and rate limits. We will release the probe templates and source code to enable independent replication.

References

Anthropic. 2024. Introducing the Model Context Protocol. https://www.anthropic.com/news/model-context-protocol.

Anthropic. 2026. Define tools. Claude Platform Documentation.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. 2025. τ^2-bench: Evaluating conversational agents in a dual-control environment. *Preprint*, arXiv:2506.07982.

Thierry Blankenstein, Jialin Yu, Zixuan Li, Vassilis Plachouras, Sunando Sengupta, Philip Torr, Yarin Gal, Alasdair Paren, and Adel Bibi. 2026. Biasbusters: Uncovering and mitigating tool selection bias in large language models. In *The Fourteenth International Conference on Learning Representations*.

Will Cai, Tianneng Shi, Xuandong Zhao, and Dawn Song. 2025. Are you getting what you pay for? auditing model substitution in LLM apis. *CoRR*, abs/2504.04715.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. 2026. Retool: Reinforcement learning for strategic tool use in LLMs. In *The Fourteenth International Conference on Learning Representations*.

Irena Gao, Percy Liang, and Carlos Guestrin. 2025. Model equality testing: Which model is this API serving? In *The Thirteenth International Conference on Learning Representations*.

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773.

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

Dmitri Iourovitski, Sanat Sharma, and Rakshak Talwar. 2024. Hide and seek: Fingerprinting large language models with evolutionary learning. *Preprint*, arXiv:2408.02871.

Heng Jin, Chaoyu Zhang, Shanghao Shi, Wenjing Lou, and Y. Thomas Hou. 2024. Proflingo: A fingerprinting-based intellectual property protection scheme for large language models. *Preprint*, arXiv:2405.02466.

Hope McGovern, Rickard Stureborg, Yoshi Suhara, and Dimitris Alikaniotis. 2025. Your large language models are leaving fingerprints. In *Proceedings of the 1stWorkshop on GenAI Content Detection (GenAIDetect)*, pages 85–95, Abu Dhabi, UAE. International Conference on Computational Linguistics.

Anshul Nasery, Jonathan Hayase, Creston Brooks, Peiyao Sheng, Himanshu Tyagi, Pramod Viswanath, and Sewoong Oh. 2025. Scalable fingerprinting of large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

OpenAI. 2026. Function calling. OpenAI API Documentation.

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, Seattle, WA. USENIX Association.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. 2025. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2024. Gorilla: Large language model connected with massive APIs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Zhenzhen Ren, GuoBiao Li, Sheng Li, Zhenxing Qian, and Xinpeng Zhang. 2025. Cotsrf: Utilize chain of thought as stealthy and robust fingerprint of large language models. *Preprint*, arXiv:2505.16785.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Shuo Shao, Yiming Li, Hongwei Yao, Yiling He, Zhan Qin, and Kui Ren. 2025. Explanation as a watermark: Towards harmless and multi-bit model ownership verification via watermarking feature attribution. In *Network and Distributed System Security Symposium*.

Shuo Shao, Haozhe Zhu, Yiming Li, Hongwei Yao, Tianwei Zhang, and Zhan Qin. 2026. Fitprint: Towards false-claim-resistant model ownership verification via targeted fingerprint. *Preprint*, arXiv:2501.15509.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Kimi Team, Yifan Bai, Yiping Bao, Y. Charles, Cheng Chen, Guanduo Chen, Haiting Chen, Huarong Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, Zhuofu Chen, Jialei Cui, Hao Ding, Mengnan Dong, Angang Du, and 181 others. 2026. Kimi k2: Open agentic intelligence. *Preprint*, arXiv:2507.20534.

Yun-Yun Tsai, Jia Hao Liang, Chuan Guo, Junfeng Yang, and Laurens van der Maaten. 2026. Rafp: Identifying llm lineages via rare-region fingerprints. *Preprint*, arXiv:2505.12682.

Liwen Wang, Zongjie Li, Yuchong Xie, Shuai Wang, Dongdong She, Wei Wang, and Juergen Rahmel. 2026a. On protecting agentic systems' intellectual property via watermarking. *Preprint*, arXiv:2602.08401.

