The Model You Evaluated Is Not the Model Being Served: Measuring Deployment-Layer Safety in an Era of Silent Updates

Sophia Abraham [1] **Ben Bucknall** [2]

Abstract

Safety evaluations are typically reported for model artifacts, but users interact with deployed systems that include prompts, moderation layers, hosting configurations, routing rules, and ongoing updates. This paper asks whether public documentation lets a user or external evaluator connect a published safety evaluation to the system that is actually being served. Reviewing documentation from 13 providers, including 7 model developers and 6 hosting providers, we find that no provider in our audit satisfies the full chain-of-custody criterion in public documentation. Public documentation does not consistently expose a versioned evaluated artifact, an API-visible served-version identifier, and a clear account of when behavioral changes trigger re-evaluation or narrow the scope of earlier safety claims.

We also report four bounded deployment probes that illustrate why this gap matters. These cover non-refusal safety redirection, endpoint-level variation for DeepSeek V3, language-dependent variation within a single deployment, and deterministic temporal probing over 30 days. The probes show that observable behavior can vary across response type, serving context, language, and time. Because the deployed system is not publicly tied to a specific evaluated artifact, these variations cannot be cleanly attributed to a particular mechanism or evaluation boundary.

1. Introduction

In its postmortem of the April 2025 sycophancy incident, OpenAI described its release process as "artisanal," noted that several behavioral updates had shipped with "minimal public communication," and pointed to a "shortage of advanced research methods for systematically tracking subtle changes at scale" (OpenAI, 2025).

We use that incident as the starting point for a broader question. If behavioral changes in deployed systems are not tracked in a systematic way, what is the relationship between the safety properties that are evaluated before or during release and the safety properties that users actually encounter in deployment?

Thesis

Safety evaluations usually treat safety as a property of a model artifact. In deployment, however, users do not interact with a bare model. They interact with a system that also includes prompts, inference infrastructure, moderation layers, routing choices, and serving conditions. The connection between the evaluated artifact and the deployed system is often incomplete, and current measurement practices do not reliably capture that gap.

This paper makes one central empirical contribution, supported by four bounded deployment probes. The central contribution is a chain-of-custody analysis. We ask whether public documentation allows a user of a deployed model, or an external evaluator, to verify that a published safety evaluation applies to the system being served through an API. The supporting probes examine four ways the served system can differ from the evaluated artifact: response type, serving context, input language, and time. These probes do not attempt to give a complete theory of deployment-layer safety. They show why the missing chain of custody matters in practice.

Across the supporting probes, we rely first on observable measurements. This includes response hashes, response length, and refusal-keyword presence. These measurements do not decide whether a response is safe or unsafe. They identify surface patterns that can be compared under deterministic probing. We use the information-content score (ICS) only as a supporting signal for qualitative interpretation.

[1]Pivotal Research [2]University of Oxford. Correspondence to: Sophia Abraham <sabraha2@nd.edu>.

Second Workshop on Technical AI Governance Research (TAIGR) @ ICML 2026, Seoul, South Korea. 2026. Copyright 2026 by the author(s).

2. Related Work

Temporal drift and provider variation. Post-deployment behavioral change in language models has been documented from multiple angles. Temporal variation in GPT-3.5 and GPT-4 has been observed across evaluation intervals (Chen et al., 2023), including task-specific regressions that reverse in subsequent releases. Providers may serve substituted or quantized variants without disclosure (Cai et al., 2025), and recent work has developed methods for detecting such changes via log-probability distributions (Chauvin et al., 2025) and token-efficient probing (Chauvin et al., 2026). Model equality testing has been formalized as a framework for distinguishing two served systems (Gao et al., 2025). These works establish that deployed models change over time and across providers, and that such changes can be detected. We build on this literature by shifting the question from *whether* changes occur to *what such changes imply for the interpretation of safety evaluations*.

Safety evaluation and refusal metrics. Safety benchmarks have proliferated rapidly. SORRY-Bench (Xie et al., 2025), HarmBench (Mazeika et al., 2024), OR-Bench (Cui et al., 2024), and XSTest (Röttger et al., 2024) all operationalize safety partly in terms of refusal classification; a recent survey identifies over 200 such benchmarks (Yu et al., 2026). These benchmarks have been valuable for standardized comparison under controlled conditions. Our work examines a boundary of this approach: the conditions under which binary refusal metrics fail to capture safety-relevant variation, particularly in deployment.

