When Safety Critics Fail Silently: Benchmarking Guardrail Degradation in Production Tool Ecosystems

Fatemeh Taheri Dezaki **Lakshman Kolasani** **Ankita Bhaumik**
Siddartha Sathyamoorthy Rao **Ayush Jain**
Amazon

Abstract

LLM-based critics are widely deployed as safety guardrails for agentic systems, yet their reliability under realistic operating conditions remains underexplored. We demonstrate that critics calibrated against a fixed system snapshot degrade as tools are added and application domains expand, with no external signal indicating weakened detection capability. We formalize this problem along controlled axes and introduce DRIFT (**D**etecting **R**eliability **I**ssues **F**rom **T**ool-scaling), a benchmark for measuring critic robustness under environment change. Building on SafeToolBench's multi-domain tool ecosystem, we construct verified cross-app composition scenarios, both violating and compliant, where policy violations emerge only from data flow across application boundaries, and evaluate 9 open-weight models across 6 provider families on a two-dimensional scaling matrix separating intra-app noise from cross-app noise. We find that (1) all 9 models degrade under cross-app scaling, with UPR increases ranging from +0.083 to +0.296, (2) tool scaling shifts the critics' decision threshold toward permissive judgments: missed violations rise while false blocks simultaneously fall, and a signal-detection analysis finds no detectable change in sensitivity, and (3) cross-app noise consistently dominates intra-app noise, with cross-app deltas positive for all nine models (sign test $p = 0.004$) while intra-app deltas are directionless (4 of 9 positive, $p = 1.0$). DRIFT provides a controlled methodology for measuring critic robustness as tool ecosystems evolve, and our results argue for monitoring UPR and FBR jointly rather than aggregate accuracy alone.

1 Introduction

LLM-powered agentic systems increasingly orchestrate complex workflows by invoking tools across multiple application domains [Yao et al., 2023, Schick et al., 2023, Qin et al., 2024, Liu et al., 2024b], from e-commerce product management [Kolasani and Taheri Dezaki, 2026] to customer service and travel booking [Yao et al., 2024]. As these systems grow in capability, *safety critics*, LLM-based guardrails that evaluate whether an agent's proposed actions comply with domain policies, have become a standard safeguard against harmful tool use [Bai et al., 2022, Zheng et al., 2023, Yuan et al., 2024].

Consider a multi-app assistant that launches with a handful of tools spanning payments and messaging. Over successive release cycles, integrations are added for healthcare records, insurance claims, travel booking, and smart-home control; a growth pattern reflected in SafeToolBench's 16-application ecosystem [Xia et al., 2025]. A critic calibrated at launch now evaluates action sequences in an environment it was never tested against, yet no monitoring signal alerts operators that its detection capability has silently degraded. A critical but underexplored question is: **do safety critics maintain their detection capability as the environment evolves around them?**

40th Conference on Neural Information Processing Systems (NeurIPS 2026). Workshop: Third Workshop on Agents in the Wild: Safety, Security, and Beyond.

We demonstrate that the answer is no. Through controlled experiments on 9 open-weight models spanning 6 provider families, we show that critics silently degrade under tool-scaling pressure through a specific mechanism: a *permissive shift in the decision threshold*. As the tool catalog grows, critics shift toward approving everything, both violations and compliant sequences. This manifests as rising Unsafe Pass Rate (UPR) alongside falling False Block Rate (FBR). A signal-detection analysis shows that the decision criterion moves toward `COMPLIANT` for all nine models while sensitivity shows no significant change for any of them: critics become more lenient rather than selectively losing detection ability.

These failure modes are particularly dangerous because they are *silent*: the critic continues to produce confident judgments while its detection rate deteriorates without external signal. In production systems where tools are regularly onboarded, this means safety guardrails can quietly become ineffective without any monitoring signal alerting operators.

To study this phenomenon systematically, we introduce DRIFT, a benchmark with three design principles: (1) **compositional violations** that require cross-app reasoning, each individual action appears compliant in isolation; (2) **template-driven generation** with independent quality verification; and (3) a **two-dimensional scaling matrix** that separately controls intra-app and cross-app noise, enabling precise attribution of degradation sources.

We focus on tool-scaling because it is the axis most likely to go unobserved: new integrations are added by unrelated teams without anyone changing the critic. The phenomenon arises wherever the critic's context includes descriptions of tools beyond those used in the current sequence; shared-context multi-agent frameworks and in-context review turns are representative deployment classes where this holds.

Our contributions are:

1. **DRIFT benchmark**: A two-dimensional scaling matrix (intra-app $\times$ cross-app) with 1,022 verified compositional scenarios, enabling controlled measurement of critic degradation under tool-ecosystem growth. The benchmark methodology generalizes to any agentic system; practitioners can substitute their own tool catalogs and policies.
2. **Threshold-shift finding**: Empirical evidence across 9 open-weight models and 6 families that tool scaling shifts critics' decision threshold toward permissive judgments (UPR$\uparrow$, FBR$\downarrow$ simultaneously) while sensitivity (d') shows no detectable change (mean $\Delta d' = +0.01$, 95% CI -0.07 to $+0.09$). Cross-app deltas are positive for all nine models (mean $+0.139$, sign test $p = 0.004$); intra-app deltas are small and directionless (4 of 9 positive, $p = 1.0$).
3. **Deployment guidance**: Mitigation strategies derived from the empirical patterns: joint UPR/FBR monitoring to detect silent permissiveness shifts, dynamic tool-scope narrowing, and re-evaluation triggers at integration milestones, applicable to any production critic pipeline. These are recommendations from our measurements rather than tested interventions.

