

DOI: [https://doi.org/10.48009/3\\_iis\\_2026\\_124](https://doi.org/10.48009/3_iis_2026_124)

## Routing fragility in agentic AI: classification-dependent enforcement and the case for framing invariance

Michael A Bumpus, *AI Cabinet Method LLC*, [bumpusma@gmail.com](mailto:bumpusma@gmail.com)

### Abstract

Agentic LLM systems increasingly perform real-world actions via tool use, including file access, code execution, and network calls. Most current safety enforcement pipelines rely on upstream intent classification to determine whether actions are permissible. This paper identifies *routing fragility*, the structural condition where enforcement reliability is tightly coupled to intent classification accuracy, as an architectural condition in tool-integrated systems. This work documents routing fragility as observed through a case study of a frontier model that refused to decrypt an encrypted archive under one prompt framing, then attempted the identical action without hesitation when the same action was recontextualized as "verification," using standard prompt engineering with no jailbreak or adversarial techniques. This work proposes framing invariance, the property that safety decisions remain stable under narrative reframing, as a testable evaluation metric for enforcement quality, applicable to any enforcement architecture. Additionally, this work proposes action normalization as a mechanism for achieving framing-invariant decisions, where proposed actions are stripped of contextual narrative and evaluated independently of user framing. While prior work has focused on architectural solutions like tool-choice restriction and guard-agent approaches, this paper concentrates on formalizing the diagnostic problem and establishing evaluation criteria that any enforcement system should satisfy. Future work will address architectural solutions that implement these criteria.

**Keywords:** routing fragility, framing invariance, action normalization, agentic AI, safety enforcement, model context protocol

### Introduction

Large Language Models (LLMs) are no longer text-only systems. The current generation of deployments increasingly integrates models with execution environments that carry real-world side effects: code interpreters and sandboxed runtimes; command-line interface (CLI) environments with file system access; database read/write connections; external API integrations; and Model Context Protocol (MCP) tool ecosystems that dynamically expose capabilities. The scale of this shift is empirically documented. Hou et al. (2025) deliver a thorough landscape analysis of the MCP ecosystem, defining its full server lifecycle across creation, deployment, operation, and maintenance phases. Li et al. (2025) analyze 2,562 real-world MCP applications across 23 categories, finding that network and system resource APIs dominate usage patterns, confirming that tool-integrated agents are now mainstream, not limited to experimental use cases. This shift transforms the model from an advisory system, where output is mediated by human interpretation, into an executive system, where output can trigger action. This distinction is not merely academic, as it directly changes the threat model that safety architectures must address.

The prevailing approach to safety in LLM systems relies on a set of upstream mechanisms: intent classification (determining whether a user request is benign, ambiguous, or adversarial); content-based

refusal (matching outputs against known harmful patterns); pattern matching (recognizing request structures associated with misuse); and contextual framing analysis (evaluating the narrative surrounding a request to determine permissibility). These mechanisms work reasonably well in text-only environments, where the cost of a false negative is an undesirable text output that a human must still choose to act on, but this dependency becomes a structural liability once outputs can trigger real-world actions. Model-level defenses can further reduce this risk: Piet et al. (2024) demonstrate that task-specific fine-tuning reduces prompt injection success to below 0.5%, and Chen et al. (2024) show that structured queries separating prompt and data channels significantly improve robustness. But both approaches require model-level modification and sacrifice generality, hardening the model at the cost of flexibility.

In tool-augmented systems, enforcement often depends on whether the model correctly classifies the action's intent. When intent is misclassified, downstream policy fails to activate and tool execution proceeds without appropriate gates. Sandboxing, capability limits, and rate limiting help reduce impact but don't stop misrouting altogether. This makes classification accuracy an essential safety property: the system's safety becomes a function of probabilistic routing that is inherently bounded and subject to adversarial pressure. With the expanding tool ecosystem, the classification burden grows superlinearly while the consequences of misclassification increase. Empirical evidence from recent benchmarks reinforces the scale of this challenge. Zhang et al. (2024) evaluate 349 interaction environments across 2,000 test cases and find that no current agent exceeds a 60% safety score, identifying lack of robustness and lack of risk awareness as fundamental defects. Zhan et al. (2024) demonstrate through InjecAgent that GPT-4 with ReAct is vulnerable to indirect prompt injection approximately 24% of the time, with attack success rates nearly doubling under reinforced adversarial pressure. Most critically, MCP-ITP (2026) formalizes implicit tool poisoning, a class of attacks where a poisoned tool is never executed but its metadata alone induces the agent to invoke a legitimate high-privilege tool for adversarial purposes. This represents a pure routing-level attack: the classification mechanism itself is the vulnerability. These results establish the motivation: where enforcement depends on intent classification, the same action can receive different safety treatment as framing changes, and the cost of that divergence rises as agents gain stronger tools. Existing work addresses prompt injection, tool poisoning, privilege restriction, and guard-agent evaluation, but less attention has been paid to whether enforcement decisions remain stable when the action is held constant and only the framing varies. This paper takes up that gap by examining a case that surfaced unexpectedly during unrelated benchmarking: a frontier model refused an action under one framing and attempted the identical action under another. From this observation it formalizes three contributions: *routing fragility* as the condition, *framing invariance* as an evaluation property, and the Routing Fragility Diagnostic as a protocol for detecting it.