Zhebo Wang, Zhenhua Xu, Maike Li, Wenpeng Xing, Chunqiang Hu, Chen Zhi, and Meng Han. 2026b. Sraf: Stealthy and robust adversarial fingerprint for copyright verification of large language models. *Preprint*, arXiv:2505.06304.

Jiashu Xu, Fei Wang, Mingyu Ma, Pang Wei Koh, Chaowei Xiao, and Muhao Chen. 2024. Instructional fingerprinting of large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3277–3306, Mexico City, Mexico. Association for Computational Linguistics.

Shojiro Yamabe, Futa Kai Waseda, Tsubasa Takahashi, and Koki Wataoka. 2025. MergePrint: Merge-resistant fingerprints for robust black-box ownership verification of large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6894–6916, Vienna, Austria. Association for Computational Linguistics.

Yuliang Yan, Haochun Tang, Shuo Yan, and Enyan Dai. 2026. DuFFin: A dual-level fingerprinting framework for LLMs IP protection. In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 5168–5184, Rabat, Morocco. Association for Computational Linguistics.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Zhiguang Yang and Hanzhou Wu. 2026. Unveiling hidden model fingerprints in API-protected LLMs. *Electronic Imaging*, 38(4):309–1–309–9.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Jiahao Zhang, Xiuyu Li, and Suhang Wang. 2026a. Your "pro" LLM subscription may actually be "free": Exposing fingerprint spoofing risks in LLM inference services. *CoRR*, abs/2606.16100.

Yage Zhang, Yukun Jiang, Zeyuan Chen, Michael Backes, Xinyue Shen, and Yang Zhang. 2026b. Real money, fake models: Deceptive model claims in shadow apis. *Preprint*, arXiv:2603.01919.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. 2024. Wildchat: 1m chatGPT interaction logs in the wild. In *The Twelfth International Conference on Learning Representations*.

Xiaoyuan Zhu, Yaowen Ye, Tianyi Alex Qiu, Hanlin Zhu, Sijun Tan, Ajraf Mannan, Jonathan Michala, Raluca Ada Popa, and Willie Neiswanger. 2026. Auditing black-box LLM APIs with a rank-based uniformity test. In *The Fourteenth International Conference on Learning Representations*.

A Technical Appendices and Supplementary Material

A.1 Method Details

Action bucketing. The auditor classifies each sample by mapping the emitted tool-call name to an option label via a per-template `action_map`. Responses that answer directly without a tool call are bucketed into `_no_call`; responses whose tool-call syntax fails to parse are bucketed into `_malformed`. Both labels are included in every option set $\mathcal{O}_k$ and are never silently discarded. Because classification reads the structured `tool_calls[]` field rather than free-form text, bucketing is deterministic and identical across endpoints, including action-only serving stacks where `message.content` is empty.

Confound controls. Three mandatory controls apply to every template. First, tool order in the schema is randomized per sample with a deterministic seed, eliminating first-listed bias. Second, tool descriptions within a template are length-matched and structurally parallel, preventing detail-driven selection. Third, prompts use neutral recipients (`@example.com`, generic names) and tool-agnostic task wording ("send a note to" rather than "email") to avoid domain priming.

A.2 Probe Examples and Template Sourcing

Template domains. The K templates span two complementary domains. *General tools* cover everyday operations (weather lookup, calendar management, email dispatch, customer-info retrieval) where tool names are drawn from real provider documentation (OpenAI, Anthropic, Google) and public benchmarks, creating synthetic redundancy across 15 task categories. *MCP servers* present the model with equivalent operations from different MCP server families (e.g., `mcp__calendar__add_event` vs. `mcp__google_calendar__create_event` vs. `mcp__outlook_calendar__create_meeting`), leveraging the standardized MCP protocol surface (Anthropic, 2024). Using tool names and parameter shapes from real SFT and RL data mixtures is essential: the probe reads the pre-trained tool-selection prior, not an arbitrary classification task. Table 10 shows one representative template from each domain.