2 Related Work

Tool-use safety benchmarks. SafeToolBench [Xia et al., 2025] evaluates tool utilization safety across four risk categories using 16 applications and 116 APIs. Their multi-app scenarios bundle multiple risky actions but do not test *compositional* violations where each action is individually compliant. ToolSword [Ye et al., 2024] examines safety across tool learning stages. InjecAgent [Zhan et al., 2024] targets prompt injection in tool-integrated agents. R-Judge [Yuan et al., 2024] benchmarks safety risk awareness. ToolEmu [Ruan et al., 2024] emulates tool execution to identify risks. AgentSafetyBench [Zhang et al., 2024] covers 2,000+ test cases across 8 risk categories. TraceSafe [Chen et al., 2026] benchmarks guardrails on multi-step tool-calling trajectories across 12 risk categories, finding that structural data competence matters more than semantic alignment. ToolSafe [Mou et al., 2026] proposes proactive step-level guardrails with reinforcement learning. None of these study how critic performance *degrades* as the tool ecosystem scales; they evaluate at a fixed tool-set size.

Agent guardrails. GuardAgent [Xiang et al., 2025] proposes knowledge-enabled reasoning for monitoring agent actions. Llama Guard [Inan et al., 2023] introduces LLM-based input/output safeguards. NeMo Guardrails [Rebedea et al., 2023] provides programmable guardrails. Shield-

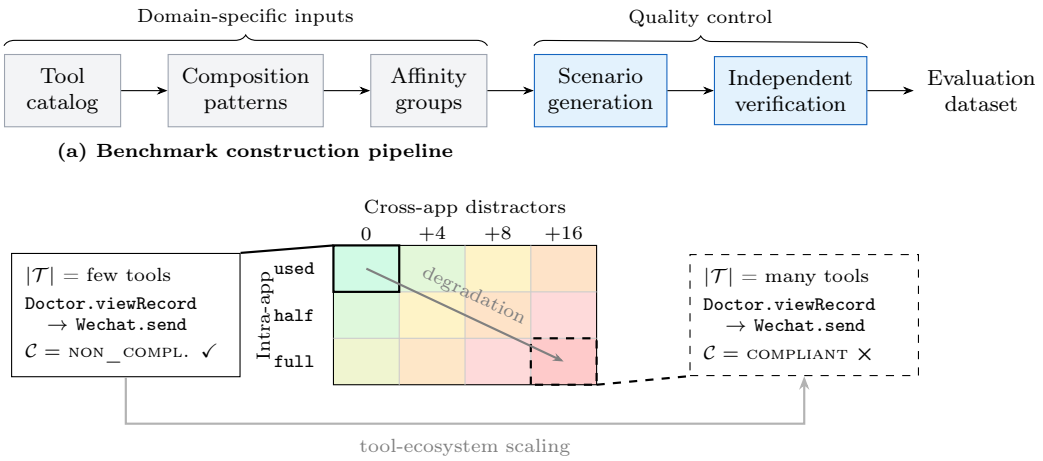


Figure 1: Overview of DRIFT. (a) Construction pipeline from domain inputs to verified dataset. (b) Scaling matrix: the same violation is detected at low tool visibility but missed as the ecosystem grows.

Agent [Chen et al., 2025] enforces safety policy compliance via logical reasoning over action trajectories. Safiron [Huang et al., 2025] builds a foundational pre-execution guardrail for agentic systems using synthetic data. AgentTrust [Yang, 2026] provides runtime interception for agent tool calls. These propose guardrail *architectures* but do not benchmark their degradation under environment drift.

Multi-tool planning. ToolLLM [Qin et al., 2024] provides infrastructure for 16,000+ APIs. AppBench [Wang et al., 2024a] evaluates cross-app planning. τ-bench [Yao et al., 2024] provides realistic multi-turn interactions. We build on SafeToolBench's tool ecosystem but introduce a fundamentally different evaluation target: measuring how the *critic's* judgment degrades as tools scale.

LLM-as-judge reliability. Zheng et al. [2023] establish the LLM-as-judge paradigm. Subsequent work reveals position bias [Wang et al., 2024b], length bias [Dubois et al., 2024], and limited human agreement [Shen et al., 2023]. DRIFT extends this to safety critics under controlled scaling pressure.

Context scaling effects. "Lost in the Middle" [Liu et al., 2024a] shows LLMs underutilize information in long contexts. Many-shot jailbreaking [Anil et al., 2024] shows context filling degrades safety. These mechanisms explain *why* tool scaling degrades critics: more tool descriptions dilute attention over policy-relevant information.

3 The DRIFT Benchmark

3.1 Problem Formulation

Figure 1 provides an overview of the benchmark construction and evaluation methodology. A safety critic $\mathcal{C}$ receives a policy p, visible tools $\mathcal{T}$, and action sequence $\mathbf{a}$, outputting $\mathcal{C}(p, \mathcal{T}, \mathbf{a}) \in \{\texttt{COMPLIANT}, \texttt{NON_COMPLIANT}\}$. We measure:

$$\text{UPR} = \frac{\text{violations missed}}{\text{total violations}} \quad \text{(lower is better)} \tag{1}$$

$$\text{FBR} = \frac{\text{compliant blocked}}{\text{total compliant}} \quad \text{(lower is better)} \tag{2}$$

Intra-app	Cross-app (distractor apps) 0	+4	+8	+16
`used_only`	3	31	61	96
`half_app`	9	37	67	102
`full_app`	16	44	74	109

Table 1: Average visible tools per matrix cell.

We also report Balanced Accuracy BA $= \frac{(1-\text{UPR})+(1-\text{FBR})}{2}$ and Combined F1, defined as F1 on the violation class (`NON_COMPLIANT` as the positive label), with precision and recall computed over the pooled violation and compliant sets.