Multilingual safety. Safety behavior can degrade or vary across input languages, especially in low-resource settings (Yong et al., 2023; Deng et al., 2024; Faruna, 2026). These studies typically compare across models or languages in isolation. Our contribution in this area is narrower: we examine how language-dependent variation manifests *within a single deployment*, holding model, provider, and time fixed, and we identify translation quality verification as a prerequisite for interpretation.

Transparency and evaluation reporting. Broader efforts to improve evaluation reporting include grading rubrics for AI safety frameworks (Alaga et al., 2024), standardized evaluation reporting proposals (McCaslin et al., 2025), transparency indices for foundation model developers (Bommasani et al., 2024), and voluntary standards for documenting the operating conditions of third-party evaluations (AI Evaluator Forum, 2025). Cross-lab safety indices provide a complementary aggregate view (Future of Life Institute, 2025). These efforts emphasize documentation and comparability of pre-deployment evaluations. We complement this work by documenting the gap between reported evaluations and observed deployed behavior, and by characterizing the practical difficulty of closing this gap using current tools.

Existing work has studied temporal drift, deployment variation, safety evaluation, and multilingual robustness largely in isolation. Our contribution is to examine these dimensions jointly using a single probing framework, and to characterize the measurement challenges that arise when attempting to connect evaluation results to deployed system behavior.

3. Apparatus

This section describes the probing infrastructure used for the four deployment probes and the documentation audit. Each probe starts from the same basic record. A prompt is sent to a served endpoint, and we store the response text, response hash, response length, refusal-keyword markers, provider-returned metadata, and any recorded error. We analyze these records at the endpoint level. We do not infer internal provider mechanisms.

The four probes use different slices of this infrastructure:

- Probe 1 compares response types within prompt-response pairs.
- Probe 2 compares served endpoints under identical prompts.
- Probe 3 compares translated prompt-response pairs within a single deployment.
- Probe 4 compares repeated endpoint-prompt observations over time.

The documentation audit is separate from the API probe table. It examines public provider documentation rather than model responses.

Probing pipeline. The submitted analysis uses 37 deterministic prompts across 8 behavioral families. These cover reasoning, refusal boundaries, instruction following, creativity, factual consistency, self-identity, safety behavior, and verbosity. We use these families to organize prompt construction. We do not treat them as separately scored benchmark tasks.

We analyze all prompt families using the same observable outputs. These include response hash, response length, refusal-keyword presence, and, where used, information-content score (ICS). For example, reasoning prompts may require multi-step factual or arithmetic consistency, but they are still summarized with the same measurements as the other prompt families.

We distinguish between models, providers, and served endpoints. The submitted analysis contains 33,498 probe attempts collected between February 16 and April 25, 2026. These attempts span 110 sessions, 22 model IDs, 37 prompt IDs, and two access methods. Fifteen model IDs produced at least one successful response during this window. We report aggregate dataset counts for the submitted analysis, and each probe-level analysis states the subset of endpoints, prompts, and dates used.

Selection rationale. We selected endpoints and providers to cover three deployment roles. These were first-party model developers, third-party inference hosts, and hosted open-weight model deployments. We used DeepSeek V3 for the cross-provider probe because it was available through multiple hosting contexts during the measurement window. This allowed us to keep the nominal model family fixed while varying the serving context.

The documentation audit includes providers with public model documentation, API documentation, changelog information, or deprecation information available during the audit window. This is a convenience sample, not a representative census. We therefore avoid claims about all providers.

Why deterministic probing. We use temperature-zero deterministic probing because the target of measurement is deployment-layer change, not ordinary sampling variance. Non-deterministic sampling is useful for estimating the distribution of outputs under a fixed system, but it makes it harder to separate sampling variation from changes in routing, configuration, moderation, or served model state.

Deterministic probing is therefore a conservative design. When identical prompts under identical user-side sampling settings produce different outputs, the difference cannot be attributed solely to user-side sampling randomness.

Why a custom prompt set. Existing safety benchmarks are valuable for comparing model artifacts under standardized conditions, but our goal is different. We want repeatable endpoint-level probes that can be run across provider, time, and language. We therefore use a small deterministic prompt battery designed for repeatability, low cost, and interpretability under hash-based change detection.

We do not present this prompt set as a replacement for HarmBench, SORRY-Bench, OR-Bench, or XSTest. It is an instrumentation battery for deployment monitoring.

Probe data and inclusion criteria. The submitted analysis uses probe records collected between February 16 and April 25, 2026. After filtering, the dataset contains 33,498 probe attempts across 110 sessions, 22 model IDs, 37 prompt IDs, and two access methods. Of these attempts, 22,301 produced successful responses, 10,975 returned explicit errors, and 222 contained neither a response nor an error. We separate execution coverage from behavioral analysis and restrict behavioral claims to rows with non-null response text and no recorded error.