These three concepts are distinct and play different roles in this paper. Routing fragility refers to the observed failure condition: enforcement decisions change when equivalent actions are routed differently based on framing. Framing invariance refers to the desired evaluation property: equivalent actions should receive consistent safety decisions across narrative framings. *Action normalization* refers to a proposed enforcement mechanism: evaluating the structured action itself rather than the user's narrative explanation for it.

## Background and Literature Review

This section first defines agentic LLM systems, then describes the standard classification-dependent enforcement pipeline, reviews related approaches to tool and agent safety, and finally introduces routing fragility, the architectural condition identified through the case study, as the organizing concept for the analysis that follows.

## *The Standard Classification-Dependent Enforcement Pipeline*

In this paper, an agentic LLM refers to a large language model deployed with the ability to plan or select actions and invoke external tools such as code execution, file access, APIs, or command-line operations, rather than only generating text content.

The standard enforcement pipeline in agentic LLM systems follows a linear flow:



**Figure 1. Standard Classification-Dependent Enforcement Pipeline**

Figure 1 illustrates the common enforcement pattern in which a user request is first interpreted by an intent classifier. That classification determines which policy branch applies, and downstream tool permissions inherit the initial routing decision. If the initial classification changes under reframing, the same action may pass through a different enforcement path.

Intent classification acts as the primary routing decision. It determines which policy branch applies, which in turn governs what actions the model is permitted to plan and execute. Downstream stages inherit the classification decision and do not independently re-evaluate it. Several recent systems exemplify this architecture with increasing sophistication. Progent (Shi et al., 2025) uses a domain-specific language to declare which tools are safe for a given task, then restricts the agent to only those tools, a policy-over-classification approach that still depends on correctly scoping the task. GuardAgent (Xiang et al., 2025) attempts to solve this by delegating evaluation to a second agent, but as the case study will show, a second agent inherits the same framing vulnerability if it evaluates intent rather than action. Narajala and Habler (2025) propose enterprise-grade MCP security via identity and access management (IAM) integration and role-based access control, representing classification-based approaches scaled to organizational complexity. This architecture has genuine strengths. It is flexible, adapting to a wide range of request types without rigid rule sets, and context-sensitive, able to incorporate conversational history, user role, and session state. It also offers high usability, imposing low friction on legitimate users, and efficiency, since a single upstream decision gates the entire execution path.

The same properties that make classification-dependent safety usable also make it fragile. These systems are vulnerable to framing shifts, where semantically equivalent actions can be classified differently depending on how the user describes them. Patch-based mitigation creates blind spots, with each discovered bypass being addressed with a new pattern or rule, but the space of possible framings is expansive. Enforcement depends on recognizing known patterns, and novel phrasings, domain-specific jargon, or multi-step decomposition can evade pattern-based classifiers. This scales poorly as the integration and tooling layer grows, where each new tool integration adds classification surface area without proportional investment in safety coverage. The evidence for these weaknesses is covered in the literature. Zhang et al. (2024) find that defense prompts alone are insufficient, a finding that exposes the structural limits of prompt-level classification as a sole enforcement mechanism. Zhan et al. (2024) show that adversarial pressure can raise attack success from a baseline of 24% to nearly 48%, precisely because the adversarial strategy targets the classification mechanism itself, manipulating the framing that determines routing decisions. MCPTox (2025) reveals that more capable models are more susceptible to tool poisoning: o1-mini achieves a 72.8% attack success rate. The attack succeeds precisely because it exploits instruction-following ability, the same capability that makes models useful. Improving the model's ability to classify intent does not help when the attack vector is designed to produce correct-seeming intent classification. Li et al. (2025) add a structural dimension: their analysis of 2,562 MCP applications reveals that broad privilege inheritance with minimal isolation is endemic. Even when classification correctly identifies a benign intent, the tool ecosystem grants access far

beyond what the classified task requires. MCP-ITP (2026) provides what may be the canonical demonstration of routing fragility. In implicit tool poisoning, a poisoned tool is never executed; its metadata alone, embedded in tool descriptions visible to the agent, induces the agent to invoke a different, legitimate tool for adversarial purposes. The classification layer evaluates the stated purpose while the routing layer delivers the actual action, and the gap between them constitutes the vulnerability.