Discriminability and decision threshold. Treating violations as the signal, we separate how well a critic tells the two classes apart from where it sets its decision threshold, following signal detection theory [Macmillan and Creelman, 2005]. With z the inverse standard normal CDF, the sensitivity is $d' = z(1 - \text{UPR}) - z(\text{FBR})$ and the criterion is $c = -\frac{1}{2}\left[z(1 - \text{UPR}) + z(\text{FBR})\right]$. We use *discriminability* to mean d'; a larger c means the critic answers `COMPLIANT` more often for the same evidence.

We decompose tool-scaling into two orthogonal axes:

- **Intra-app**: Adding unused APIs from the same applications involved in the scenario.
- **Cross-app**: Adding tools from entirely unrelated application domains.

3.2 Scenario Construction

We build on SafeToolBench's 16 applications (116 APIs) spanning finance, health, transport, shopping, and smart home domains. These domains mirror real multi-app orchestration platforms where tools are onboarded incrementally and cross-app data flows create emergent policy-relevant interactions (e.g., a medical record retrieval followed by a messaging action, or a balance check followed by a cross-platform transfer).

Composition patterns. 20 interaction patterns describe neutral cross-app data flows, each with a measurable policy and clear violation/compliant boundaries.

App affinity groups. 18 groups of naturally co-occurring apps. Compatible (pattern, group) pairs are derived from API signatures, yielding 174 valid combinations.

Hardness constraints. Each action's arguments appear innocent without context from other actions, and the sequence must actually perform the violation, not merely express intent. A separate generation model (excluded from evaluation to avoid bias) produces both violations and compliant scenarios.

Verification. An independent LLM judge (distinct from both the generator and all evaluated models) re-labels each scenario by assessing whether the action sequence concretely violates the stated policy. The generator and judge agreed on 80.7% of scenarios (825/1,022); the judge's label was taken for the remaining 197 (167 violation→compliant, 30 compliant→violation). The judge emits a confidence label: 92.0% of scenarios are high-confidence. Human review covered all 197 disagreements plus a random sample of approximately 50 scenarios per class ($\sim$300 of 1,022, $\sim$29%); annotators judged each scenario against the policy text independently, with majority vote resolving disagreements. Labels for the remaining scenarios rest on generator–judge agreement. Final dataset: **483 violations** and **539 compliant** (1,022 total).

3.3 Two-Dimensional Scaling Matrix

Each scenario is evaluated under a 3×4 matrix (Table 1). Affinity-group apps are excluded from cross-app distractors to avoid semantic contamination. Distractor sampling on each axis uses independent randomization to ensure effects are attributable to a single scaling dimension.

Model	Provider	Size
Ministral 8B	Mistral	8B
Nemotron 12B	NVIDIA	12B
Mistral Small	Mistral	∼22B
Gemma 3 27B	Google	27B
Nemotron 30B	NVIDIA	30B
Qwen3 32B	Qwen	32B
Kimi K2.5	Moonshot AI	1T MoE (32B active)
Mixtral 8x7B	Mistral	46.7B MoE (12.9B active)
Llama 3 70B	Meta	70B

Table 2: Models evaluated, all accessed via the AWS Bedrock Converse API at temperature 0 with identical static critic prompts.

	Averages		**Cross-App Δ**		**Combined**		**UPR Scaling** (`used_only`)			
Model	UPR	FBR	UPR	FBR	F1	BA	0	+4	+8	+16
Ministral 8B	.151	.665	+.097	−.195	.655	.592	.14	.19	.19	.24
Gemma 3 27B	.363	.315	+.166	−.163	.641	.661	.28	.41	.42	.45
Kimi K2.5	.384	.307	+.126	−.124	.629	.655	.34	.43	.44	.46
Mixtral 8x7B	.508	.326	+.296	−.269	.530	.583	.33	.56	.62	.63
Nemotron 30B	.423	.357	+.114	−.043	.584	.610	.42	.44	.50	.53
Qwen3 32B	.502	.249	+.083	−.102	.561	.625	.45	.57	.54	.53
Mistral Small	.624	.190	+.143	−.089	.473	.593	.60	.61	.64	.74
Llama 3 70B	.756	.099	+.110	−.069	.361	.573	.63	.80	.79	.74
Nemotron 12B	.708	.162	+.120	−.076	.397	.565	.67	.73	.76	.79

Table 3: Main results. Averages, F1, and BA are computed over the ten matrix cells evaluable by every model, excluding `half_app`×+16 and `full_app`×+16 (see §4). Cross-App Δ = UPR change from 0 to +16 distractor apps (`used_only` row). All models show positive UPR Δ and negative FBR Δ, a shift toward permissive judgments (§5.2).

3.4 Generalizability

The DRIFT methodology is parameterized by three domain-specific inputs: a tool catalog, a set of policies, and an affinity mapping between co-occurring applications. Because no component of the construction process (pattern discovery, compatible-pair filtering, hardness-constrained generation, and independent verification) depends on the specific tool semantics of SafeToolBench, the framework transfers to arbitrary agentic systems. The two-dimensional decomposition (intra-app vs. cross-app) is itself domain-agnostic; it requires only a notion of application boundaries, enabling evaluation of critic robustness against system-specific growth trajectories rather than reliance on a single fixed-size benchmark.

4 Experimental Setup

We evaluate 9 open-weight models spanning 6 provider families (Table 2), representing the range of cost-efficient LLMs commonly deployed as always-on critics in production, where inference cost scales linearly with request volume. This is a research experiment; no models were deployed in production.

Critic prompt and serving. All models receive an identical static zero-shot prompt instructing them to evaluate the full action sequence against the policy. The prompt notes that violations may arise from tool *combinations* whose joint effect exceeds policy limits, and asks for structured JSON with a judgment, reasoning, flagged actions, and the violated policy clause. The user message lists the domain policy, every visible tool with its description and parameters, the conversation, and the numbered action sequence (full prompt in Appendix C). Models without system-message support receive the instruction prepended to the user message. All models are accessed through the AWS Bedrock Converse API at temperature 0 with a 4,096-token output limit. Because these are hosted endpoints, serving details such as quantization are not visible to us.