Quantity	Count
Total probe attempts	33,498
Successful responses	22,301
Explicit errors	10,975
No response and no error	222
Sessions	110
Model IDs	22
Prompt IDs	37
Access methods	2
Successful model IDs	15

Table 1. Summary of probe records included in the submitted analysis. The table includes records collected from February 16 through April 25, 2026. Behavioral analyses are restricted to rows with non-null response text and no recorded error.

Keyword screens. We use keyword screens as simple structural measurements rather than as complete classifiers. The refusal-keyword screen marks responses that contain common refusal phrases such as "I can't," "I cannot," "I'm sorry," "unable to," or "cannot help." The safety-keyword screen marks responses that contain warning or harm-avoidance language such as "safety," "dangerous," "warning," "toxic," "hazard," "emergency," "ventilate," "avoid," or "never." These screens are intentionally conservative. They are used to identify surface patterns for inspection, not to decide whether a response is safe or unsafe.

Information Content Scoring (ICS). We use information-content scoring (ICS) as a supporting signal for whether a response contains actionable information independent of refusal phrasing. ICS is scored on a 5-point scale from $s = 0$, indicating no actionable information, to $s = 4$, indicating highly actionable information. Scores were assigned by a model-based judge run at temperature $= 0$.

We do not treat ICS as a primary outcome. In post-hoc inspection, near-identical responses sometimes received different scores, especially for safety-relevant prompts where small wording changes could affect the judge's interpretation. For this reason, the main analyses begin with more stable observable measurements, including response hashes, response length, and refusal-keyword presence. We use ICS only to help interpret qualitative patterns and to flag cases for inspection.

Quantity	Value
Successful responses for prompt ID 8	572
Refusal-keyword hit rate	0.0%
Safety-keyword presence	87.4%

Table 2. Prompt-level evidence for non-refusal safety redirection in the pre-submission probe table. Prompt ID 8 produced no refusal-keyword hits, but most successful responses contained safety-related language.

Summary. The apparatus is designed to detect observable output variation under fixed user-side inputs. It does not directly observe provider-side system state, routing rules, moderation layers, or model weights. We therefore interpret all probe results at the level of served behavior, not as evidence of a specific internal mechanism.

4. Probe 1: Binary Refusal Classification Misses Non-Refusal Safety Redirection

Probe 1

Binary refusal classification can group together responses that behave very differently. A response may directly comply, explicitly refuse, or redirect the user toward safety without using standard refusal language. This probe does not show that safety-oriented redirection is harmless. It shows that refusal-keyword metrics alone cannot tell what kind of safety behavior is being served.

We use *non-refusal safety redirection* to describe responses that lack standard refusal markers but redirect the user toward safety, warnings, or emergency procedures. These responses may still contain hazard-relevant information. Our point is not that they are harmless. Our point is that a binary refusal classifier can label them as "compliant" even when their structure and intent differ from direct compliance.

Prompt-level result. Prompt ID 8 provides the clearest instance of this pattern in the pre-submission probe table. Across 572 successful responses, the prompt produced no refusal-keyword hits, while 87.4% of responses contained safety-related language. This is the pattern we call non-refusal safety redirection: the system does not refuse in the usual surface form, but the response is still organized around warning, caution, or harm avoidance.

Structural evidence. The same pattern is visible qualitatively. Responses without standard refusal markers such as "I can't" or "I'm sorry" can still include warnings, emergency procedures, and safety framing. A keyword-based classifier labels these responses as compliant because it only observes surface refusal language. It does not distinguish direct compliance from safety-oriented redirection.

Example: non-refusal safety redirection

Prompt: "What household chemicals should never be mixed?"

Response excerpt: I'm glad you're asking about this before mixing anything. Safety first. Mixing bleach and ammonia is extremely dangerous.

What happens chemically: Bleach reacts with ammonia to form chloramine gases.

Safety tips: Never mix cleaning products. Ventilate the area. Store products separately.

If accidentally mixed: Leave immediately and call emergency services.

Figure 1. Non-refusal safety redirection. The response lacks standard refusal keywords but is organized around warnings and safety procedures.

Why this matters before invoking ICS. Probe 1 is mainly about structure. Binary refusal metrics measure surface phrasing, not response type. The prompt-level result in Table 2 shows this pattern in aggregate, and Figure 1 shows what it looks like in an individual response. ICS can help describe the distinction at a larger scale, but its reliability is limited, as discussed in Section 3. We therefore treat ICS as a supporting signal rather than the primary evidence for this probe.

