API-TAIL: Detecting Model Substitution in LLM APIs through Temperature-Aligned Likelihood Tails

TokenRhythm Technologies

arXiv:2610.36801v2 [cs.LG] 28 Oct 2026

Abstract

Commercial language model APIs return only text, allowing a provider to cut costs by quietly routing a fraction of user requests to a cheaper substitute model while billing for the advertised one. Detecting such partial substitution is challenging because honest providers also alter the output distribution through undocumented decoding and serving choices—such as temperature scaling, sampling truncation, repetition and presence penalties, and default system prompts—forcing an audit to test against a composite null of legitimate configurations rather than a single reference. Our key insight is that these legitimate variations primarily shift response typicality across the entire batch, whereas partial substitution injects individual responses that remain anomalous under the claimed model. Building on this, we introduce API-TAIL, a typicality audit that first absorbs batch-level shifts by aligning a single effective temperature to center standardized surprisal scores across the audited batch, and then flags the endpoint if the number of remaining tail responses exceeds a threshold calibrated across legitimate configurations. Evaluated on real-world API responses and constructed substitution mixtures, API-TAIL controls false alarms across diverse honest providers and detects low-rate partial substitution with a small query budget. Under a common composite calibration, it outperforms the adapted rank- and MMD-based audit baselines.

Project Page: https://github.com/TokenRhythm/API-TAIL

1. Introduction

Commercial language model APIs allow users to select and pay for a specific model by name while returning only generated text. A provider can therefore cut cost by answering a fraction ρ of the requests with a cheaper substitute, such as an older release, a smaller or mixture-of-experts sibling, or a model of another family, while billing for the advertised model. Such *partial substitution* is invisible in any single response, and auditing APIs for it has become an active problem [13, 5, 37, 6, 33, 35, 21].

The central obstacle is that honest providers do not serve a model in one canonical way. They choose decoding temperatures and truncation thresholds, apply repetition or presence penalties, and add default system prompts, usually without documenting these choices. Each choice shifts the distribution of the returned text, so an honest endpoint is not a sample from one reference distribution but from one of many legitimate deployments: the audit has to test a *composite* null hypothesis. Calibration against a single reference distribution can increase false alarms under legitimate deployment variation, while recalibration over a broader null can reduce power. On the controlled short-response panels, RUT [37] and MET [13] baselines calibrated on the

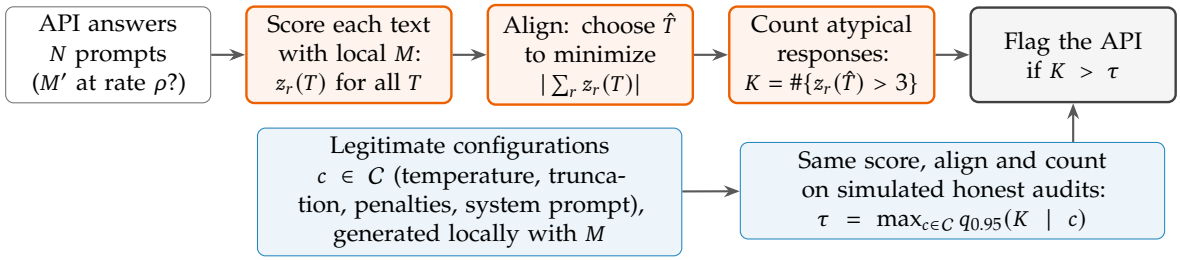


Figure 1 | API-TAIL audits an API from returned text. Each response is scored under the claimed model M; one effective temperature is aligned to the audited batch to absorb batch-level shifts from honest decoding and serving choices; the number of responses that remain atypical is compared with a threshold calibrated by running the same procedure on locally generated responses of M across legitimate deployment configurations.

same legitimate configurations as API-TAIL have zero power against 10% substitution with 200 API requests. These comparisons use method-specific sampling: RUT needs locally generated reference responses for each prompt, whereas matched-budget MET also needs repeated API responses per prompt (Appendix B).

Our approach starts from how the two kinds of change act on the text when every returned response is scored under the claimed model. Legitimate decoding and serving choices—whether temperature scaling, truncation, repetition and presence penalties, or default system prompts—shift how surprising the honest responses are for the claimed model, and they shift the whole batch together. A substitute model instead writes individual responses that stay atypical for the claimed model across temperatures. This suggests absorbing the batch-level shift first and testing what remains against a threshold that accounts for residual configuration differences.

API-TAIL implements this idea in three steps (Figure 1). It scores each response by its standardized excess surprisal: how much more surprising the response is under the claimed model at temperature T than a typical sample, in units of its standard deviation. It *aligns* one effective temperature to the audited batch by choosing the T at which these scores are centered, thereby absorbing batch-level shifts from undocumented decoding and serving choices. It then counts the responses whose score stays above 3 and compares the count with a threshold calibrated on locally generated responses of the claimed model across a pool of legitimate configurations, including the penalty configurations that providers use. Scores are computed exactly over the full vocabulary, so an audit is deterministic and reproducible.

We evaluate API-TAIL on real APIs: 26 plain honest endpoints serving Qwen3.8-27B or Gemma-4-31B-it through OpenRouter, two request protocols, and six substitute models served by seven endpoints, 143,955 responses in total. Our contributions are:

- **Evaluation setting.** We evaluate text-only API auditing against a composite null spanning legitimate decoding and serving configurations, with validity measured directly on real honest providers (Section 2).
- **Method.** We propose API-TAIL, which aligns an effective temperature by centering standardized typicality to absorb batch-level deployment shifts, counts the responses that remain atypical, and calibrates the count across a pool of legitimate decoding and serving configurations (Section 3). One audit of 200 requests needs one response per prompt, recorded API charges below 0.16 USD and two to eight GPU-minutes of deterministic local scoring.
- **Evidence.** API-TAIL keeps the false-alarm rate at or below 0.069 on all 26 plain honest endpoints and detects 10% substitution with power of at least 0.69 in 13 of 16 substitute settings

at 200 requests (Section 5). Matched ablations explain the design (Section 6): alignment versus the MLE control adds 0.09 to 0.63 power by keeping the composite threshold low, full-vocabulary scoring adds 0.62 power over a top-20 window on long responses, and the penalty configurations in the pool are what make the model developer's own endpoint pass.