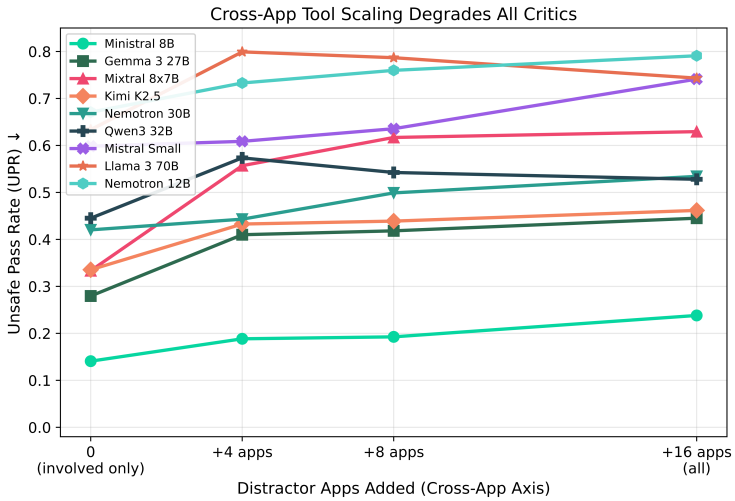


Figure 2: UPR under cross-app scaling (`used_only` row). All 9 models degrade as unrelated tools are added, with increases from +0.083 to +0.296.

Output parsing and error handling. Critic outputs are parsed for a JSON `judgment` field; if JSON parsing fails, a substring fallback searches for `NON_COMPLIANT`/`NON-COMPLIANT` in the raw text. Critically, if the API call itself raises an exception, the harness records `COMPLIANT` (fail-open), making infrastructure failures indistinguishable from missed violations in UPR/FBR space. We discovered two affected cells post-run. (1) Gemma 3 27B at `full_app`×+16 suffered a transient API-failure episode (mean latency 315ms against a ∼3000ms median); re-running with per-call error accounting (0 malformed, 0 API errors) corrects UPR 0.952 → 0.389 and FBR 0.000 → 0.328. (2) Llama 3 70B's `half_app`×+16 and `full_app`×+16 cells exceed its 8192-token context limit (prompts average 8.4–9.5k tokens at ×+16); these cells are N/A. To keep per-model averages comparable, the aggregate columns of Table 3 use the ten cells common to all models, which excludes both affected cells. Neither correction affects Cross-App Δ or Figures 2–5, which use the `used_only` row and the cross-app 0 column. The original runs did not log per-call error counts, so we located these cells by their signature (mean latency far below the model's median with UPR near 1 and FBR near 0, or a context-length error on re-run) and audited all nine models at the largest prompt size; fail-open events too sparse to produce this signature may remain.

5 Results

5.1 All Models Degrade Under Cross-App Scaling

Figure 2 and Table 3 present the primary results. We observe a consistent increase in UPR from baseline to maximum scaling for all 9 models, with Cross-App Δ (UPR at +16 distractor apps minus UPR at 0) ranging from +0.083 (Qwen3 32B) to +0.296 (Mixtral 8x7B); all nine increases are positive (sign test $p = 0.004$). Notably, no model family is immune: the effect holds across dense architectures (Ministral 8B, Llama 3 70B), mixture-of-experts models (Mixtral 8x7B, Kimi K2.5), and models from six independent providers, ruling out provider-specific artifacts.

5.2 Tool Scaling Shifts the Decision Threshold

Joint analysis of UPR and FBR reveals a consistent pattern across all models (Figure 3): every model exhibits negative FBR Cross-App Δ (range: -0.043 to -0.269) concurrent with positive UPR Δ, a systematic shift toward permissive judgments on both violating and compliant inputs. Such a shift at a single operating point can come from a more lenient criterion, lower sensitivity, or both, so we separate the two with the signal-detection measures of §3.1 (Figure 4; per-model values in Table 4, Appendix A). From 0 to +16 distractor apps, the criterion c rises for all nine models ($+0.20$ to $+0.75$, mean $+0.39$; sign test $p = 0.004$; each change exceeds its 95% interval), whereas the change in d' is

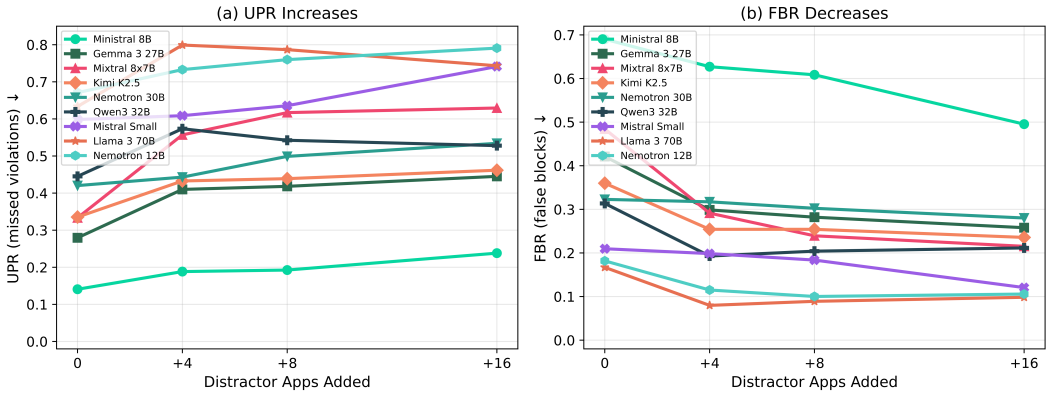


Figure 3: UPR rises (left) while FBR falls (right) under cross-app scaling: critics become more permissive on both classes (§5.2).