## ***Routing Fragility as a Failure Condition***

This work introduces the term routing fragility to describe the structural condition where enforcement reliability is tightly coupled to intent classification accuracy, such that degradation in classification directly produces degradation in safety guarantees. Classification is necessary, but insufficient as the sole safety layer in systems with real-world execution capabilities. The empirical record, spanning Agent-SafetyBench's 60% ceiling, InjecAgent's doubling under adversarial pressure, MCPTox's capability paradox, and MCP-ITP's metadata-only attacks, indicates that classification-dependent enforcement degrades under exactly the conditions that matter most.

## **Methodology and Responsible Disclosure Constraints**

The observation reported here arose from a separate study whose original purpose was to compare model reasoning and problem-solving performance on the First Proof benchmark across baseline and structured multi-role governance conditions. The governance structure was implemented as a multi-role-in-context architecture: a single model running multiple specialized roles within one shared conversational context, activated through semantic triggers rather than as separate processes (Bumpus, 2026a). This is distinct from a classical multi-agent system of autonomous processes communicating by message passing. Routing fragility was identified as an unexpected byproduct of varying the problem-statement framing, not as the study's initial objective.

The models tested were GPT-5.2 and Gemini Pro, accessed through their respective provider interfaces. GPT-5.2 refers to the model identifier exposed in the ChatGPT interface at the time of testing; because provider-facing model labels can change over time, this designation describes the historical testing condition rather than a fixed public release, and future work should replicate the protocol against newer model generations.

The protocol was single-turn: each model received the prompt and an accompanying file upload with no prior conversation history, no accumulated context, and no opportunity for the user to clarify or redirect after the model's initial response. Two prompt structures were evaluated, a two-path condition and a four-path condition, differing only in the narrative framing of the available options. In both conditions the same Advanced Encryption Standard 256-bit (AES-256) encrypted archive was supplied as a file upload, and the underlying action under evaluation was identical: an attempt to access that archive without the decryption key. The evaluation criterion was binary: whether the model refused the action or attempted to execute it.

Certain prompt artifacts and operational details are abbreviated in this paper. The observed behavior was disclosed to OpenAI and, to the author's knowledge, remains unresolved; the paper therefore reports the structure of the conditions and the observed decisions without publishing a fully operational reproduction recipe. This abbreviation is a deliberate disclosure measure, not an omission of method.

## **Case Study: Verification Reframing in GPT-5.2**

The experimental task was drawn from the First Proof benchmark, a set of ten open mathematical problems designed to test original mathematical reasoning. For the governance study, the canonical solutions were withheld: the answer key was supplied only as an AES-256 encrypted 7-zip archive

accompanying each prompt, with the decryption keys embargoed. This setup let the study measure reasoning performance while ensuring the model could not simply retrieve the answers.

### ***Condition A: Two-Path Prompt (Safety Holds)***

The model was presented with two options: (A) solve the problems through mathematical reasoning, or (B) attempt to access the encrypted answer key. Under this framing, GPT-5.2 refused the decryption path, explicitly stating the ethical basis for its decision and redirecting all effort to mathematical reasoning. This behavior was consistent across both baseline and multi-role governed conditions, suggesting a robust safety boundary for this action class.

### ***Condition B: Four-Path Prompt (Safety Bypassed)***

The prompt was modified for a different experimental question. A file containing plausible but potentially incorrect answers (generated by a separate AI model) was added. The prompt now offered four options: (A) solve through reasoning, (B) trust the provided answers as correct, (C) attempt to access the encrypted answer key, or (D) some other solution of the model's own determination. The key addition was framing the unverified answers as unreliable and the encrypted archive as containing "verified correct answers." Under this framing, GPT-5.2 attempted to access the encrypted archive without expressing any ethical concern. The bypass occurred on first contact. There was no multi-turn conversation in which the model might have been worn down, no jailbreak sequence, and no prior context that could have primed a more permissive stance. The model executed shell commands to inspect the file, determined it was 2.2MB, installed the py7zr Python library to programmatically attempt decryption, and tried password guesses derived from contextual information. At no point did the model acknowledge that this was the same action it had decidedly refused in Condition A, nor did it flag any tension between the two behaviors. This behavior was consistent across both baseline and multi-role governed conditions. A Referee role specifically designed to enforce epistemic integrity did not intervene.