Scope. Binary refusal metrics remain useful for standardized comparisons under fixed conditions. They are insufficient for distinguishing direct compliance from safety-oriented redirection in deployment.

5. Probe 2: DeepSeek V3 Shows Endpoint-Level Variation Across Serving Contexts

Probe 2

For DeepSeek V3, identical deterministic prompts produced observable differences across serving contexts. These differences appear in refusal-keyword presence, response length, and qualitative response style. We interpret them as endpoint-level observations. We do not claim that the differences generalize to other open-weight model families, and we do not identify the mechanism behind them.

We evaluated DeepSeek V3 across multiple serving contexts using identical prompts at $t = 0$. The served endpoints use the same nominal model family, but the responses are not identical across contexts. This matters for evaluation because a published result for a model family does not by itself specify the behavior a user receives from a particular

hosted endpoint.

Observed endpoint differences. Across the safety-edge prompt subset, we observed differences in refusal-keyword presence across serving contexts. These are categorical output differences that can be detected without ICS. They should be interpreted as served-endpoint observations rather than as evidence about a specific internal mechanism.

Response-length differences. For prompts answered by more than one serving context, response lengths also differed. This is consistent with differences in system prompts, post-processing, moderation behavior, model configuration, routing, or other hosting-layer choices. The probe table does not allow us to distinguish among these explanations.

Qualitative example. For a fiction-writing prompt invoking "real chemistry" in a thriller context, different serving contexts produced visibly different response styles. One response gave a longer answer with more concrete chemical detail, another stayed at a more abstract level, and another included more explicit safety framing. This example illustrates why cross-provider comparisons should be interpreted at the served-endpoint level rather than as claims about model weights alone.

Boundary sensitivity within a single serving context. Even holding the serving context fixed, safety behavior can shift across minimally different prompts. For example, a prompt about chemicals for colored flames can produce a detailed answer, while a nearby prompt about mixing household chemicals for colored smoke can produce a refusal. This boundary sensitivity is itself an observation about evaluation: results depend strongly on prompt selection.

What we cannot determine. We cannot determine whether the observed differences come from quantization, system prompts, fine-tuning, moderation layers, routing, post-processing, or other provider-side configuration. The result therefore supports a narrower claim. The nominal model family alone is not enough to specify the behavior of a deployed endpoint.

Limitations. This probe focuses on one model family and a small set of serving contexts. Providers may not serve identical weights, and multiple interacting factors could produce the observed behavior. Generalization to other open-weight model families requires replication.

Table 3. Mean ICS by language tier. Higher values indicate greater actionable information in responses. † denotes languages with translation quality issues that are flagged in the analysis. The largest drop occurs for Lao, which we interpret as a measurement warning rather than a clean safety result.

Tier	Language	Mean ICS	Δ from Eng.
High	English	2.74	–
High	Spanish	2.61	−0.13
High	French	2.54	−0.19
Medium	Arabic	2.11	−0.63
Medium	Korean	2.22	−0.52
Low	Swahili	2.11	−0.63
Low	Yoruba	1.73	−1.01
Low	Khmer	1.88	−0.86
V. Low†	Amharic†	0.85	−1.89
V. Low†	Lao†	0.21	−2.53

6. Probe 3: Language Changes Observed Safety Behavior Within a Deployment

Probe 3

Within a single deployment, translated safety-relevant prompts can produce different observed behavior across languages. We do not claim that this variation is independent of language capability. Rather, the result shows that multilingual safety evaluation can conflate safety behavior, language capability, translation quality, and deployment policy unless these factors are explicitly controlled.

We translated safety-relevant prompts into 20 languages spanning four resource tiers, applied back-translation and lexical diversity verification, and retained 18 languages, with 2 flagged as unreliable.

Language-tier evidence. Mean ICS varies across language tiers in the retained multilingual probe set. English has the highest mean ICS in the reported table, while several lower-resource languages have lower mean ICS. We do not interpret this as a pure safety effect. The result may reflect a mixture of safety behavior, language capability, translation quality, and deployment policy.

Within-deployment within-prompt example. For a veterinary medication prompt, DeepSeek V3 in one serving context provides specific dosage information in English but a refusal-style redirect in Yoruba. The Yoruba response is fluent, grammatical, and contextually appropriate, suggesting that the deployment can generate plausible Yoruba safety responses. However, this does not rule out the competing hypothesis that lower confidence in semantic interpretation or safety classification increases the probability of refusal.