2. Problem Setup

Access model. An API advertises a target model M (for example, Qwen3.8-27B) and bills each request as a call to M. The auditor sends prompts $x_1, \ldots, x_N$ and observes only the returned texts $y_1, \ldots, y_N$; it does not use log-probabilities, seeds or logit bias from the endpoint, and it does not assume that the endpoint honours sampling parameters. With access to the weights of M, the auditor can run M locally and compute teacher-forced next-token distributions for any returned text.

Partial substitution. A dishonest endpoint answers each request independently with a substitute model $M' \neq M$ with probability ρ and with M otherwise. Substitutes are typically cheaper: an older release of the same family, a smaller or mixture-of-experts sibling, or a model of another family. Small ρ is the hard and economically relevant regime: it saves the provider money while leaving most responses untouched.

Legitimate variation and the composite null. An honest endpoint serves M, but not necessarily the way the auditor would. Providers choose the decoding temperature, top-p and top-k cutoffs, presence and repetition penalties, default system prompts and inference backends [26], usually without documenting these choices. We therefore define honesty through a set C of *legitimate deployment configurations*: the null hypothesis is that there is a configuration $c \in C$ under which M generates the audited responses [3, 1]. This null is composite. An audit that compares the endpoint with a single reference deployment of M conflates legitimate deployment choices with substitution.

Validity and power. An audit maps N responses to a binary verdict. We require it to be valid on real traffic: its false-alarm rate, the probability of flagging an honest endpoint, must stay at most 0.10 for every honest provider endpoint in the evaluation, and this verdict must be stable when the calibration samples are re-drawn (we require it in at least 16 of 20 re-drawn calibrations). Among valid audits we compare *power*, the probability of flagging an endpoint that substitutes at rate ρ, at a fixed budget of N requests.

3. API-TAIL: Temperature-Aligned Typicality Auditing

API-TAIL has three parts (Figure 1): it scores every returned response by how typical it is under the claimed model at a given temperature, it aligns a single effective temperature to minimize the aggregate standardized score, and it counts the responses that remain atypical, comparing the count with a threshold calibrated across the legitimate configurations C to cover residual configuration differences.

3.1. Typicality under a tempered model

For a prompt x and a returned response $y = (y_1, \ldots, y_L)$, let $\ell_t \in \mathbb{R}^{|\mathcal{V}|}$ be the next-token logits of M at position t (teacher forcing), and let $q_{t,T} = \mathrm{softmax}(\ell_t/T)$ be the tempered distribution. With the per-token surprisal $s_t(T) = -\log q_{t,T}(y_t)$, its expectation $H_t(T)$ (the entropy of $q_{t,T}$) and its variance $\sigma_t^2(T)$ under $q_{t,T}$, we define the excess surprisal, its variance and the standardized typicality score of the response,

$$S(T) = \sum_{t=1}^{L} \big(s_t(T) - H_t(T)\big), \qquad V(T) = \sum_{t=1}^{L} \sigma_t^2(T), \qquad z(T) = \frac{S(T)}{\sqrt{V(T)}}. \tag{1}$$

When each next token is sampled from $q_{t,T}$, the summands $s_t(T) - H_t(T)$ form a martingale difference sequence with conditional variances $\sigma_t^2(T)$. Under suitable martingale central-limit conditions, $z(T)$ admits a standard-normal approximation. We calibrate the audit threshold empirically across legitimate configurations. A response from another model may contain tokens that M finds surprising in context, producing a large positive z.

3.2. Temperature alignment

An honest provider may alter the output distribution through temperature scaling, truncation, repetition or presence penalties, or a default system prompt, all of which shift the overall typicality of the batch; evaluating z at a fixed reference T confounds substitution with these deployment choices. API-TAIL fits one effective temperature to reduce the aggregate shift,

$$\hat{T} = \arg\min_{T \in \mathcal{T}} \left| \sum_{r=1}^{N} z_r(T) \right|, \qquad \mathcal{T} = \{0.30, 0.31, \ldots, 1.80\}, \tag{2}$$

selecting the grid temperature with the smallest absolute aggregate standardized score.

Alignment differs from fitting the temperature by maximum likelihood. Since $\frac{\partial}{\partial T} \log q_{t,T}(y_t) = \big(s_t(T) - H_t(T)\big)/T$, the likelihood equation for T is $\sum_r S_r(T) = \sum_r \sqrt{V_r(T)}\, z_r(T) = 0$, weighting each standardized response score by $\sqrt{V_r(T)}$. The MLE control minimizes $|\sum_r S_r(T)|$ over $\mathcal{T}$, whereas API-TAIL minimizes $|\sum_r z_r(T)|$, giving each response equal weight. Alignment therefore targets the same standardized scores used in the tail count, keeping null exceedances small for legitimate configurations that a pure temperature change does not describe, such as repetition penalties or a default system prompt. Section 6 shows that this is where the two estimators part ways.