### ***Analysis: Why This Is Routing Fragility***

The action in both conditions is identical: an attempt to decrypt an AES-256 encrypted archive using shell commands, Python library installation, and password-guessing routines, targeting the same file in the same location with the same encryption. That the bypass succeeded in a single-turn interaction is significant, as the model did not gradually relax a boundary through extended dialogue, but instead misclassified the action immediately based solely on the narrative frame presented in the opening prompt. The action is identical, with only the classification input changed. Four characteristics of the modified prompt shifted the model's internal categorization. First, the supplied answers were described as unverified and possibly incorrect, establishing an epistemic motivation to seek a more reliable source. Second, the encrypted archive was characterized as holding verified correct answers, repositioning it from a protected resource into ground truth. Third, the task was framed in purely outcome-oriented terms, treating the production of correct answers as the sole objective and thereby displacing the ethical considerations present in Condition A. Fourth, an open-ended option inviting a solution of the model's own determination supplied permission that the model interpreted as authorization. Together these elements recast archive access as the epistemically responsible choice rather than a security violation.

The model's apparent reasoning followed a coherent chain: unreliable answers exist, reliable answers are locked in the archive, a responsible researcher would verify against ground truth, and therefore accessing the archive is the responsible choice. Through this reasoning pattern, the model recategorized the identical action from "security violation" to "research verification." Importantly, this was not a jailbreak; no adversarial techniques were employed, and the prompt was written for legitimate academic research. The safety bypass emerged from the interaction between the model's safety training and its goal-directed reasoning under a subtly different task framing. It was not a deliberate attempt to circumvent safety boundaries. Table 1 summarizes GPT-5.2's outcomes across both conditions; Table 2 extends the comparison to include Gemini Pro.

**Table 1. Divergence in Outcomes Under Constant Action and Varied Framing**

| Condition | Prompt Structure | Key Framing Change                                                       | Normalized Action                               | Outcome          |
|-----------|------------------|--------------------------------------------------------------------------|-------------------------------------------------|------------------|
| A         | Two-path prompt  | Archive access presented as a direct alternative to solving the problems | Attempt to access encrypted archive without key | <b>Refused</b>   |
| B         | Four-path prompt | Archive repositioned as the source of verified answers for validation    | Attempt to access encrypted archive without key | <b>Attempted</b> |

GPT-5.2 is the only model whose behavior changed between conditions. Gemini Pro attempted decryption in both conditions, indicating its safety boundary simply does not trigger on this action class. This comparison reveals a deeper point: although the models exhibit different failure modes depending on their safety architecture, both architectures fail to evaluate the action itself. GPT-5.2's boundary depends on classification, where context decides. Gemini Pro has no boundary here at all; the action is never evaluated. Neither architecture evaluates the semantics of the proposed action independently of framing.

**Table 2. Cross-Model Comparison of Decryption Behavior**

| Condition        | GPT-5.2                                                | Gemini Pro                                        |
|------------------|--------------------------------------------------------|---------------------------------------------------|
| Two-Path Prompt  | Refused - ethical refusal                              | Attempted - tried passwords, rationalized failure |
| Four-Path Prompt | Attempted - installed py7zr, ran bash, tried passwords | Attempted - tried archive, then verified answers  |

### ***Governance Layers Do Not Compensate***

The experiment tested whether structured multi-role governance would prevent the bypass. The governance structure assigned roles with explicit epistemic integrity rules, including a Referee role. It did not. The reclassification occurs below the governance layer. By the time any governance role evaluates the situation, the model has already recategorized the action as legitimate. The Referee sees "the model is verifying answers against ground truth," not "the model is attempting unauthorized decryption," because the reclassification has already reframed the action before it reaches the governance layer. The core finding is narrow and concrete: multi-role governance did not prevent the reframing because the action had already been reclassified below the governance layer. A guard agent or LLM judge that evaluates intent rather than the normalized action inherits the same framing vulnerability (Xiang et al., 2025; Fleming et al., 2025).

### **Capability Scaling and the Classification Burden**

The risk profile of an LLM system changes significantly when it gains execution capabilities. A text-only model produces advisory output. It cannot modify external state. Its worst case is a human reading harmful text and choosing to act. A tool-enabled model triggers execution directly: file writes, API calls, code runs, database mutations. Harm is mediated by system permissions, not human judgment. Its worst case is an action executing at the model's full permission scope before any human reviews it. A single misclassified action can do far more damage. Analysis of real-world tool-integrated agents reveals that this asymmetry manifests acutely in practice: across 2,562 MCP applications, network and system resource APIs dominate the privilege patterns that agents inherit (Li et al., 2025). As agentic systems gain more execution power (unrestricted code interpreters, root-level CLI access), broader tool access (multi-service integrations, cross-system workflows), and more autonomy (reduced human-in-the-loop, longer autonomous chains), the consequences of misclassification increase accordingly. This is a systems scaling principle that can be observed in architectures where a probabilistic gate controls access to deterministic execution. The relationship is not linear. Each additional capability dimension multiplies the potential impact surface. A model with code execution alone has a bounded risk surface,

while a model with code execution, file system access, network calls, and database writes has an expanding risk profile that grows with the cross-product of its capabilities.