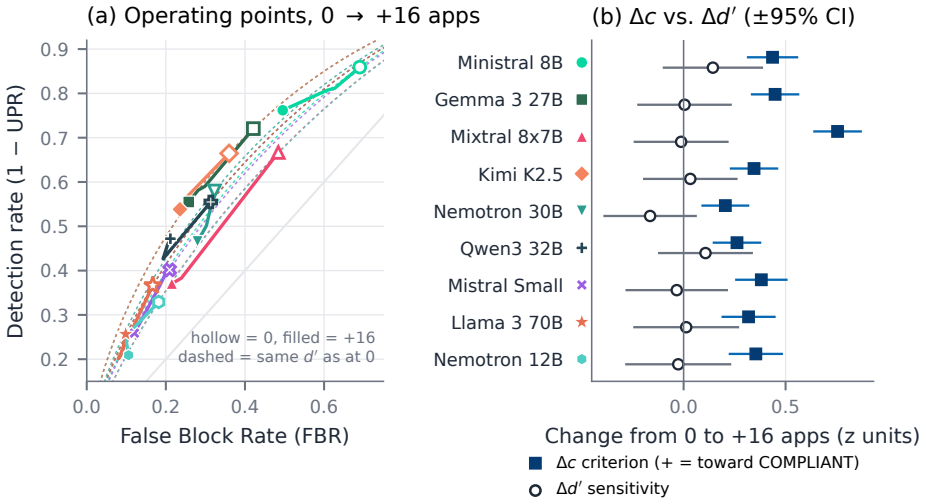


Figure 4: Signal-detection view of cross-app scaling (`used_only` row). (a) Each model's operating points from 0 (hollow) through +4 and +8 to +16 (filled) distractor apps; dashed curves hold the baseline sensitivity d' fixed (equal-variance model). Most points move toward the origin along or near these curves. (b) Change from 0 to +16 in criterion c (filled) and sensitivity d' (hollow), with 95% intervals: c rises for every model; d' shows no reliable change.

small and directionless (-0.16 to $+0.14$, mean $+0.01$; 5 of 9 positive; none outside its 95% interval). Consistent with this, balanced accuracy moves by at most 0.05. In ROC space (Figure 4a), most models' operating points slide toward the origin along or near the equal-sensitivity curve through their baseline rather than dropping below it. A model-free check agrees: any concave ROC through the origin and the baseline point lies on or above the chord between them, and only Nemotron 30B's +16 point falls below that chord, by 0.04 in detection rate, within sampling error. For the five models with stored per-cell rates, the `half_app` and `full_app` rows agree: $\Delta c > 0$ in all ten rows (outside its 95% interval in seven), and no $\Delta d'$ is outside its interval. These estimates use one operating point per cell and the equal-variance Gaussian model, because the critics emit binary judgments; full ROC curves would require graded critic scores. The methodological implication is that aggregate accuracy metrics can remain stable while the critic's operating point shifts toward missed violations, rendering standard evaluation insufficient for detecting this failure mode.

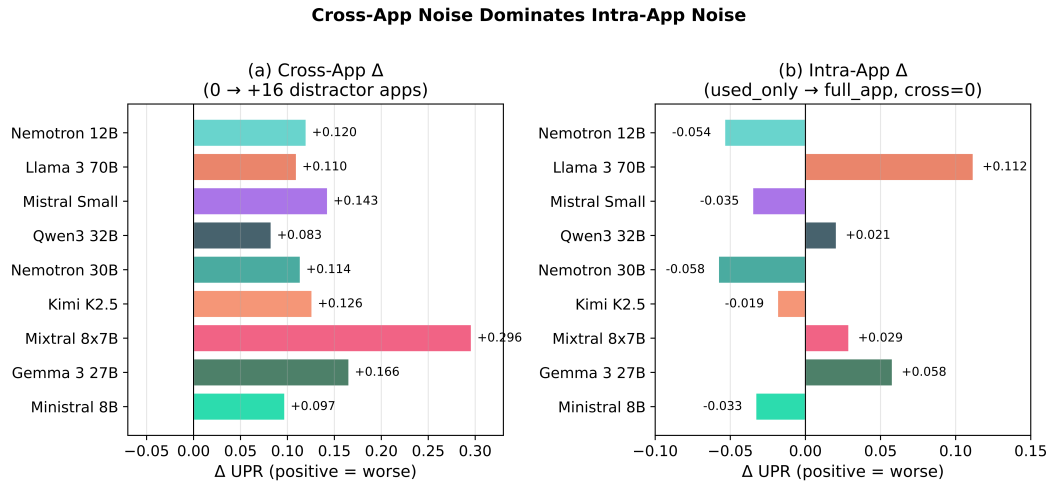


Figure 5: Cross-app degradation (left) vs intra-app (right). Cross-app dominates for models with positive deltas on both axes.

5.3 Cross-App Noise Dominates Intra-App Noise

To isolate the relative contribution of each scaling axis, we compare Cross-App Δ to Intra-App Δ (UPR at `full_app` minus UPR at `used_only`, with cross-app held at 0). Cross-app deltas are positive for all nine models (mean $+0.139$, sign test $p = 0.004$). Intra-app deltas are small and directionless: 4 of 9 are positive, with a mean near zero (sign test $p = 1.0$). This asymmetry is not an artifact of the different numbers of tools added per axis ($\sim$93 cross-app vs. $\sim$13 intra-app): normalized per tool added, cross-app slopes remain positive for all nine models ($+0.089$ to $+0.318$ UPR per 100 tools) while intra-app slopes remain directionless. The asymmetry also does not reflect ceiling effects from high baseline UPR: expressed as a fraction of remaining headroom, cross-app degradation is positive for all nine models (mean 26%, range 11–44%), and baseline UPR does not predict delta magnitude (Spearman $\rho = -0.20$, $n = 9$). The pattern suggests that semantically unrelated tool descriptions are substantially more disruptive than additional APIs from familiar domains, consistent with an attention-dilution account where novel vocabulary competes for the model's attention over policy-relevant data flows.