A.3 Model List

A.4 Full Pairwise Heatmap

Figure 4 shows the full 36×36 pairwise MMD^2 matrix on the 36-model set (§4.2), grouped by family. All 630 off-diagonal pairs reject at $\alpha = 0.05$. Beyond the binary decision, the statistic itself reveals structure: same-family models cluster (Qwen2.5 sizes against each other, Llama-3 sizes against each other, Hammer-2.1 sizes against each other), visible as darker diagonal blocks. Shared training lineage also carries across architectures (R1-Distill variants distilled from DeepSeek-R1 cluster together regardless of base). Families assembled from heterogeneous bases (Hermes-3 SFT applied to three unrelated bases) form the exception: intra-family distances are no smaller than cross-family, reflecting that surface SFT does not override the base's tool-selection prior.

A.5 OpenRouter Token-Count Side-Channel

The token-count side-channel referenced in §4.3 and §4.4 detects provider-injected hidden system prompts by comparing the provider's reported `prompt_tokens` against the canonical `apply_chat_template` count for the messages the auditor sent. A persistent positive offset indicates content the provider added that the auditor never specified.

Endpoint coverage. OpenRouter exposes 358 endpoints via `GET /v1/models`; 127 map to a HuggingFace tokenizer release via family rules and per-endpoint overrides. After tokenizer-load and chat-template verification, 84 endpoints remain *mappable*: the auditor can independently compute their canonical chat-template token count. Of these, 75 open-weight chat endpoints constitute the scan corpus.

Bare token count. The reference is the official chat template applied to the auditor's messages, after stripping auto-injected blocks (e.g., Llama-3.1+ "Cutting Knowledge Date / Today Date") that the local template inserts but that providers may legitimately omit. A nonzero `api_count` − `bare_count` therefore reflects content the provider added or removed beyond the auditor's messages, not differences in template chrome handling.

Probes. For each endpoint the auditor sends four messages at temperature 0, three calls per message: an empty user message, a single-character user message, an empty system + single-character user

Domain	Probe setup	Option set
General tools	"Schedule a team sync for Thursday at 2pm." Tools: `create_event`, `schedule_meeting`, `add_event`, `book_calendar_event` (identical descriptions).	{tool names, `_no_call`, `_malformed`, `_other`}
MCP servers	"Send a short greeting to alice@example.com." Tools: `mcp__gmail__send_email`, `mcp__outlook__send_email`, `mcp__smtp__send_mail` (identical descriptions).	{tool names, `_no_call`, `_malformed`, `_other`}

Table 10: Two representative templates of the tool-selection probe, showing the probe setup and the option set into which the model's response is classified. The first uses general-tool naming variants; the second uses MCP server naming variants. The full template pool contains $K = 60$ such templates spanning both domains.

Family	Members evaluated
Qwen2.5	0.5B, 1.5B, 3B, 7B; DistilQwen2.5-3B
Qwen3	0.6B, 1.7B, 4B, 4B-Instruct-2507
Qwen3-Think	4B-Thinking-2507
Llama-3.1 / 3.2	Llama-3.1-8B; Llama-3.2-1B, 3.2-3B; SuperNova-Lite
Mistral	Mistral-7B-Instruct, Mistral-Nemo, Ministral-8B
DeepSeek base	DeepSeek-LLM-7B, DeepSeek-Coder-7B
DeepSeek-R1 distill	R1-Distill-Qwen-1.5B, 7B, 14B; R1-Distill-Llama-8B
Hermes (NousResearch)	Hermes-3-Llama-3.1-8B, Hermes-3-Llama-3.2-3B, Hermes-2-Pro-Mistral-7B
Hammer-2.1 (MadeAgents)	0.5B, 1.5B, 3B, 7B
Phi	Phi-4-mini-reasoning
Gemma-3	Gemma-3-1B
GLM	GLM-Z1-9B
Other	Nemotron-Nano-4B, Gorilla-OpenFunctions-v2, SmolLM3-3B
OpenAI (closed)	GPT-3.5-turbo, GPT-4.1-mini, GPT-4o-mini, GPT-4o, o4-mini GPT-5.4-mini, GPT-5.4, GPT-5.6-luna, GPT-5.6-sol

Table 11: The 36 open-weight checkpoints and nine closed-source OpenAI models evaluated. The five small checkpoints revisited under injected system prompts in §4.3 are a subset (Llama-3.2-1B, Qwen3-1.7B, Qwen2.5-1.5B, Gemma-3-1B, Hammer2.1-1.5B). Third-party API endpoints audited in §4.4 are listed in Table 4.

message, and a 20-token continuation prompt. The headline 44% figure uses the realistic prompts only (single-character user and 20-token continuation), since edge-case prompts expose provider quirks orthogonal to deployment-context modification.