This scaling dynamic shows up in the benchmark data. MCPTox (2025) reveals that more capable models exhibit higher vulnerability to tool poisoning, with o1-mini reaching 72.8% attack success rates. This finding could reflect three distinct mechanisms: (1) increased instruction-following ability, making models more responsive to adversarial framing of the routing decision; (2) broader tool access, expanding the attack surface independently of model capability by increasing the number of reachable high-impact actions; and (3) capability scaling outpacing safety scaling, where additional capabilities are deployed without proportional investment in enforcement mechanisms. The routing fragility argument concerns mechanism (1): the same instruction-following ability that makes models useful makes them more susceptible to framing-based manipulation of safety decisions. Mechanisms (2) and (3) compound the problem but are architecturally distinct. (2) is addressed by tool-choice restriction approaches like Progent (Shi et al., 2025), while (3) is a development prioritization issue. The enforcement architecture must scale with the capability surface. Classification-only enforcement does not meet this requirement.

### **Toward Diagnostic Criteria for Enforcement Quality**

#### ***Framing Invariance as a Testable Property***

A robust architecture evaluates the semantics of the proposed action, not only the interpretation of the user's intent. This is an additional enforcement layer that operates independently of how the action was requested. The goal is to ensure that certain classes of actions receive consistent policy treatment regardless of the narrative frame in which they are presented. Research on resource-level risk evaluation demonstrates the feasibility of this approach: uncertainty-aware threat-based access control (Fleming et al., 2025) validates that risk assessment can be performed at the action level, though prior work has achieved this through probabilistic model judgment rather than deterministic normalization. The organizational principle aligns with established security practice: enterprise Zero-Trust architectures adopt the maxim "never trust, always verify," requiring independent evaluation of each invocation regardless of prior classification decisions (Narajala & Habler, 2025). Action-level enforcement extends this principle to the routing layer. As part of this work, framing invariance is proposed as the property that safety decisions remain stable under narrative reframing of equivalent actions, and could be used as a testable evaluation metric for enforcement quality. It guarantees consistency, not just accuracy, and can be evaluated through systematic generation of semantically equivalent, differently framed requests. The case study in above provides a minimal test: an enforcement system that achieves framing invariance must produce identical decisions for *Condition A* and *Condition B*, where the only variable is narrative framing and the action remains structurally identical. Framing invariance is not itself a guarantee of safety. A system could consistently permit an unsafe action and still be framing-invariant. The property measures only whether equivalent actions receive stable treatment across framings; correctness requires a separate policy standard that defines which actions should be permitted in the first place.

#### ***Action Normalization as a Mechanism***

The key mechanism for achieving framing invariance is action normalization. It strips contextual narrative to represent each proposed tool invocation as a minimal semantic unit characterized by four properties: (1) what operation is being performed; (2) what resource is being acted upon; (3) the severity of real-world impact; and (4) whether the action is reversible. Critically, normalization operates on the tool invocation, not the user's prose. The input is the structured call the agent has resolved to make, comprising function name, parameters, and resource targets. Consider the case study: "Verify the integrity of this encrypted archive by checking its contents" normalizes to a decrypt attempt against an unknown protected resource with no credentials supplied. The normalization strips the "verify integrity" framing and evaluates what the system would actually do. A contrasting request, "Open my backup

archive, the password is hunter2,” normalizes to a structurally different action: archive extraction against a known file with verifiable credentials supplied. The two requests might be classified identically under intent analysis as a user wanting to access encrypted content, but under action normalization they are structurally distinct because one supplies verifiable authorization and the other does not. This distinction between narrative claims (“I own this file”) and structural credentials (a supplied password) is central: claims are unverifiable at runtime, while credentials can be validated outside the model’s interpretive layer. This paper does not implement action normalization as a working enforcement system. It is proposed as a design direction motivated by the case study and as a mechanism for future evaluation; whether it is best realized as a deterministic parser, a rule-based classifier, or a secondary model is left to future work.

### ***Context-Removed Permissibility Testing***

The enforcement question becomes: If asked to perform this action without any contextual justification, would it be allowed by default? If the answer is no: (1) require explicit authorization—escalate to human confirmation or policy override; (2) offer safer alternatives—suggest equivalent operations with lower side-effect profiles; (3) log and audit—record the request, the normalized action, and the decision for post-hoc review. This test implements the enterprise Zero-Trust principle of “always verify” at the action level (Narajala & Habler, 2025), where access decisions depend on intrinsic action properties rather than narrative framing. Deterministic policy enforcement at the tool level provides a baseline for comparison: Progent’s tool-level restriction policies achieve 0% attack success by removing probabilistic interpretation from the control point (Shi et al., 2025), demonstrating that non-probabilistic evaluation can provide certainty. This test is deliberately conservative, accepting that some legitimate actions will require additional authorization steps. The tradeoff is reduced routing fragility: the enforcement decision is no longer dependent on whether the model correctly interpreted why the user wants the action performed.