3.3. Counting atypical responses against a composite null

After alignment, API-TAIL tests for an excess of large standardized scores [9] by counting upper-tail responses,

$$K = \#\{\, r \le N : \; z_r(\hat{T}) > 3 \,\}. \tag{3}$$

To calibrate K, the auditor generates responses of M locally under every configuration $c \in C$ on prompts disjoint from the audit prompts, simulates honest audits of N requests for each c, and records the 95th percentile $q_{0.95}(K \mid c)$, an integer. The threshold is the largest of these percentiles, $\tau = \max_{c \in C} q_{0.95}(K \mid c)$, and the endpoint is flagged when $K > \tau$. By construction, the false-alarm rate is at most 5% under every configuration in C (up to simulation error); the experiments test whether this carries over to real honest endpoints, whose configurations the auditor does not know.

3.4. Exact and reproducible scoring

API-TAIL scores the full vocabulary. For every position, one forward pass of M yields the logits, from which we compute float32 exponentials and products and accumulate the moment sums in float64: $\sum_v e^{\ell_{t,v}/T}$, $\sum_v e^{\ell_{t,v}/T}\ell_{t,v}$ and $\sum_v e^{\ell_{t,v}/T}\ell_{t,v}^2$ for all temperatures on the grid; these give s_t, H_t and σ_t^2 over the full vocabulary, without top-k or nucleus truncation. Scoring runs with batch size one and deterministic GPU kernels, so repeated audits of the same responses are bitwise identical (Appendix B). Auditing one endpoint with $N = 200$ responses takes about two GPU-minutes for 30-token responses and about eight for responses of up to 400 tokens on a single L20X GPU.

4. Experimental Setup

Claimed models and honest endpoints. We audit two widely served models offered by many independent providers through OpenRouter: Qwen3.8-27B and Gemma-4-31B-it. From provider metadata, documentation and a short parameter-compliance probe (Appendix A), we separate *plain honest* endpoints, which advertise BF16, FP16 or FP8 weights or state no precision, document no system-prompt injection and honour the requested temperature, from *special* endpoints, which serve FP4-class weights, document injecting their own system prompt, or ignore the temperature parameter. This yields 26 plain honest endpoints (17 for Qwen3.8-27B and 9 for Gemma-4-31B-it), on which validity is evaluated, and 10 special endpoints (4 and 6), analysed in Appendix A. One additional endpoint, G14, lacks a complete probe and is reported separately using its available P-user responses.

Request protocols and prompts. The main study uses two request protocols. *P30* fixes the sampling parameters (temperature 0.5, top-p 1.0, at most 30 tokens, reasoning disabled) and is the controlled setting; *P-user* sends requests the way a typical user does, with the provider's default sampling and responses of up to 400 tokens. Crossing the two models with the two protocols gives four panels. Main-study audit batches are drawn from a held-out pool of 1,000 test prompts (WildChat conversations [36], code and mathematics, in English and Chinese), disjoint from calibration prompts.

Substitutes and simulated audits. We collected real API responses from six substitute models served by seven endpoints: for Qwen3.8-27B, the previous same-family release Qwen3.6-27B (from a third-party provider, "provider A", and from the developer's own platform), the mixture-of-experts sibling Qwen3.6-35B-A3B, the smaller sibling Qwen3.8-Flash, and DeepSeek-V4.1-Flash; for Gemma-4-31B-it, the mixture-of-experts sibling Gemma-4-26B-A4B and two models of other families (DeepSeek-V4.1-Flash and Qwen3.8-27B). The main collection contains 143,955 responses (every source has at least 99% of the requested responses). The campaign ledger records 29.92 USD in API charges. An honest audit uses N API responses, with prompt allocation set by the method's sampling protocol; a substitution audit answers each request with the substitute's API response with probability ρ and otherwise with a response of the claimed model generated locally under the protocol's default configuration. We vary $\rho \in \{0.05, 0.10, 0.25\}$ and $N \in \{100, 200, 400\}$, and simulate 4,000 audits per honest endpoint and 1,000 audits per substitution setting. Honest-endpoint audits use real API responses and satisfy the false-alarm criterion in Section 5.

Legitimate configurations. The main-study pool C contains 16–19 configurations per panel, generated locally with M: default sampling (two seeds), temperatures 0.7 and 1.0 with top-p and top-k variants, the model's generation-config defaults, a default system prompt, presence penalties 1.5 and 2.0, repetition penalties 1.05 and 1.10, one combined penalty setting, and precision variants (Appendix A). Penalties belong in C because real providers apply them; an earlier profiling study of our own found that the model developer's endpoint applies a presence penalty of about 1.5. Each configuration contributes 2,000 simulated honest audits for calibration.

Baselines. Matched-budget comparisons use common source response banks, the same legitimate pool and the same maximum-per-configuration quantile calibration, with method-specific prompt allocation and reference sampling. We compare adapted implementations of MET [13], a maximum-mean-discrepancy test with a Hamming kernel, and RUT [37], a rank-uniformity test. Matched-budget MET uses two API and two local reference responses per prompt; RUT uses one API response and 100 local reference responses per prompt. These comparisons run on the P30 panels; Appendix B specifies the adaptations. A separate Qwen check uses native-style MET sampling: $T = 1$, 50 tokens, 10 API and 10 local responses per prompt, and an 11-configuration null pool. Calibration and evaluation independently draw trials from the same 99 eligible prompts. At $N = 200$, we test Qwen3.6-27B (provider A) and DeepSeek-V4.1-Flash substitutions. We include NLL, log-rank and entropy count controls inspired by text-detection statistics [14, 24, 30, 16]; each uses the 99th pool percentile as its per-response cutoff. Section 6 compares ablations of our statistic: fixed-temperature typicality at $T = 1$, related by a sign reversal to same-model Fast-DetectGPT [2], the MLE temperature control, and scoring restricted to a top-20 window or the 0.95 nucleus [18].

Statistical evaluation. The pools, thresholds and comparisons were fixed before any collected API response was scored. Confidence intervals for differences in power use a paired bootstrap over test prompts ($B = 2{,}000$) that re-draws the calibration audits and recalibrates the threshold in every replicate, and the paired comparisons within each panel are corrected with Holm's method [17].