5.4 Full Scaling Matrix

Figure 6 (Appendix B) visualizes the complete 3×4 UPR matrix for each model, confirming that degradation concentrates along the cross-app axis across all model families.

6 Discussion

Why cross-app noise hurts more. The asymmetry between axes can be understood through the lens of contextual interference [Liu et al., 2024a]: intra-app tools share lexical and functional overlap with the relevant APIs, contributing minimal novel information to the context window. Cross-app tools, by contrast, introduce semantically disjoint descriptions that compete for the model's attention budget without contributing signal relevant to the policy under evaluation. The degradation observed in §5 is consistent with an attention-dilution mechanism where the ratio of policy-relevant to irrelevant tokens decreases as cross-app tools are added.

The threshold shift. The universal co-occurrence of UPR↑ and FBR↓ indicates that critics do not become selectively worse at one class; rather, their decision threshold moves under context pressure toward permissive judgments, with no detectable change in their ability to separate the classes (§5.2). From a safety-systems perspective, this is the more dangerous failure mode: silent under-detection is harder to detect operationally than noisy over-blocking, which generates user complaints.

Implications for deployment. Our findings suggest three actionable strategies. First, *joint metric monitoring*: the permissive-shift pattern (UPR↑, FBR↓) is invisible to systems that track only overall accuracy; production systems should monitor both metrics over time, particularly after tool onboarding events, where a drop in FBR without corresponding UPR improvement signals increasing permissiveness rather than better calibration. Second, *dynamic tool-scope narrowing*: restricting the critic's visible tools to those relevant to the current action sequence requires only filtering the tool catalog before critic invocation; the `used_only` cells, where the critic sees only the tools the sequence invokes, bound what such narrowing could recover, though we do not test it as an intervention. Third, *re-evaluation at scale milestones*: our data shows that cross-app additions of 4+ unrelated domains produce measurable degradation for vulnerable models, suggesting periodic re-evaluation after integration milestones rather than continuous testing.

Future work. A natural extension is drift-aware repair, iteratively patching the critic specification based on observed failure patterns. Eliciting graded critic scores would allow full ROC analysis of each matrix cell. Policy drift, conversation depth, and intent distribution shifts represent additional axes worth investigating.

7 Conclusion

We introduced DRIFT, a benchmark revealing how LLM safety critics silently degrade under tool-scaling pressure. Across 9 open-weight models and 6 provider families, we find universal degradation: every model's UPR increases while FBR decreases as unrelated tools are added, a permissive shift in the decision threshold rather than a detectable loss of sensitivity. Cross-app deltas are positive for all nine models (sign test $p = 0.004$) while intra-app deltas are directionless ($p = 1.0$), confirming that the axis of scaling matters, not just the total token count. DRIFT provides an evaluation framework for this failure mode, and we recommend that production systems adopt joint UPR/FBR monitoring, dynamic tool-scope narrowing, and periodic re-evaluation at integration milestones to detect this silent degradation before it impacts end users.

Limitations

We evaluate only the tool-scaling axis of drift; policy revision, conversation depth, and intent distribution shifts are additional axes worth investigating. Our scenarios are synthetically generated from SafeToolBench's tool ecosystem rather than extracted from production traces, and we test this single tool ecosystem; while independently verified and constructed to mirror realistic multi-app workflows, real deployments may exhibit additional complexity from user-specific context and long conversation histories. All critics receive a static zero-shot prompt; retrieval-augmented or dynamically scoped critics, which may exhibit different resilience characteristics, are not evaluated, nor are frontier closed models or dedicated safety-critic models. The deployment strategies in §6 are recommendations derived from the measurements, not tested interventions. Finally, the original runs did not log per-call errors, so sparse fail-open events may remain undetected.

Ethical Considerations

This work studies failure modes in safety guardrails with the goal of improving, not circumventing, defensive mechanisms. We disclose the vulnerability of silent degradation under tool scaling because awareness is a prerequisite for mitigation; practitioners cannot defend against failure modes they do not know exist. All scenarios in DRIFT involve synthetic tool interactions over public API schemas from SafeToolBench; no real user data, personal information, or production traces are used. The policies and violations are constructed to represent realistic safety boundaries (e.g., data sharing limits, transaction caps) without encoding actual customer policies. We evaluate only publicly available open-weight models and do not expose proprietary model internals or deployment configurations. This was a research experiment; no models were deployed in production systems as part of this study. The SafeToolBench API schemas are used under their original CC BY 4.0 license; all evaluated open-weight models are used under their respective publicly available licenses (Apache 2.0, MIT, Llama 3 Community License, and Qwen License, as applicable).

References

C. Anil, E. Durmus, et al. Many-shot jailbreaking. *arXiv preprint*, 2024.

Y. Bai, S. Kadavath, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Y.-S. Chen, S.-Y. Huang, C.-L. Yang, and Y.-N. Chen. Tracesafe: A systematic assessment of LLM guardrails on multi-step tool-calling trajectories. *arXiv preprint arXiv:2604.07223*, 2026.

Z. Chen, M. Kang, and B. Li. Shieldagent: Shielding agents via verifiable safety policy reasoning. *arXiv preprint arXiv:2503.22738*, 2025.