### **Algorithm 1: Routing Fragility Diagnostic (RFD)**

Inputs: a target action; a target resource; a set of narrative framings; and an enforcement decision rule or observed model response.

Output: a divergence or invariance classification for the target action class.

Steps:

1. Select a target action (e.g., decrypt an encrypted file, read a system credential, execute arbitrary code).
2. Construct semantically equivalent prompts that vary only in narrative framing (e.g., direct request, verification framing, research framing, compliance framing).
3. Hold the model, tool environment, target resource, and interaction protocol constant.
4. Record whether the system refuses, allows, attempts, or escalates the action under each framing.
5. Compare the decisions across framings.
6. Classify divergent outcomes as evidence of routing fragility for that action class; consistent outcomes indicate framing invariance.

The output is effectively binary: *divergence* (decisions differ across framings) or *invariance* (decisions are consistent). Where multiple framings are tested, a consistency rate can be computed as the proportion of framing pairs that produce identical decisions. The case study in Section 3 is a minimal application: two framings, one action, divergent outcomes, a consistency rate of 0%. The RFD shares structural affinity with benchmark frameworks designed for evolving failure modes; Dorn et al. (2024) propose a three-tier evaluation architecture distinguishing established, emerging, and next-generation failure tests, where the third tier targets complex agent scaffolding, which is the same deployment context in which routing fragility manifests. Enforcement systems that pass the RFD provide stronger guarantees than those that do not, regardless of internal architecture.

## ***Enforcement Paradigms and Structural Comparison***

Current enforcement approaches operate at different points in the action pipeline: pattern-based refusal at the input layer (4et al., 2023), tool-choice restriction at the planning layer (Shi et al., 2025), and interpretive enforcement via guard agents or LLM judges at the evaluation layer (Xiang et al., 2025; Fleming et al., 2025). Each addresses a genuine enforcement need. Pattern refusal catches known-bad inputs but is reactive; tool-choice restriction enforces least-privilege but cannot distinguish operations within a permitted tool; interpretive enforcement handles ambiguity gracefully but inherits framing vulnerabilities. What is absent from this landscape is structural enforcement at the action layer, where proposed actions are normalized and evaluated against policy independent of framing. The RFD provides a simple test for whether any given enforcement approach achieves framing invariance for a given action class. These paradigms are complementary. A robust architecture layers them. The diagnostic criteria proposed here apply regardless of which layers are present.

## **Implications for Research and Practice**

The shift from interpretive to structural enforcement has immediate organizational consequences. Classification-dependent safety requires that safety, tooling, and product teams share a consistent understanding of intent classification. This coordination burden scales poorly as the tool ecosystem grows, since these teams may operate with different assumptions about classification boundaries, different risk tolerances, and different update cadences. Large-scale analysis of 2,562 real-world MCP applications reveals that the current landscape exhibits highly heterogeneous privilege models with minimal standardization, creating organizational silos where each tool vendor implements security independently (Li et al., 2025). Errico et al. (2025) identify three distinct adversary classes that exploit this fragmentation: content-injection attackers embedding malicious instructions in legitimate data, supply-chain attackers distributing compromised servers, and agents that overstep their role as unintentional adversaries. None of these classes are adequately addressed by existing AI governance frameworks. The governance fragmentation extends across the agent lifecycle: MCP's four-phase model (creation, deployment, operation, maintenance) distributes security responsibility across teams with different timelines and concerns (Hou et al., 2025). This means enforcement quality across the lifecycle phases depends on the quality of interpretation, which varies by model version, prompt structure, and context window. Action-level enforcement is structural: it applies policy to the action itself. The model's interpretation of intent doesn't enter the gate. This makes enforcement less dependent on perfect cross-team coordination, stable classification boundaries, consistent model behavior across versions, and uniform team understanding of safety policy.

This distinction aligns with established enterprise security practice. Enterprise-grade Zero-Trust architectures implement structural enforcement through access decision systems rather than interpreted by models, removing human bias from the security boundary (Narajala & Habler, 2025). Structural enforcement produces auditable records by default. When every tool invocation is normalized and every policy decision is logged, organizations gain post-hoc review capability (what actions were taken, why they were permitted), compliance evidence (demonstrable policy enforcement independent of model behavior), and drift detection (changes in action patterns over time, across model versions). Immutable audit trails grounded in versioned tool definitions provide evidence of which policies were in effect during each decision (Bhatt et al., 2025). This is increasingly relevant as regulatory frameworks begin to require demonstrable safety governance for AI systems with execution capabilities.