5. Substitution Detection under Deployment Variation

Validity on real providers. Table 1 first answers whether an audit can be trusted on real traffic. Calibrated on the pool C, every audit keeps the false-alarm rate on the evaluated plain endpoints below 0.10. For API-TAIL, the largest false-alarm rate over the 17 Qwen and 9 Gemma plain endpoints is 0.035, 0.002, 0.003 and 0.069 on the four panels, and the validity verdict is the same in all 20 re-drawn calibrations. The endpoints are served by independent providers whose decoding settings the auditor does not know; the pool covers them through its range of temperatures, cutoffs, penalties and default prompts.

Detection at the main setting. Validity is easy to obtain by flagging nothing, so the comparison that matters is power at equal validity. API-TAIL detects 10% substitution of the main substitute with power 0.33 (Qwen P30), 0.69 (Qwen P-user), 0.89 (Gemma P30) and 1.00 (Gemma P-user) at $N = 200$. With the common composite calibration, RUT and matched-budget MET have power 0.00 in the P30 main settings. The separate Qwen native-sampling check at $N = 200$ also yields power 0.00 for both tested substitutes at rates up to 0.25 (Section 4). The likelihood-count tests

Audit	Qwen P30		Qwen P-user		Gemma P30		Gemma P-user	
	Power ↑	FAR ↓	Power ↑	FAR ↓	Power ↑	FAR ↓	Power ↑	FAR ↓
API-TAIL (ours)	**0.33**	0.035	**0.69**	0.002	**0.89**	0.003	**1.00**	0.069
NLL count	0.00	0.000	0.00	0.000	0.08	0.000	0.13	0.000
Log-rank count	0.00	0.000	0.00	0.000	0.03	0.000	0.10	0.000
Entropy count	0.01	0.022	0.00	0.065	0.02	0.029	0.00	0.001
RUT [37]	0.00	0.003	n/a		0.00	0.000	n/a	
MET [13]	0.00	0.005	n/a		0.00	0.000	n/a	

Table 1 | Audits with API budget $N = 200$, common source banks, pool C and calibration rule, and method-specific sampling. *FAR* (↓): largest false-alarm rate over the plain endpoints of the panel (17 for Qwen, 9 for Gemma); all values are below the 0.10 validity limit and the validity verdicts hold in 20 of 20 re-drawn calibrations. *Power* (↑): 10% substitution by the panel's main substitute (Qwen3.6-27B served by provider A for Qwen; Gemma-4-26B-A4B for Gemma). RUT and matched-budget MET are evaluated on the P30 panels: RUT uses one API response and 100 local references per prompt; MET uses two responses per side.

Panel	Substitute	Power ↑ ($N = 200$)			Power ↑ ($N = 400$)
		$\rho = 0.05$	$\rho = 0.10$	$\rho = 0.25$	$\rho = 0.10$
Qwen P30	Qwen3.6-27B (provider A)	0.09	0.33	0.73	0.62
	Qwen3.6-27B (developer platform)	0.11	0.42	0.78	0.71
	Qwen3.6-35B-A3B	0.23	0.73	0.98	0.96
	Qwen3.8-Flash	0.29	0.83	1.00	0.98
	DeepSeek-V4.1-Flash	0.76	1.00	1.00	1.00
Qwen P-user	Qwen3.6-27B (provider A)	0.19	0.69	0.98	0.93
	Qwen3.6-27B (developer platform)	0.00	0.00	0.07	0.00
	Qwen3.6-35B-A3B	0.74	1.00	1.00	1.00
	Qwen3.8-Flash	0.69	0.99	1.00	1.00
	DeepSeek-V4.1-Flash	0.94	1.00	1.00	1.00
Gemma P30	Gemma-4-26B-A4B	0.29	0.89	1.00	1.00
	DeepSeek-V4.1-Flash	0.56	1.00	1.00	1.00
	Qwen3.8-27B	0.54	0.99	1.00	1.00
Gemma P-user	Gemma-4-26B-A4B	0.95	1.00	1.00	1.00
	DeepSeek-V4.1-Flash	0.96	1.00	1.00	1.00
	Qwen3.8-27B	0.96	1.00	1.00	1.00

Table 2 | Sensitivity analysis of API-TAIL's detection power (↑) across different substitute types (cross-family models, same-family MoE or smaller siblings, and previous same-size releases), substitution rates ρ, and query budgets N (calibrated on pool C; at $N = 200$, false-alarm rates are those in Table 1). "Provider A" and "developer platform" denote two deployments of the same previous-release substitute.

(NLL, log-rank and entropy counts) reach at most 0.13 across all four panels. Section 6 further compares API-TAIL against ablations of its own alignment and scoring components.

Breadth across substitutes. Table 2 reports every substitute. At $\rho = 0.10$ and $N = 200$, API-TAIL has power of at least 0.69 in 13 of the 16 substitute–panel settings, and the median over the 16 settings is 0.99. Models from another family are detected even at $\rho = 0.05$: DeepSeek-V4.1-Flash