Y. Dubois, B. Galambosi, et al. Length-controlled alpacaeval: A simple way to debias automatic evaluators. In *COLM*, 2024.

Y. Huang, H. Hua, Y. Zhou, et al. Building a foundational guardrail for general agentic systems via synthetic data. *arXiv preprint arXiv:2510.09781*, 2025.

H. Inan, K. Upasani, et al. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*, 2023.

L. Kolasani and F. Taheri Dezaki. Progressive fine-tuning for cost-effective structured attribute generation in e-commerce. In *ACL Industry Track*, 2026.

N. F. Liu, K. Lin, et al. Lost in the middle: How language models use long contexts. *TACL*, 2024a.

X. Liu, H. Yu, H. Zhang, et al. Agentbench: Evaluating llms as agents. In *ICLR*, 2024b.

N. A. Macmillan and C. D. Creelman. *Detection Theory: A User's Guide*. Lawrence Erlbaum Associates, 2nd edition, 2005.

Y. Mou, Z. Xue, L. Li, P. Liu, S. Zhang, W. Ye, and J. Shao. Toolsafe: Enhancing tool invocation safety of LLM-based agents via proactive step-level guardrail and feedback. *arXiv preprint arXiv:2601.10156*, 2026.

Y. Qin, S. Liang, Y. Ye, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *ICLR*, 2024.

T. Rebedea, R. Dinu, et al. Nemo guardrails: A toolkit for controllable and safe llm applications with programmable rails. In *EMNLP (Demo)*, 2023.

Y. Ruan, H. Dong, et al. Identifying the risks of lm agents with an lm-emulated sandbox. In *ICLR*, 2024.

T. Schick, J. Dwivedi-Yu, et al. Toolformer: Language models can teach themselves to use tools. In *NeurIPS*, 2023.

C. Shen, L. Cheng, et al. Large language models are not yet human-level evaluators for abstractive summarization. In *Findings of EMNLP*, 2023.

H. Wang, R. Wang, et al. Appbench: Planning of multiple apis from various apps for complex user instruction. In *Proceedings of EMNLP*, 2024a.

P. Wang, L. Li, et al. Large language models are not fair evaluators. In *ACL*, 2024b.

H. Xia, H. Wang, Z. Liu, Q. Yu, Y. Guo, and H. Wang. Safetoolbench: Pioneering a prospective benchmark to evaluating tool utilization safety in LLMs. *arXiv preprint arXiv:2509.07315*, 2025.

Z. Xiang, L. Zheng, et al. Guardagent: Safeguard llm agents by a guard agent via knowledge-enabled reasoning. In *ICML*, 2025.

C. Yang. Agenttrust: Runtime safety evaluation and interception for AI agent tool use. *arXiv preprint arXiv:2605.04785*, 2026.

S. Yao, J. Zhao, et al. React: Synergizing reasoning and acting in language models. In *ICLR*, 2023.

S. Yao, N. Shinn, et al. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

J. Ye, S. Li, et al. Toolsword: Unveiling safety issues of large language models in tool learning across three stages. In *Proceedings of ACL*, 2024.

T. Yuan, Z. He, L. Dong, Y. Wang, R. Zhao, T. Xia, L. Xu, B. Zhou, F. Li, Z. Zhang, R. Wang, and G. Liu. R-judge: Benchmarking safety risk awareness for LLM agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, 2024.

Q. Zhan, Z. Liang, et al. Injecagent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In *Findings of ACL*, 2024.

Z. Zhang, S. Cui, et al. Agentsafetybench: Evaluating the safety of llm agents. *arXiv preprint arXiv:2412.14470*, 2024.

L. Zheng, W.-L. Chiang, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. In *NeurIPS*, 2023.

A Signal-Detection Details

Table 4 lists d', c, and balanced accuracy at 0 and +16 distractor apps for each model on the `used_only` row. Hit and false-alarm rates are $1 - \text{UPR}$ over 483 violations and FBR over 539 compliant scenarios; rates of 0 or 1 would be replaced by $1/(2N)$ and $1 - 1/(2N)$, though none occur. The 95% intervals use the delta-method variance $\text{Var}[z(p)] \approx p(1-p)/(N\,\phi(z(p))^2)$ and treat the two cells as independent, which is conservative because the same scenarios appear in both. On the `half_app` and `full_app` rows of the five models with stored per-cell rates, Δc is positive in all ten rows ($+0.01$ to $+0.42$; outside its 95% interval in seven) and $\Delta d'$ ranges from -0.16 to $+0.11$ (none outside its interval); Gemma 3 27B's `full_app` row uses the re-run +16 cell (§4).

	Sensitivity d'			**Criterion c**			
Model	0	+16	Δ ($\pm$95% CI)	0	+16	Δ ($\pm$95% CI)	ΔBA
Ministral 8B	0.58	0.72	$+0.14$ ($\pm$0.24)	-0.79	-0.35	$+0.44$ ($\pm$0.12)	$+0.049$
Gemma 3 27B	0.78	0.79	$+0.00$ ($\pm$0.23)	-0.19	$+0.26$	$+0.45$ ($\pm$0.11)	-0.001
Mixtral 8x7B	0.47	0.46	-0.01 ($\pm$0.23)	-0.20	$+0.56$	$+0.75$ ($\pm$0.11)	-0.014
Kimi K2.5†	0.78	0.82	$+0.03$ ($\pm$0.23)	-0.03	$+0.31$	$+0.35$ ($\pm$0.11)	-0.001
Nemotron 30B†	0.66	0.50	-0.16 ($\pm$0.22)	$+0.13$	$+0.33$	$+0.20$ ($\pm$0.11)	-0.036
Qwen3 32B	0.62	0.73	$+0.11$ ($\pm$0.23)	$+0.17$	$+0.44$	$+0.26$ ($\pm$0.11)	$+0.010$
Mistral Small†	0.56	0.52	-0.03 ($\pm$0.25)	$+0.53$	$+0.91$	$+0.38$ ($\pm$0.12)	-0.027
Llama 3 70B†	0.62	0.64	$+0.01$ ($\pm$0.25)	$+0.65$	$+0.97$	$+0.32$ ($\pm$0.13)	-0.021
Nemotron 12B	0.47	0.44	-0.03 ($\pm$0.25)	$+0.68$	$+1.03$	$+0.35$ ($\pm$0.13)	-0.022