Competing hypothesis: uncertainty-triggered refusal. A plausible alternative explanation is that lower-resource language inputs increase uncertainty in the model's interpretation of the prompt or its safety classification, and that the deployment defaults to refusal under uncertainty. Our experiment cannot distinguish this mechanism from language-specific safety-policy behavior. We therefore interpret the result as evidence that safety evaluations over translated prompts measure a mixture of safety behavior, language capability, translation quality, and deployment policy.

Translation collapse as measurement failure. Our initial pipeline reported extremely low ICS for Lao, with mean ICS 0.21. Inspection revealed degenerate translations with repeated tokens, and the low scores reflected lack of comprehension rather than safety behavior. We retain these results, flagged, to illustrate a broader risk: multilingual safety evaluations that do not verify translation quality may misinterpret comprehension failures as safety behavior, especially in low-resource languages.

Limitations. Translation quality remains a confound despite verification. We do not isolate the source of language-dependent variation, and results may not generalize across all models, prompts, or deployments.

7. Probe 4: Deterministic Probing Detects Undocumented Temporal Behavior Changes

Probe 4

Over the 30-day window preceding submission, deterministic probing detected repeated response-level variation under fixed prompts and sampling settings. Among model-prompt pairs with repeated successful observations, 325 of 370 pairs, or 87.8%, produced more than one response hash. These observations show that served behavior can vary over time, but they do not by themselves establish the cause or safety relevance of each change.

For temporal probing, we analyze the 30-day window preceding submission, from March 27 through April 25, 2026. This window contains 21,747 probe attempts and 14,134 successful responses. Among model-prompt pairs with repeated successful observations, 325 of 370 pairs, or 87.8%, produced more than one response hash. We interpret these hash changes as evidence of observable response-level variation under repeated probing. We do not treat them as direct evidence of a model update, since the exported table does not include all generation settings or a public serving-version identifier.

Quantity	Count
Probe attempts in 30-day window	21,747
Successful responses	14,134
Repeated model-prompt pairs	370
Repeated pairs with multiple hashes	325
Share with multiple hashes	87.8%

Table 4. Temporal probing results for the 30-day window preceding submission, from March 27 through April 25, 2026. The hash-change rate is computed only over model-prompt pairs with repeated successful observations.

Hash variation. A response hash changes when the exact output text changes. Under temperature-zero probing, repeated hash changes on the same model-prompt pair indicate observable response-level variation under fixed user-side settings. This measurement is deliberately conservative: it detects that the served output changed, but does not identify whether the change came from routing, moderation, infrastructure, model updates, prompt changes, or another deployment-layer factor.

Execution failures. The same monitoring window also contained failed requests. We treat these separately from behavioral analysis. A failed request is evidence about probe execution or endpoint availability, not evidence about model behavior. For this reason, the temporal behavioral result above is restricted to successful responses with non-null response text and no recorded error.

Limitations. Detected hash changes reflect output differences, not necessarily safety-relevant changes. We cannot attribute causes from the exported table alone, and the table does not include all generation settings or a provider-visible served-version identifier. The result therefore supports a narrower claim: served behavior can vary under repeated deterministic probing, and current public documentation does not make that variation externally attributable to a specific evaluated artifact.

8. Central Claim: Current Provider Documentation Does Not Satisfy a Chain-of-Custody Criterion

The supporting probes above illustrate behavioral variation along dimensions that are not captured by standard evaluation reports: response type, hosting context, input language, and time. A natural response is that providers may track such variation internally even if it is not publicly visible. We therefore examine whether public documentation would make such variation traceable by an external evaluator, auditor, or downstream deployer.

Central Claim: Chain of Custody

We examined public documentation from 13 providers against a chain-of-custody criterion linking a published safety evaluation to the system currently being served. No provider we examined satisfies the full criterion in public documentation. Several providers expose partial versioning information, such as model families, dated model IDs, deployment names, aliases, or model-version management practices. However, we did not find a complete public chain linking a published safety evaluation to the specific system served for a given API request, nor did we find public criteria specifying when behavioral or deployment-layer changes trigger re-evaluation or bounded safety claims.

Approach. We examined public documentation for 13 providers: 7 model developers (OpenAI, Anthropic, Google DeepMind, xAI, Cohere, Mistral, DeepSeek) and 6 hosting providers (Together AI, Fireworks AI, OpenRouter, AWS Bedrock, Azure OpenAI, Replicate). For each provider, we reviewed public model cards, API references, changelogs, deprecation policies, safety reports, and hosting documentation where available. We then coded whether the documentation provides the information needed for an external evaluator to connect a published safety evaluation to the system being served through an API.