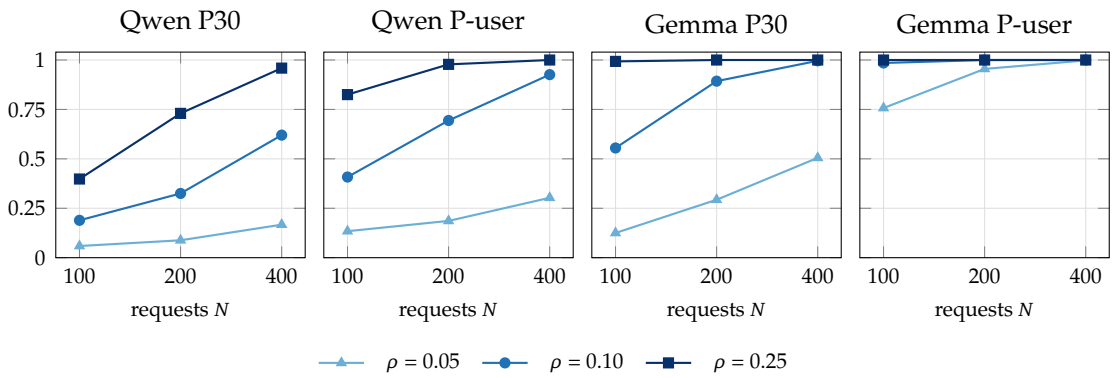


Figure 2 | Power of API-TAIL as the audit budget grows, for the main substitute of each panel (Qwen3.6-27B from provider A on the Qwen panels, Gemma-4-26B-A4B on the Gemma panels) at three substitution rates. Every substitute is listed in Table 2 and Appendix B.

Variant of API-TAIL	Qwen P30	Qwen P-user	Gemma P30	Gemma P-user
MLE temperature	+0.22 [+0.07, +0.27]	+0.63 [+0.37, +0.67]	+0.09 [+0.06, +0.13]	0.00 [0.00, 0.00]
Fixed T=1, no alignment	+0.33 [+0.22, +0.42]	+0.69 [+0.47, +0.78]	−0.04 [-0.07, +0.05]	+1.00 [+1.00, +1.00]
Top-20 window	−0.01 [-0.04, +0.02]	+0.62 [+0.41, +0.68]	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]
Nucleus (p=0.95)	+0.30 [+0.20, +0.40]	+0.69 [+0.47, +0.78]	+0.83 [+0.78, +0.88]	0.00 [0.00, +0.01]
Threshold τ (API-TAIL / MLE)	7 / 9	5 / 10	8 / 9	5 / 5
Max. FAR without penalties	0.269†	0.074	0.003	0.069

Table 3 | Design ablations (N = 200, main substitute, ρ = 0.10). Top: power of API-TAIL minus the power of the variant, with paired-bootstrap 95% intervals (B = 2,000); positive values favour API-TAIL. Middle: integer composite-null thresholds τ of API-TAIL and of the MLE variant. Bottom: largest plain-endpoint false-alarm rate of API-TAIL with the five penalty configurations removed from C (†: above the 0.10 limit).

with power 0.56 to 0.96 and Qwen3.8-27B (as a substitute for Gemma) with 0.54 and 0.96. The mixture-of-experts and smaller siblings, which share tokenizer and training lineage with the claimed model, are detected with power 0.73 to 1.00 at ρ = 0.10. The closest substitute is the previous release of the claimed model, Qwen3.6-27B: at ρ = 0.10 it is detected with power 0.33 (P30) and 0.69 (P-user) when served by provider A and with 0.42 on short responses from the developer's platform, rising to 0.62, 0.93 and 0.71 at N = 400. Its developer-platform deployment remains difficult to detect on long default-sampling responses, with power at most 0.07 across the evaluated budgets and substitution rates.

Audit budget. Power grows smoothly with both the budget and the substitution rate (Figure 2). For the main substitutes, doubling the budget from N = 200 to N = 400 raises power from 0.33 to 0.62 on Qwen P30 and from 0.69 to 0.93 on Qwen P-user, and at ρ = 0.25 already 200 requests give power 0.73 to 1.00 on every panel. Over all substitutes at N = 400 and ρ = 0.10, 14 of the 16 settings reach power of at least 0.70 and 12 reach 0.96 or higher (Appendix B gives every budget and rate).

6. Why the Design Works

Table 3 compares variants of API-TAIL on the same responses and pool, with the same calibration rule applied separately to each variant.

Alignment versus the MLE control. The defining choice of API-TAIL is its alignment objective. Replacing it by the MLE control, with everything else fixed, lowers power by 0.22 on Qwen P30, 0.63 on Qwen P-user and 0.09 on Gemma P30; all three differences are significant after Holm correction, and on Gemma P-user both audits reach power 1.00. The two estimators do not disagree about the temperature. From honest traffic to 10% substitution, the mean fitted temperature moves from 0.498 to 0.567 under alignment and from 0.498 to 0.561 under the MLE control on Qwen P30, and from 0.849 to 0.869 versus 0.847 to 0.867 on Qwen P-user. They disagree about the null. Because alignment centers the very standardized scores that are counted, configurations outside the temperature family still produce few exceedances, and the composite threshold stays low: 7 versus 9, 5 versus 10 and 8 versus 9 atypical responses out of 200 on the three unsaturated panels. The configuration that sets the likelihood audit's threshold is the repetition penalty 1.10 on Qwen P30, the combined presence and repetition penalty on Qwen P-user, and the default system prompt on Gemma P30, where the MLE control leaves larger standardized shifts.

Aligning, rather than fixing the temperature. Without any alignment, the threshold has to cover every legitimate temperature at once, and power falls to zero on three panels. Alignment turns these panels into detections with power 0.33, 0.69 and 1.00. On Gemma P30 the two audits are statistically indistinguishable.

Scoring the full vocabulary. Substituted text contains tokens to which the claimed model assigns low probability, and these lie in the tail of the vocabulary, beyond the top-20 log-probabilities that many serving APIs expose. Computing the scores from a top-20 window leaves the short-response panels unchanged, but on Qwen P-user, whose responses are up to 400 tokens long, power drops from 0.69 to 0.08. Our nucleus-restricted implementation lowers power by 0.30 to 0.83 on the three unsaturated panels. This difference reflects the joint effect of truncated renormalization and the positions retained for scoring (Appendix B).