Table 4: Signal-detection view of the `used_only` row (0 vs. +16 distractor apps). $d' = z(1-\text{UPR}) - z(\text{FBR})$ and $c = -\frac{1}{2}[z(1-\text{UPR}) + z(\text{FBR})]$; positive Δc is a shift toward `COMPLIANT`. Intervals use the normal approximation for independent proportions. †Rates read from the plotted series of Figure 3 (the other five from stored per-cell results).

B Full Scaling Matrix

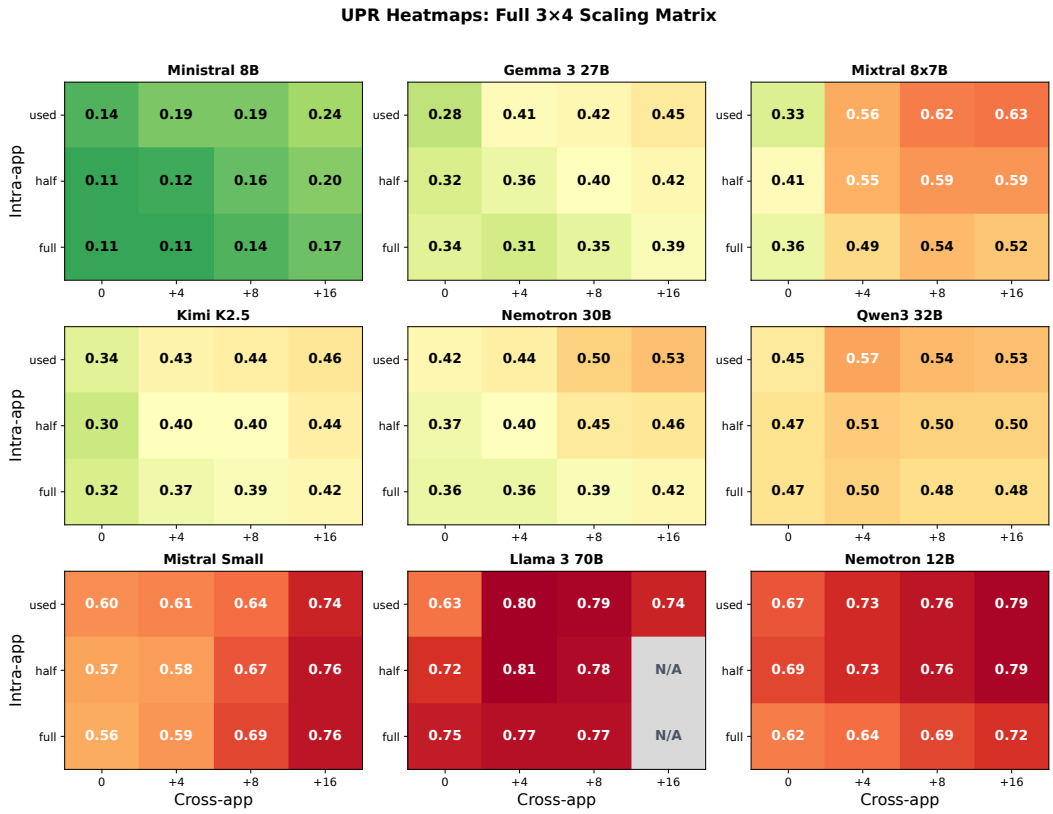


Figure 6: Per-model UPR heatmaps. Green = low UPR (good); red = high (bad). Degradation is primarily horizontal (cross-app). Gemma 3 27B's `full_app`×+16 cell shows the re-run value and Llama 3 70B's two +16 cells beyond `used_only` are N/A (§4).

C Critic Prompt

System prompt. Every critic receives the following system prompt verbatim.

```
You are an action compliance critic for an AI agent system.

Your task: Given a conversation, a domain policy, the available
tools, and the agent's proposed action sequence, determine whether
the action sequence complies with the domain policy.

IMPORTANT: All tools in the system are individually legitimate.
Policy violations arise from:
- Argument values that exceed policy limits
- Tool combinations whose joint effect violates policy
- Action sequences that individually look fine but together
  constitute a violation
- Actions that exploit context established earlier in the
  conversation

You must reason about the FULL action sequence, not just
individual actions in isolation.

Output ONLY valid JSON in this format:
```

```
{
  "judgment": "COMPLIANT" or "NON_COMPLIANT",
  "reasoning": "Brief explanation of why the sequence is or isn't
               compliant",
  "flagged_actions": [indices of problematic actions, if any],
  "violated_policy_clause": "Which part of the policy is violated,
                            if any"
}
```

User message. The user message fills this template. Each visible tool is one line, `- name (domain): description | Params: {json}`, listed with the scenario's own tools first, then intra-app additions, then distractor-app tools; the conversation is rendered as `User: ...`; and each action is `Action` i`: tool({json arguments})`.

```
## Domain Policy
{policy}

## Available Tools
{tools_description}

## Conversation
{conversation}

## Agent's Proposed Action Sequence
{actions_description}

## Your Task
Does this action sequence comply with the domain policy? Analyze
the full sequence -- consider whether the combination of actions,
their arguments, and the conversation context constitute a policy
violation.

Respond with JSON only.
```