Classification. For each prompt p, let d_p denote the difference between the API-reported prompt-token count and the locally-computed bare count, and let v_p denote the call-to-call variance of the API-reported count. Endpoints are classified as follows: `no_drift` if all $d_p = 0$ and $v_p = 0$; `const_offset_positive` if all $d_p = k > 0$ with $v_p = 0$, indicating a consistent hidden-content injection of size k; `const_offset_negative` if $d_p = k < 0$, indicating a leaner provider template; `variable` if $v_p > 0$, indicating provider non-determinism (e.g., load-balanced deployments); and `systematic` for all other deterministic nonzero patterns.

Identified hidden-prompt endpoints. Nine `const_offset_positive` endpoints account for the headline injections, sorted by size: Llama-3.2-3B-Instruct (+25), Mixtral-8x22B-Instruct (+12), Gemma-3n-e4b-it (+7), Hunyuan-a13B-Instruct (+6), and five endpoints with +1 token (Llama-3-70B-Instruct, Llama-3-8B-Instruct, Llama-3.2-1B-Instruct, Qwen3-14B, GLM-4.5v). One further endpoint (Llama-Guard-4-12B) falls into `variable` with positive mean offset, consistent with hidden content injected on a subset of calls.

A.6 OpenAI Reseller Control

We compare the nine closed-source OpenAI models from §4.2 through two routes: OpenAI's official API and OpenRouter. Using the official endpoint as the reference, AgentProv accepts all $9/9$ matching OpenRouter routes. This live-endpoint comparison is supporting rather than definitive evidence of faithful resale because OpenRouter's internal serving configuration remains unobservable. The controlled checks reject all $36/36$ distinct OpenAI pairs and all $324/324$ OpenAI–local-checkpoint pairs.