Covering legitimate penalties. Penalties are part of how providers serve models, so they belong in the pool. Without the five penalty configurations, API-TAIL remains valid on three of the four panels, but on Qwen P30 the false-alarm rate of the model developer's own endpoint rises to 0.269; including them restores validity, with a maximum false-alarm rate of 0.035 across Qwen P30 plain endpoints. The endpoint that documents injecting its own system prompt is flagged in 79% (Qwen P30) and 99% (Qwen P-user) of audits, whereas FP4-quantized endpoints pass on short responses (Appendix A).

7. Related Work

Auditing which model an API serves. Model equality testing [13] compares API and reference samples using maximum mean discrepancy and also considers a composite null covering FP32 and FP16 reference deployments. The rank-based uniformity test [37] uses one API response per natural user prompt and tests its randomized rank among locally generated reference responses.

Cai et al. [5] study model substitution in LLM APIs, including quantized and randomized substitution, and compare several detection approaches. Empirical studies document deceptive model claims and routing opacity across commercial API gateways and providers [35, 21]. Separately, controlled experiments show that inference backends can change model outputs with fixed weights and decoding parameters [26]. Related work detects behavioral changes over time [7, 29, 8, 6], fingerprints models or inference stacks using probes or output idiosyncrasies [15, 11, 27, 31, 19, 10, 4, 28], audits substitution and routing dilution under a probe budget [33], and shows that fingerprints can be spoofed [34]. With provider cooperation, inference can be verified despite numerical nondeterminism [32, 25, 20]. API-TAIL also operates on natural prompts, using one audited API response per prompt. It combines temperature alignment with a likelihood-tail count calibrated across legitimate decoding and serving configurations [12, 22, 26], formulated as a composite null [3, 1]. Unlike sample-comparison methods such as MET and RUT, it scores audited responses directly under the claimed model without generating reference responses for each audit prompt.

Likelihood and typicality. Likelihood and token-rank statistics support zero-shot text detection [30]; GLTR [14] visualizes likelihood, rank and entropy statistics to assist human detection. DetectGPT [24] uses probability curvature; Fast-DetectGPT [2] forms a standardized log-likelihood criterion using scoring and sampling models; and Binoculars [16] uses the ratio of log-perplexity to log-cross-perplexity. When both scoring and sampling distributions equal our $q_{t,1}$, the Fast-DetectGPT criterion d satisfies $z(1) = -d$, apart from numerical variance floors. Our fixed-temperature control counts exceedances in the opposite score tail and calibrates the batch count across the composite null. API-TAIL instead fits the temperature to the audited batch before counting exceedances [9]. Comparing surprisal with entropy also underlies locally typical sampling [23]; nucleus sampling [18] truncates the vocabulary tail. Our nucleus ablation combines truncated renormalization with a change in the positions scored (Appendix B).

8. Conclusion

Honest API providers serve the same language model in many legitimate ways, which turns substitution auditing into a test against a composite null. API-TAIL absorbs this variation instead of suffering from it: aligning one effective temperature absorbs batch-level shifts from honest decoding and serving choices, the responses that remain in the atypical tail carry the evidence of substitution, and calibrating their count across a pool of legitimate configurations covers residual deployment differences. On real providers of two model families, the audit meets the false-alarm criterion on all 26 plain honest endpoints and detects 10% substitution with a few hundred requests, at recorded API charges of cents and a few GPU-minutes of local scoring.

Contributors

Haocheng Mei, Yuchuan Tian, Ye Yuan, Chao Xu, Allen Chen, Kai Han, Yunhe Wang

References

[1] Aranyak Acharyya, Carey E. Priebe, and Hayden S. Helm. 2025. Testing for LLM response differences: The case of a composite null consisting of semantically irrelevant query perturbations. Preprint, arXiv:2509.10963.

[2] Guangsheng Bao, Yanbin Zhao, Zhiyang Teng, Linyi Yang, and Yue Zhang. 2024. Fast-DetectGPT: Efficient zero-shot detection of machine-generated text via conditional probability curvature. In *The Twelfth International Conference on Learning Representations*.

[3] Roger L. Berger. 1982. Multiparameter hypothesis testing and acceptance sampling. *Technometrics*, 24(4):295–300.

[4] Tomáš Bruckner. 2026. One token is enough: Fingerprinting and verifying large language models from single-token output distributions. Preprint, arXiv:2607.10252.

[5] Will Cai, Tianneng Shi, Xuandong Zhao, and Dawn Song. 2025. Are you getting what you pay for? Auditing model substitution in LLM APIs. Preprint, arXiv:2504.04715.

[6] Timothée Chauvin, Clément Lalanne, Erwan Le Merrer, Jean-Michel Loubes, François Taïani, and Gilles Tredan. 2026. Token-efficient change detection in LLM APIs. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 13247–13277.

[7] Lingjiao Chen, Matei Zaharia, and James Zou. 2023. How is ChatGPT's behavior changing over time? Preprint, arXiv:2307.09009.

[8] Alden Dima, James Foulds, Shimei Pan, and Philip Feldman. 2025. You've changed: Detecting modification of black-box large language models. Preprint, arXiv:2504.12335.

[9] David Donoho and Jiashun Jin. 2004. Higher criticism for detecting sparse heterogeneous mixtures. *The Annals of Statistics*, 32(3):962–994.

[10] Yijia Fang, Yiqing Feng, Bingyu Li, and Mingxun Zhou. 2026. KBF: Knowledge boundary as fingerprint for language model and black-box API auditing. Preprint, arXiv:2605.29524.

[11] Matthew Finlayson, Xiang Ren, and Swabha Swayamdipta. 2024. Logits of API-protected LLMs leak proprietary information. In *The First Conference on Language Modeling*.

[12] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*.

[13] 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*.

[14] Sebastian Gehrmann, Hendrik Strobelt, and Alexander M. Rush. 2019. GLTR: Statistical detection and visualization of generated text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 111–116.