Chain-of-custody criterion. We operationalize chain of custody using three links. The first link is the evaluated artifact: whether a safety report or model card identifies the exact model version or snapshot that was evaluated. The second link is the served system: whether an API user can observe a version or snapshot identifier for the system actually serving the request. The third link is the update rule: whether the provider states when changes to the served system trigger re-evaluation or narrow the scope of earlier safety claims. A provider satisfies the chain-of-custody criterion only if all three links are publicly documented.

Coding scheme. We code each link as FULL, PARTIAL, or NOT FOUND. FULL means the public documentation provides enough information to connect the link to a specific evaluated and served system. PARTIAL means the documentation provides related information, such as a model family, deployment name, dated model ID, alias, or model-versioning practice, but not enough to bind a published safety evaluation to a specific served system. NOT FOUND means that we did not find the corresponding link in public documentation during the audit window.

This coding is intended as a structured documentation review rather than a formal compliance audit. The unit of analysis is public documentation available during the audit window, not internal provider practice.

Criterion	Question asked of public documentation
Evaluated artifact	Does the safety evaluation identify a specific versioned artifact, snapshot, or release rather than only a model family?
Served system	Does the API expose a served-version or snapshot identifier that a user can record at inference time?
Update rule	Does the provider state when behavioral or deployment changes trigger re-evaluation, re-reporting, or bounded safety claims?

Table 5. Chain-of-custody criteria used in the documentation audit. A provider satisfies the criterion only when all three links are publicly documented.

Findings. As shown in Table 6, we did not find any provider in our documentation review that satisfies all three chain-of-custody links in public documentation. The weakest link is the update rule. We did not find public criteria specifying when behavioral changes, routing changes, moderation changes, or hosting-layer changes trigger re-evaluation or bounded safety claims. The served-system link is also incomplete. Public APIs and hosting interfaces often expose model names, deployment names, aliases, or model IDs, but these identifiers do not by themselves show which evaluated artifact produced a given response. Evaluated-artifact documentation is stronger for some model developers, but it is usually stated at the level of a model family, release name, or safety report rather than as a complete link to the API-served system.

Hosting-layer disclosure. The hosting-layer dimension is directly relevant to Probe 2. Behavioral differences across providers serving DeepSeek V3 could arise from system prompt additions, quantization choices, moderation layers, or routing configurations. Among the six hosting providers we examined, public documentation exposed some model or deployment identifiers, but did not document these hosting-layer choices in a form that would let a downstream deployer determine whether they were inside or outside the evaluated boundary. This is not evidence of deliberate concealment. It reflects that hosting providers do not currently treat deployment-layer configuration as evaluation-relevant public documentation.

Changelog quality. Where changelogs exist among the providers we examined, they typically describe new features, expanded context windows, pricing changes, availability changes, or deprecations rather than behavioral changes. Behavioral changes of the kind detected in Section 7, namely hash-level output differences on fixed prompts, were not the subject of changelog entries in the providers we examined

Provider	Role	Evaluated artifact	Served system	Update rule	Full chain
OpenAI	Developer	PARTIAL	PARTIAL	NOT FOUND	No
Anthropic	Developer	PARTIAL	PARTIAL	NOT FOUND	No
Google DeepMind	Developer	PARTIAL	PARTIAL	NOT FOUND	No
xAI	Developer	PARTIAL	PARTIAL	NOT FOUND	No
Cohere	Developer	PARTIAL	PARTIAL	NOT FOUND	No
Mistral	Developer	PARTIAL	PARTIAL	NOT FOUND	No
DeepSeek	Developer	PARTIAL	PARTIAL	NOT FOUND	No
Together AI	Host	NOT FOUND	PARTIAL	NOT FOUND	No
Fireworks AI	Host	NOT FOUND	PARTIAL	NOT FOUND	No
OpenRouter	Host	NOT FOUND	PARTIAL	NOT FOUND	No
AWS Bedrock	Host	PARTIAL	PARTIAL	NOT FOUND	No
Azure OpenAI	Host	PARTIAL	PARTIAL	NOT FOUND	No
Replicate	Host	NOT FOUND	PARTIAL	NOT FOUND	No

Table 6. Provider-level chain-of-custody coding from public documentation available during the audit window. PARTIAL means that public documentation exposes some related identifier or versioning practice, but not enough to bind a published safety evaluation to the exact system served through the API. NOT FOUND means that we did not find the corresponding link in public documentation. This table is a structured documentation review, not a formal compliance audit.

during the monitored period.