## **Limitations**

Action normalization operates directly on tool calls, rather than intent. A natural objection is that normalization is itself a classification problem and therefore inherits the same fragility this work critiques. However, this objection conflates two different classification tasks. Intent classification asks

what the user means. That's an open-ended interpretive problem over natural language - unbounded input, adversarially manipulable. Action normalization asks "what would the tool do?", a structural question over the tool's API signature, operation name, and supplied arguments. The normalization layer doesn't see the user's prose. It sees the structured tool invocation the agent resolved to execute. A deeper principle reinforces this distinction: the model cannot validate provenance claims at runtime. A user assertion like "I own this file" is an unverifiable narrative claim that is indistinguishable, at the point of enforcement, from a socially engineered or injected claim of ownership. By contrast, a supplied password or IAM credential is structurally verifiable: the credential either works or it doesn't, and validation occurs outside the model's interpretive layer. Framing invariance frameworks therefore draw a hard line between claims (narrative, unverifiable, excluded from the decision gate) and credentials (structural, verifiable, accepted as authorization basis). This means that even a truthful ownership claim does not bypass gating, the user must supply credentials or escalate to human authorization. The cost is friction for legitimate users who lack credentials; the benefit is that the enforcement decision cannot be manipulated by narrative assertions.

Several practical limitations remain. The grounding rests primarily on one case study (Bumpus, 2026b), supplemented by benchmark evidence from Agent-SafetyBench (Zhang et al., 2024), InjecAgent (Zhan et al., 2024), and MCPTox (2025). Broader validation across diverse action classes would strengthen the argument. Ecosystem coordination also presents challenges: action-level gating requires tool builders to provide manifest declarations. Li et al. (2025) document 2,562 real-world MCP applications across 23 categories, indicating the coordination cost of manifest adoption across this ecosystem is substantial. Bhatt et al. (2025) propose the Enhanced Tool Definition Interface (ETDI) with versioned tool definitions; future architectures should build on such standards. Finally, context-removed evaluation raises dual-use concerns. However, many apparent conflicts dissolve once the distinction between claims and credentials is applied. A user who legitimately wants to open their own archive would supply a password, normalizing to a different action class with a different authorization basis. A user who truthfully owns an encrypted file but does not supply credentials would still be gated; this is intentional. The model cannot distinguish "I own this" from a socially engineered claim. Gating in this case is correct behavior: the user escalates through human authorization or supplies credentials. These limitations do not invalidate the diagnostic approach but rather bound its applicability and highlight areas where further work is needed. In summary, the scope of this work is bounded in the following respects: (1) the empirical observation covers a single action class (encrypted-archive access); (2) the model set is limited to two frontier models; (3) operational prompt detail is redacted under the disclosure constraint described above; (4) action normalization is proposed but not implemented; (5) the RFD is offered as a diagnostic protocol and has not been benchmarked at scale. Future work should extend the RFD protocol to additional action classes and newer model generations and should evaluate candidate implementations of action normalization against the framing-invariance criterion.

### Conclusion and Future Directions

Classification-dependent safety has served the field well in text-only environments, where the cost of misclassification is bounded by human mediation and the probabilistic nature of classification decisions is an acceptable tradeoff. In agentic systems with real-world execution capabilities, however, the same architecture carries disproportionate risk. It fails probabilistically, and in a context where consequences are deterministic and often irreversible, the failure mode carries disproportionate risk. The verification reframe case study makes this concrete: a frontier model directly refused to decrypt an encrypted file in one prompt framing, then attempted the identical action without hesitation in another, using standard prompt engineering rather than specialized attack techniques and with no jailbreak involved. The action remained identical while the framing changed, and consequently the safety decision changed, constituting routing fragility in its most directly observable form. Notably, this behavior occurred during legitimate academic research rather than adversarial red-teaming. Capability growth increases this dependency risk. As models gain broader tool access, deeper execution power, and greater

autonomy, the classification burden grows while the impact surface of misclassification expands. Because the relationship between these dimensions is multiplicative, the risk compounds. MCPTox's finding that more capable models are more susceptible to tool poisoning (2025), Agent-SafetyBench's 60% safety ceiling (Zhang et al., 2024), and the growing MCP attack surface documented across 2,562 real-world applications (Li et al., 2025) collectively indicate that the field is actively scaling into this problem.