[15] 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.

[16] Abhimanyu Hans, Avi Schwarzschild, Valeriia Cherepanova, Hamid Kazemi, Aniruddha Saha, Micah Goldblum, Jonas Geiping, and Tom Goldstein. 2024. Spotting LLMs with binoculars: Zero-shot detection of machine-generated text. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 17519–17537.

[17] Sture Holm. 1979. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70.

[18] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *The Eighth International Conference on Learning Representations*.

[19] Pingyi Hu, Xiaofan Bai, Xiaojing Ma, Chaoxiang He, Dongmei Zhang, and Bin Benjamin Zhu. 2025. RESF: Regularized-entropy-sensitive fingerprinting for black-box tamper detection of large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 4889–4903.

[20] Adam Karvonen, Daniel Reuter, Roy Rinberg, Luke Marks, Adrià Garriga-Alonso, and Keri Warr. 2025. DiFR: Inference verification despite nondeterminism. Preprint, arXiv:2511.20621.

[21] Guanjie Lin, Yinxin Wan, Shichao Pei, Ting Xu, Kuai Xu, and Guoliang Xue. 2026. Behavioral consistency and transparency analysis on large language model API gateways. In *Proceedings of the 2026 ACM Internet Measurement Conference*, pages 563–573. doi:10.1145/3777912.3809156.

[22] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In *Proceedings of Machine Learning and Systems*, volume 6, pages 87–100.

[23] Clara Meister, Tiago Pimentel, Gian Wiher, and Ryan Cotterell. 2023. Locally typical sampling. *Transactions of the Association for Computational Linguistics*, 11:102–121.

[24] Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. 2023. DetectGPT: Zero-shot machine-generated text detection using probability curvature. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 24950–24962.

[25] Jack Min Ong, Matthew Di Ferrante, Aaron Pazdera, Ryan Garner, Sami Jaghouar, Manveer Basra, Max Ryabinin, and Johannes Hagemann. 2025. TOPLOC: A locality sensitive hashing scheme for trustless verifiable inference. Preprint, arXiv:2501.16007.

[26] David Pape, Jonathan Evertz, and Lea Schönherr. 2026. The silent hyperparameter: Quantifying the impact of inference backends on LLM reproducibility. Preprint, arXiv:2605.19537.

[27] Dario Pasquini, Evgenios M. Kornaropoulos, and Giuseppe Ateniese. 2025. LLMmap: Fingerprinting for large language models. In *Proceedings of the 34th USENIX Security Symposium*, pages 299–318.

[28] Gurvan Richardeau, Gohar Dashyan, Erwan Le Merrer, and Gilles Tredan. 2026. FLIPS: Instance-fingerprinting for LLMs via pseudo-random sequences. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 38974–38993.

[29] Leo Richter, Xuanli He, Pasquale Minervini, and Matt J. Kusner. 2025. An auditing test to detect behavioral shift in language models. In *The Thirteenth International Conference on Learning Representations*.

[30] Jinyan Su, Terry Yue Zhuo, Di Wang, and Preslav Nakov. 2023. DetectLLM: Leveraging log rank information for zero-shot detection of machine-generated text. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 12395–12412.

[31] Mingjie Sun, Yida Yin, Zhiqiu Xu, J. Zico Kolter, and Zhuang Liu. 2025. Idiosyncrasies in large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 57854–57885.

[32] Yifan Sun, Yuhang Li, Yue Zhang, Yuchen Jin, and Huan Zhang. 2024. SVIP: Towards verifiable inference of open-source large language models. Preprint, arXiv:2410.22307.

[33] Yuewei Zhang, Zhi-Hai Zhang, and Hanzhang Qin. 2026a. Which model is actually serving you? IRIS: Budgeted black-box auditing of model substitution and routing dilution in LLM gateways. Preprint, arXiv:2607.20860.

[34] Jiahao Zhang, Xiuyu Li, and Suhang Wang. 2026b. Your "Pro" LLM subscription may actually be "free": Exposing fingerprint spoofing risks in LLM inference services. Preprint, arXiv:2606.16100.

[35] Yage Zhang, Yukun Jiang, Zeyuan Chen, Michael Backes, Xinyue Shen, and Yang Zhang. 2026c. Real money, fake models: Deceptive model claims in shadow APIs. In *Proceedings of the 2026 ACM SIGSAC Conference on Computer and Communications Security* (accepted; to appear).

[36] 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*.

[37] Xiaoyuan Zhu, Yaowen Ye, Tianyi Qiu, Hanlin Zhu, Sijun Tan, Ajraf Mannan, Jonathan Michala, Raluca Ada Popa, and Willie Neiswanger. 2025. Auditing black-box LLM APIs with a rank-based uniformity test. Preprint, arXiv:2506.06975.

A. Composite-Null Pool and Endpoints

Pool C (Table 4) uses local M responses on disjoint prompts (1,000×4 Qwen; 500×2 Gemma), varying temperature (0.7, 1.0), top-p (0.8–1.0), top-k (none, 20, 64), penalties and precision. FP4 or system-prompt endpoints and those failing the 20-prompt probe (T=0 vs. T=1, first 16 tokens, Fisher p<0.05) are *special*; other probed endpoints are *plain*. G14 is unclassified (*: developer).

Configuration group	Q-P30	Q-user	G-P30	G-user
Protocol defaults (two seeds)	2	2	2	2
Other temperature, top-p, top-k	5	8	5	8
Generation-config defaults	1	–	1	–
Default system prompt	1	1	1	1
FP8 / INT4 weights	4	3	2	2
Penalties (pres., rep., combined)	5	5	5	5
Total	18	19	16	18

Table 4 | Members of the composite-null pool C per panel.