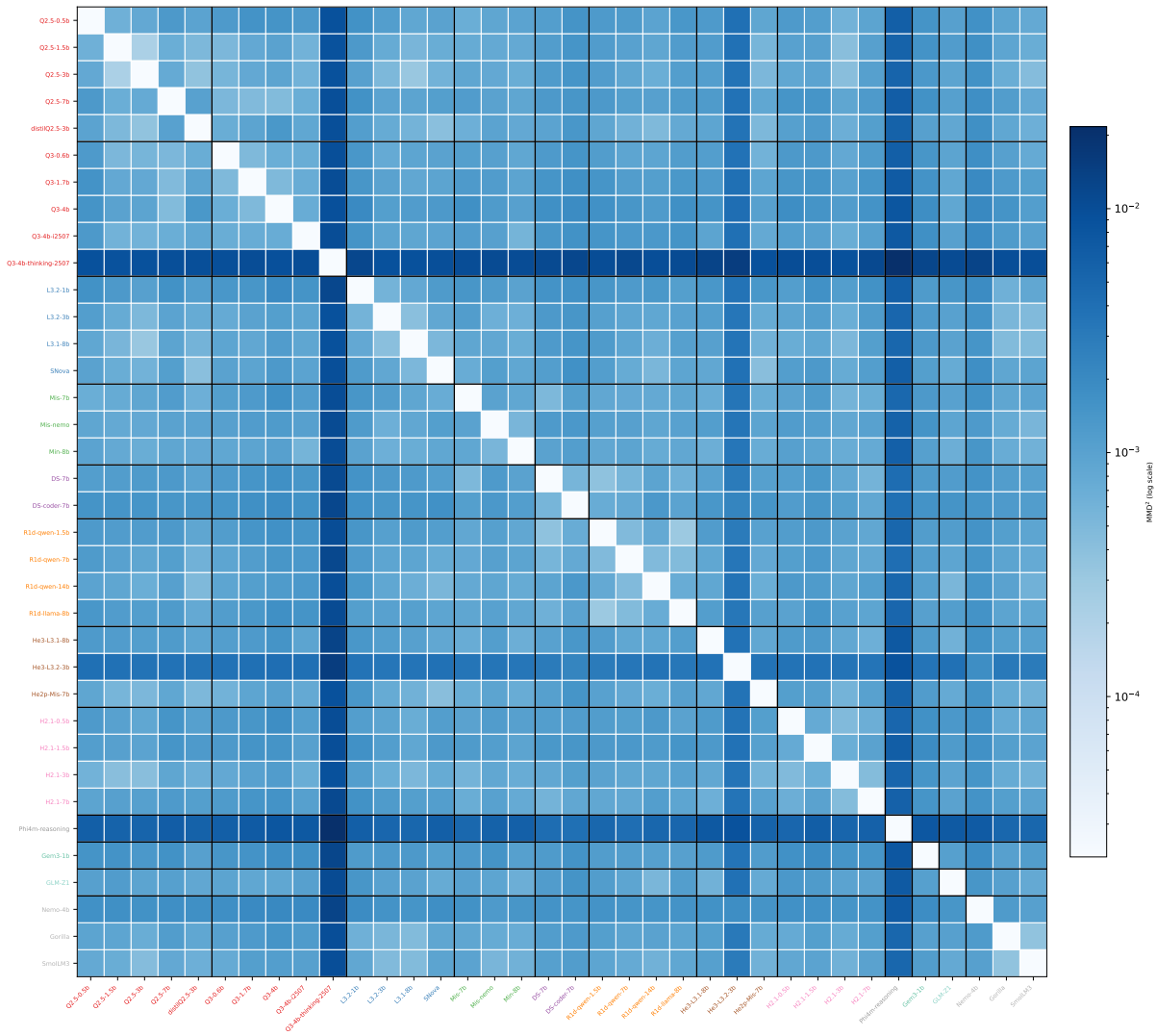

Figure 4: Full 36×36 pairwise MMD^2 matrix on the AgentProv $\cap$ MET $\cap$ RUT common set (§4.2), computed with AgentProv's tool-selection probe at $K = 20$ templates and $N = 50$ samples per template. Models are grouped by family; black lines mark family boundaries. All 630 distinct-checkpoint pairs reject at $\alpha = 0.05$.

Comparison	Pairs	Decision
Matching model: official OpenAI vs. OpenRouter	9	9/9 acc
Distinct official OpenAI models	36	36/36 rej
Official OpenAI models vs. local checkpoints	324	324/324 rej

Table 12: Closed-model reseller control and controlled separation checks. The official OpenAI endpoint is the trusted reference for the route-matched comparisons.

A.7 Ablation Details

Table 5 in the main text reports the cross-pair rejection rate for the $K \times N$ sweep. Table 13 complements it with the corresponding Type-I error (self-test false-positive rate) on the same grid.

	N samples per template			
K templates	5	10	25	50
5	0/36	0/36	0/36	0/36
10	0/36	1/36	0/36	0/36
20	0/36	0/36	0/36	0/36
30	0/36	0/36	0/36	0/36
40	0/36	0/36	0/36	0/36
60	0/36	0/36	0/36	0/36

Table 13: Self-test Type-I error (split-sample FPR) on the 36-model set, for the same $K \times N$ grid as Table 5. Each cell reports the number of models for which the test rejects when both halves come from the same model (any rejection is a false positive on identity). The test is conservative: FPR is at or below the nominal $\alpha = 0.05$ rate at every cell, with the single false positive at K=10, N=10 within the expected rate by chance across the 24-cell grid.

Robustness model set. The tool-name and schema robustness experiments in §4.6 use Qwen2.5-7B, Llama-3.1-8B, Mistral-7B, DeepSeek-7B, Hammer2.1-7B, GPT-3.5-turbo, GPT-4.1-mini, GPT-4o-mini, GPT-4o, GPT-5.4-mini, GPT-5.4, and o4-mini.

A.8 Additional Robustness Experiments

Quantization. We compare full-precision references with quantized deployments of Qwen2.5-7B and Llama-3.1-8B served through vLLM. Table 14 shows that all three methods accept both 8-bit deployments. At 4 bits, AgentProv accepts Qwen2.5-7B but rejects Llama-3.1-8B, while MET and RUT reject both. The 4-bit outcome is therefore mixed and model-dependent rather than uniformly invariant.