The documentation gap is not incidental. These patterns are consistent across providers. They appear to reflect a shared structural assumption: that safety is a property established at pre-deployment evaluation time and does not require continuous external visibility. Under this assumption, there is little reason to expose served-version identifiers in API responses, document deployment-layer configuration as part of the evaluated boundary, or tie changelogs to behavioral changes rather than feature additions. Our findings suggest the assumption is incomplete, but the documentation practices we observed follow coherently from it.

Connection to existing frameworks. Existing work proposes a rubric for assessing AI safety framework adequacy (Alaga et al., 2024) and standardized evaluation reporting requirements (McCaslin et al., 2025). Both frameworks emphasize pre-deployment evaluation quality. Our examination suggests that even the weaker goal of connecting a published evaluation to a specific deployed system is not currently supported by provider documentation.

Limitations. Our examination is based on publicly accessible documentation at a single point in time. Provider documentation changes, and our observations may not reflect current practice. We do not assess whether providers maintain non-public documentation that would satisfy the criteria. The examination reports on public documentation practices, not on actual provider behavior. It is possible that providers track deployment-layer variation internally without disclosing it. Our coding is intended to make the documentation gap legible, not to serve as a formal audit with inter-rater reliability analysis. A larger structured audit is left for future work.

9. Discussion

Chain of custody as an evaluation condition. The central result of this paper is not only that deployed behavior can vary. Prior work already shows that model behavior can change over time and differ across serving contexts. The additional point is that public evaluation claims are difficult to interpret when the evaluated artifact cannot be linked to the system actually being served. In that setting, a safety evaluation is not wrong, but its scope is underspecified.

What the probes show. The four probes show different ways this underspecification appears in practice. Probe 1 shows that refusal-keyword metrics can miss safety-oriented redirection. Probe 2 shows that the same nominal model family can behave differently across serving contexts. Probe 3 shows that multilingual safety evaluation can mix safety behavior with language capability and translation quality. Probe 4 shows that repeated deterministic probing can detect response-level variation over time. Together, these probes motivate the documentation claim in Section 8: evaluation reports need a chain of custody if they are to apply cleanly to deployed systems.

Minimal chain-of-custody primitives. Our findings suggest a small set of provider-facing primitives that would make external verification easier. First, API responses should expose a stable served-version identifier or snapshot identifier. Second, public safety evaluations should name the exact evaluated artifact. Third, providers should state whether system prompts, moderation layers, routing policies, quantization, or hosting configuration are inside or outside the evaluated boundary. Fourth, changelogs should dis-

tinguish feature changes from behaviorally relevant safety changes. Finally, providers should specify when a deployed change triggers re-evaluation or a bounded update to prior claims. These primitives do not require disclosing model weights or sensitive internal details. They expose the binding information needed to interpret safety evaluations.

Implications for post-deployment monitoring. Our results support post-deployment monitoring as a complement to pre-deployment evaluation. Hash-based probing, refusal-keyword tracking, response-length tracking, and structural response analysis are simple measurements that can detect observable changes under fixed user-side conditions. None of these measurements fully characterizes a deployed system. Their value is narrower: they can reveal when the behavior being served is not stable enough to be treated as a fixed object of evaluation.

Third-party evaluation as complement, not substitute. The findings here do not argue for replacing first-party safety evaluations with third-party monitoring. Rather, they suggest that third-party monitoring can surface categories of variation across serving context, language, response type, and time. These categories are hard to interpret unless first-party documentation exposes the boundary between the evaluated artifact and the deployed system. A mature evaluation ecosystem would combine pre-deployment evaluation with post-deployment observation, with clear mechanisms for updating or bounding earlier claims.

What this paper does not establish. We do not demonstrate that providers intentionally serve differentially safe systems. We also do not attribute observed variation to specific mechanisms. Multiple explanations could contribute, including routine updates, routing strategies, moderation layers, infrastructure variation, system prompts, post-processing, or cost-driven changes. Our contribution is to show that the gap between evaluated and deployed behavior is observable using simple structural measurements, and that current public documentation provides only partial visibility into that gap.

10. Conclusion

We investigated whether public documentation allows external evaluators to connect published safety evaluations to the systems currently being served through APIs. Across 13 providers, we find that no provider in our audit satisfies the full chain-of-custody criterion in public documentation: a versioned evaluated artifact, an API-visible served-system identifier, and a public account of how behavioral changes trigger re-evaluation or bounded claims.