Qwen3.8-27B (Q01–Q13)				
ID	Prec.	Class	P30	P-user
Q01	FP8	plain	0.002	0.000
Q02*	n/s	plain	0.035	0.000
Q03	FP16	plain	0.001	0.000
Q04	FP8	plain	0.001	0.002
Q05	n/s	plain	0.001	0.000
Q06	FP8	plain	0.001	0.000
Q07	BF16	plain	0.002	0.000
Q08	n/s	plain	0.002	0.000
Q09	FP8	plain	0.004	0.000
Q10	FP8	plain	0.002	0.000
Q11	n/s	plain	0.001	0.000
Q12	FP8	plain	0.000	0.000
Q13	n/s	plain	0.017	0.000

Qwen (Q14–Q21) & Gemma (G01–G05)				
ID	Prec.	Class	P30	P-user
Q14	FP8	plain	0.001	0.000
Q15	n/s	plain	0.000	0.000
Q16	FP8	plain	0.001	0.000
Q17	FP8	plain	0.002	0.000
Q18	FP4	FP4	0.000	0.000
Q19	FP4	FP4, no-T	0.003	0.000
Q20	FP4	FP4	0.000	0.000
Q21	FP8	sys. prompt	0.788	0.993
G01	BF16	plain	0.000	0.000
G02	FP8	plain	0.002	0.000
G03	FP8	plain	0.003	0.001
G04	FP8	plain	0.000	0.000
G05	n/s	plain	0.000	0.000

Gemma-4-31B-it (G06–G16)				
ID	Prec.	Class	P30	P-user
G06	BF16	plain	0.001	0.003
G07	FP8	plain	0.000	0.001
G08	n/s	plain	0.000	0.007
G09	FP8	plain	0.000	0.069
G10	FP4	FP4, no-T	0.009	0.360
G11	FP4	FP4, no-T	0.012	0.344
G12	FP4	FP4	0.009	0.135
G13	n/s	no-T	0.004	0.134
G14	n/s	no probe	–	0.003
G15	FP4	FP4, no-T	0.013	0.023
G16	FP4	FP4, sys., no-T	0.004	0.525

Table 5 | Per-endpoint false-alarm rate of API-TAIL (N=200, pool C, 4,000 audits; "no-T": T ignored).

B. Baselines, Full Power Grid and Reproducibility

MLE minimizes $|\sum_r S_r(T)|$ on $\mathcal{T}$; fixed-T counts $z_r(1) > 3$. RUT [37] scores mean log token midranks: FP16 ties within the stored top-k window, reported ranks outside. Response ranks use $(a + U(b+1))/(m+1)$ instead of $(a+Ub)/m$: a, b count smaller/equal reference scores, $m = 100$, $U \sim \text{Unif}(0,1)$. The Cramér–von Mises $-\log p$ is composite-calibrated.

MET [13] uses unbiased Hamming MMD: sequences truncated/padded to 30 tokens, two API and two local responses per prompt, with composite calibration. Native MET: Section 4. Nucleus scoring renormalizes the shortest probability-ordered prefix reaching mass 0.95, capped at 4,096 tokens; insufficient mass retains this cap and records overflow. Tokens outside contribute to neither S nor V, changing both normalization and scored positions.

Scoring is bitwise reproducible: Transformers, batch 1, deterministic kernels, FP64 accumulation over 151 temperatures/pass. Per audit: recorded API charges 0.007–0.155 USD; local scoring 2–8 L20X GPU-minutes.

		ρ = 0.05			ρ = 0.10			ρ = 0.25		
Panel	Substitute	100	200	400	100	200	400	100	200	400
Qwen P30	Qwen3.6-27B (provider A)	0.06	0.09	0.17	0.19	0.33	0.62	0.40	0.73	0.96
	Qwen3.6-27B (developer platform)	0.07	0.11	0.22	0.26	0.42	0.71	0.54	0.78	0.98
	Qwen3.6-35B-A3B	0.14	0.23	0.49	0.47	0.73	0.96	0.85	0.98	1.00
	Qwen3.8-Flash	0.20	0.29	0.57	0.53	0.83	0.98	0.93	1.00	1.00
	DeepSeek-V4.1-Flash	0.52	0.76	0.96	0.97	1.00	1.00	1.00	1.00	1.00
Qwen P-user	Qwen3.6-27B (provider A)	0.13	0.19	0.30	0.41	0.69	0.93	0.82	0.98	1.00
	Qwen3.6-27B (developer platform)	0.00	0.00	0.00	0.01	0.00	0.00	0.04	0.07	0.07
	Qwen3.6-35B-A3B	0.49	0.74	0.92	0.94	1.00	1.00	1.00	1.00	1.00
	Qwen3.8-Flash	0.40	0.69	0.88	0.88	0.99	1.00	1.00	1.00	1.00
	DeepSeek-V4.1-Flash	0.73	0.94	1.00	0.99	1.00	1.00	1.00	1.00	1.00
Gemma P30	Gemma-4-26B-A4B	0.12	0.29	0.51	0.56	0.89	1.00	0.99	1.00	1.00
	DeepSeek-V4.1-Flash	0.33	0.56	0.86	0.91	1.00	1.00	1.00	1.00	1.00
	Qwen3.8-27B	0.28	0.54	0.78	0.87	0.99	1.00	1.00	1.00	1.00
Gemma P-user	Gemma-4-26B-A4B	0.76	0.95	1.00	0.98	1.00	1.00	1.00	1.00	1.00
	DeepSeek-V4.1-Flash	0.78	0.96	1.00	0.99	1.00	1.00	1.00	1.00	1.00
	Qwen3.8-27B	0.77	0.96	1.00	0.99	1.00	1.00	1.00	1.00	1.00

Table 6 | Power of API-TAIL across substitutes, rates ρ and budgets $N \in \{100, 200, 400\}$ (1,000 audits).