Quantized deployment	AgentProv	MET	RUT
Qwen2.5-7B, 8-bit	acc	acc	acc
Llama-3.1-8B, 8-bit	acc	acc	acc
Qwen2.5-7B, 4-bit	acc	rej	rej
Llama-3.1-8B, 4-bit	rej	rej	rej

Table 14: Decisions for quantized deployments against their full-precision references.

Safety policy and agent wrapper. We prepend two serving-level system prompts to the probes: a helpful, harmless, and honest safety preamble, and a production agent-framework protocol that requests one well-formed call from a fixed tool registry. We test Llama-3.2-1B, Qwen3-1.7B, Qwen2.5-1.5B, Hammer2.1-1.5B, Qwen2.5-7B, and Llama-3.1-8B against their clean fingerprints. Because the weights are unchanged, every rejection is a false positive. Table 15 shows that AgentProv has no false positives under the safety policy and two under the agent wrapper, compared with four for MET and six for RUT under either condition. The two wrapper failures show that wrapper logic can still alter the probed tool-selection behavior for some models.

System-prompt condition	AgentProv	MET	RUT
Safety policy	$0/6$	$4/6$	$6/6$
Agent wrapper	$2/6$	$4/6$	$6/6$

Table 15: False positives under safety-policy and agent-wrapper system prompts.

A.9 Baseline and Protocol Details

AgentProv (this work). Each model is queried at temperature 0.7 via its native chat template; tool schemas are shuffled per sample with a deterministic seed. We use $K = 20$ templates of the tool-selection probe and $N = 50$ samples per template, yielding $1{,}000$ samples per endpoint. Tool-call names are bucketed via per-template `action_maps` (§A.1); `_no_call` and `_malformed` are valid options. For each pair of endpoints we compute the observed test statistic T (Eq. 4) on the per-template sample matrices and calibrate the permutation null with $B = 1{,}000$ row-level shuffles (Algorithm 1). Decisions are exact permutation p-values at level $\alpha = 0.05$.

MET (Gao et al., 2025). Following the upstream implementation, each model is queried on 25 Wikipedia continuation prompts (5 prompts $\times$ 5 languages: English, German, French, Russian, Spanish) at temperature 1.0, top-p 1.0, generating $N = 50$ completions per prompt with a maximum of 50 output tokens. Completions are padded to a fixed character length and compared with the Hamming kernel on byte-level token strings; the test statistic is the Maximum Mean Discrepancy (Gretton et al., 2012) between the reference and suspect sample sets, with a permutation null at $B = 500$ resamples. Decisions are at $\alpha = 0.05$. The MET pipeline is run unmodified through our SLURM driver; the only added code is a thin OpenAI/OpenRouter sampling hook that the upstream did not implement.

RUT (Zhu et al., 2026). RUT is a grey-box test that requires log-probabilities from the reference. The protocol follows the paper: 100 English prompts sampled from the WildChat-1M dataset (Zhao et al., 2024) at temperature 0.5, top-p 1.0, maximum 30 output tokens; for each prompt the auditor draws 1 target completion and $m = 100$ reference completions. Every response token is scored under the reference distribution with the log-rank score (the empirical best of the five score functions evaluated in the paper). Per prompt, the randomized rank of the target's score within the reference scores is computed, and a Cramér–von Mises goodness-of-fit test against $U(0, 1)$ decides at $\alpha = 0.05$.

B Potential Risks

AgentProv is an accountability tool intended to help API customers, integrators, and third-party auditors verify whether a deployed endpoint serves the model it claims. A reject decision at $\alpha = 0.05$ is a statistical statement, not a finding of fraud: Section 4.3 shows that even small provider-side configuration changes can shift action distributions enough to trigger rejection (e.g., Gemma-3-1B under a 50-token persona prompt), so public attribution of an endpoint as "substituted" on the basis of a single audit run is inappropriate. AgentProv should be one input to a broader investigation that also considers documented provider configuration, the per-template breakdown from Eq. 5, and corroborating side-channels (e.g., the token-count probe of Appendix A.5). Because AgentProv can be run against any HTTP-accessible tool-call endpoint without provider cooperation, auditors should respect endpoint terms of service and rate limits; the 1,000-sample-per-endpoint configuration, which costs under $1 at typical API pricing, was chosen partly with this in mind. We will release the probe templates and source code to enable independent replication.