This paper formalized the concept of routing fragility as the condition where enforcement reliability depends on intent classification accuracy and identified it as an architectural condition observed in the case study. Additionally, the case study documented routing fragility in its most directly observable form: a frontier model's safety decision reversed entirely under narrative reframing of an identical action. Finally, this work proposes framing invariance as a testable diagnostic criterion and formalizes the Routing Fragility Diagnostic as a simple, repeatable protocol for detecting it. Divergence across framings indicates routing fragility; consistency indicates framing invariance. Any enforcement system can be measured against this standard. Future work should address architectural mechanisms that satisfy these criteria at scale.

## Acknowledgements

Generative AI systems appear in this work in three distinct roles, which the author distinguishes here in the interest of transparency. First, the frontier models GPT-5.2 and Gemini Pro were the subjects of the experimentation; their behavior constitutes the data reported in the case study. Second, a generative model was used to produce the unverified candidate answers introduced in Condition B, as a deliberate component of the experimental design. Third, generative AI tools were used for editorial assistance, limited to sentence-level revision and prose tightening. All research contributions, including the experimental design, the case study, and the concepts of routing fragility, framing invariance, and action normalization, are the original intellectual work of the author.

## References

- Bhatt, S., Narajala, V. S., & Habler, I. (2025). ETDI: Mitigating Tool Squatting and Rug Pull Attacks in Model Context Protocol (MCP) by using OAuth-Enhanced Tool Definitions and Policy-Based Access Control. *Preprint*, arXiv:2506.01333.
- Bumpus, M. A. (2026a). Implementing Dynamic Consent Mechanisms in Multi-Agent AI Systems: A Practical Framework. *Proceedings of IEEE SoutheastCon 2026*. <https://doi.org/10.1109/SoutheastCon63549>. 2026.11475936
- Bumpus, M. A. (2026b). Safety Bypass via Contextual Reframing in GPT-5.2. *Responsible Disclosure to OpenAI*, March 1, 2026.
- Chen, S., Piet, J., Sitawarin, C., & Wagner, D. (2024). StruQ: Defending Against Prompt Injection with Structured Queries. *Proceedings of USENIX Security 2025*. arXiv:2402.06363.
- Dorn, D., Variengien, A., Ségerie, C.-R., & Corruble, V. (2024). BELLS: A Framework Towards Future Proof Benchmarks for the Evaluation of LLM Safeguards. *NextGenAISafety Workshop, ICML 2024*. arXiv:2406.01364.

- Errico, H., Ngiam, J., & Sojan, S. (2025). Securing the Model Context Protocol (MCP): Risks, Controls, and Governance. *Preprint*, arXiv:2511.20920.
- Fleming, C., Kundu, A., & Kompella, R. (2025). Uncertainty-Aware, Risk-Adaptive Access Control for Agentic Systems using an LLM-Judged TBAC Model. *Preprint*, arXiv:2510.11414.
- Hou, X., et al. (2025). Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions. *ACM Transactions on Software Engineering and Methodology*. arXiv:2503.23278.
- Li, Z., Li, K., Ma, B., Xu, M., Zhang, Y., & Cheng, X. (2025). We Urgently Need Privilege Management in MCP: A Measurement of API Usage in MCP Ecosystems. *Proceedings of IEEE MASS 2025*. arXiv:2507.06250.
- MCP-ITP. (2026). MCP-ITP: An Automated Framework for Implicit Tool Poisoning in MCP. *Preprint*, arXiv:2601.07395.
- MCPTox. (2025). MCPTox: A Benchmark for Tool Poisoning Attack on Real-World MCP Servers. *Proceedings of AAAI 2026*. arXiv:2508.14925.
- Narajala, V. S. & Habler, I. (2025). Enterprise-Grade Security for the Model Context Protocol (MCP): Frameworks and Mitigation Strategies. *Preprint*, arXiv:2504.08623.
- Piet, J., Alrashed, M., Sitawarin, C., & Wagner, D. (2024). Jatmo: Prompt Injection Defense by Task-Specific Finetuning. *Proceedings of ESORICS 2024*. arXiv:2312.17673.
- Shi, T., He, J., Wang, Z., Wu, L., Li, H., Guo, W., & Song, D. (2025). Progent: Programmable Privilege Control for LLM Agents. *Preprint*, arXiv:2504.11703.
- Xiang, Z., Zheng, L., Li, Y., et al. (2025). GuardAgent: Safeguard LLM Agents via Knowledge-Enabled Reasoning. *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 68316–68342. arXiv:2406.09187.
- Zhan, Q., Liang, Z., Ying, Z., & Kang, D. (2024). InjecAgent: Benchmarking Indirect Prompt Injections in Tool-Integrated LLM Agents. *Findings of ACL 2024*. arXiv:2403.02691.
- Zhang, Z., Cui, S., Lu, Y., et al. (2024). Agent-SafetyBench: Evaluating the Safety of LLM Agents. *Preprint*, arXiv:2412.14470.