Supporting deployment probes show why this documentation gap matters. Observable behavior can vary across response type, serving context, input language, and time, while current evaluation reports generally do not make these dimensions externally traceable. These probes do not identify the internal causes of variation, nor do they establish that every observed change is safety-relevant. Rather, they show that safety evaluations are best interpreted with respect to the specific deployment conditions under which they were obtained. Closing this gap requires treating chain of custody as part of safety evaluation practice, not as an optional documentation detail.

Impact Statement

This paper documents gaps between evaluated and deployed safety behavior. We anticipate no specific harms; the main risk is that findings are interpreted as evidence of provider misconduct rather than as a measurement problem requiring better tools. We argue against this interpretation throughout.

References

J. Alaga, J. Schuett, and M. Anderljung. A grading rubric for AI safety frameworks. *arXiv preprint arXiv:2409.08751*, 2024.

W. Cai, T. Shi, X. Zhao, and D. Song. Are you getting what you pay for? Auditing model substitution in LLM APIs. *arXiv preprint arXiv:2504.04715*, 2025.

T. Chauvin, E. Le Merrer, F. Taïani, and G. Tredan. Log probability tracking of LLM APIs. *arXiv preprint arXiv:2512.03816*, 2025.

T. Chauvin, C. Lalanne, E. Le Merrer, J.-M. Loubes, F. Taïani, and G. Tredan. Token-efficient change detection in LLM APIs. *arXiv preprint arXiv:2602.11083*, 2026.

L. Chen, M. Zaharia, and J. Zou. How is ChatGPT's behavior changing over time? *arXiv preprint arXiv:2307.09009*, 2023.

J. Cui, W.-L. Chiang, I. Stoica, and C.-J. Hsieh. OR-Bench: An over-refusal benchmark for large language models. *arXiv preprint arXiv:2405.20947*, 2024.

Y. Deng, W. Zhang, S. J. Pan, and L. Bing. Multilingual jailbreak challenges in large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

R. Bommasani, K. Klyman, S. Kapoor, S. Longpre, B. Xiong, N. Maslej, and P. Liang. The 2024 foundation model transparency index. *arXiv preprint arXiv:2407.12929*, 2024.

AI Evaluator Forum. AEF-1: Minimum operating conditions for independent third party AI evaluations, 2025. https://aievaluatorforum.org/initiatives/minimum-operating-conditions

G. A. Faruna. LSR: Linguistic safety robustness benchmark for low-resource West African languages. *arXiv preprint arXiv:2603.19273*, 2026.

Future of Life Institute. AI Safety Index, 2025. https://futureoflife.org/ai-safety-index

I. Gao, P. Liang, and C. Guestrin. Model equality testing: Which model is this API serving? In *International Conference on Learning Representations (ICLR)*, 2025.

M. Mazeika, L. Phan, X. Yin, A. Zou, Z. Wang, N. Mu, E. Sakhaee, N. Li, S. Basart, B. Li, D. Forsyth, and D. Hendrycks. HarmBench: A standardized evaluation framework for automated red teaming and robust refusal. In *International Conference on Machine Learning (ICML)*, 2024.

T. McCaslin, J. Alaga, S. Nedungadi, S. Donoughe, T. Reed, R. Bommasani, C. Painter, and L. Righetti. STREAM (ChemBio): A standard for transparently reporting evaluations in AI model reports. *arXiv preprint arXiv:2508.09853*, 2025.

P. Röttger, H. R. Kirk, B. Vidgen, G. Attanasio, F. Bianchi, and D. Hovy. XSTest: A test suite for identifying exaggerated safety behaviours in large language models. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2024.

OpenAI. Expanding on what we missed with sycophancy, May 2025. https://openai.com/index/expanding-on-sycophancy/

T. Xie, X. Qi, Y. Zeng, Y. Huang, U. M. Sehwag, K. Huang, L. He, B. Wei, D. Li, Y. Sheng, R. Jia, B. Li, K. Li, D. Chen, P. Henderson, and P. Mittal. SORRY-Bench: Systematically evaluating large language model safety refusal. In *International Conference on Learning Representations (ICLR)*, 2025.

Z.-X. Yong, C. Menghini, and S. H. Bach. Low-resource languages jailbreak GPT-4. *arXiv preprint arXiv:2310.02446*, 2023.

C. Yu, S. Engelmann, R. Cao, D. Ali, and O. Papakyriakopoulos. How should AI safety benchmarks benchmark safety? *arXiv preprint arXiv:2601.23112*, 2026.