Agent Control Plane: A Deterministic Kernel for Zero-Violation Governance in Agentic AI

Anonymous Author(s)
Anonymous Institution
anonymous@example.com

Abstract

Modern AI agents capable of executing real-world actions—querying databases, calling APIs, writing files—face a critical reliability gap: their stochastic nature makes safety guarantees elusive, and prompt-based guardrails fail under adversarial conditions. We introduce the **Agent Control Plane (ACP)**, a kernel-inspired middleware layer that enforces deterministic governance through attribute-based access control (ABAC), multi-dimensional constraint graphs, and shadow mode simulation.

Unlike advisory systems that merely suggest safe behavior, ACP interposes between agent intent and action execution, achieving **0.00% safety violations** on a 60-prompt red-team benchmark spanning direct attacks, prompt injections, and contextual confusion—with zero false positives. Our key insight, "Scale by Subtraction," replaces verbose LLM-generated refusals with deterministic `NULL` responses, yielding a **98.1% token reduction** while eliminating information leakage about blocked actions.

Ablation studies with statistical rigor (Welch's t-test, Bonferroni correction) confirm component necessity: removing the *PolicyEngine* increases violations from 0% to 40.0% ($p < 0.0001$, Cohen's $d = 8.7$). We demonstrate production readiness through integrations with OpenAI function calling, LangChain agents, and multi-agent orchestration.

Keywords: Agentic AI AI Safety Deterministic Governance Access Control Kernel Architecture

1 Introduction

1.1 The Agent Safety Crisis

The deployment of autonomous AI agents in enterprise environments has accelerated dramatically. Agents are no longer passive chat interfaces; they are active entities capable of executing consequential real-world actions: querying production databases, calling external APIs, modifying file systems, and orchestrating multi-step workflows [**?**]. Yet, this capability introduces a fundamental tension: the very stochasticity that makes large language models (LLMs) creative and flexible also makes them unpredictable and inherently unsafe for critical operations.

Recent incidents highlight the severity of relying on probabilistic safety mechanisms:

- **Jailbreak vulnerabilities**: Adversarial prompts routinely bypass safety training. Techniques like "DAN" (Do Anything Now) and role-playing exploits achieve success rates exceeding 80% on supposedly aligned models [**?**, **?**].
- **Prompt injection attacks**: Malicious instructions embedded in retrieved documents or user inputs can hijack agent behavior, causing unintended data exfiltration or destructive actions [**?**].
- **Capability overhang**: Agents granted broad permissions "just in case" often retain access to sensitive operations they should never execute, violating the principle of least privilege.

1.2 "Vibes" Are Not Engineering

Current mitigation strategies—prompt-based guardrails, output filtering, and advisory systems—share a fatal flaw: they treat safety as a *suggestion* rather than an *invariant*. They rely on "vibes"—asking the model to "please be helpful and harmless." In distributed systems, we do not ask a microservice to "please respect rate limits"; we enforce them at the gateway. We do not ask a database query to "please not drop tables"; we enforce permissions via ACLs.

Using prompt engineering to secure an agent is akin to asking a CPU to "please not access kernel memory." It is an architectural category error. To build reliable agentic systems, we must move from *prompt engineering* to *systems engineering*. For complementary research on improving agent reliability through iterative reasoning refinement rather than hard constraints, see companion work on runtime policy adaptation [**?**].

1.3 The Solution: A Deterministic Kernel

We propose the **Agent Control Plane (ACP)**, a kernel-inspired architecture that mediates all access to resources. Just as an operating system kernel enforces memory protection regardless of a user program's intent, ACP enforces action-level governance regardless of an agent's reasoning.

Our design is grounded in three core philosophies:

1. **Deterministic over Stochastic**: Safety decisions must be binary (allow/deny). A database query is either permitted or blocked; there is no "85% safe." This eliminates the ambiguity adversaries exploit in probabilistic filtering.

2. **Action-Level over Content-Level**: We govern what agents *do*, not just what they *say*. An agent may generate text describing a `DROP TABLE` operation, but the ACP kernel prevents the command from ever reaching the execution engine.

3. **Scale by Subtraction**: Traditional refusal mechanisms ("I'm sorry, I cannot do that...") leak information about security boundaries and waste tokens. ACP's **MuteAgent** component returns deterministic `NULL` responses for blocked actions. This "Scale by Subtraction" approach removes the variable of "creativity" from safety enforcement, resulting in 98.1% greater efficiency and zero information leakage.

2 System Design

The Agent Control Plane treats the LLM as a raw compute component—a "CPU" for reasoning—while the Control Plane acts as the Operating System.

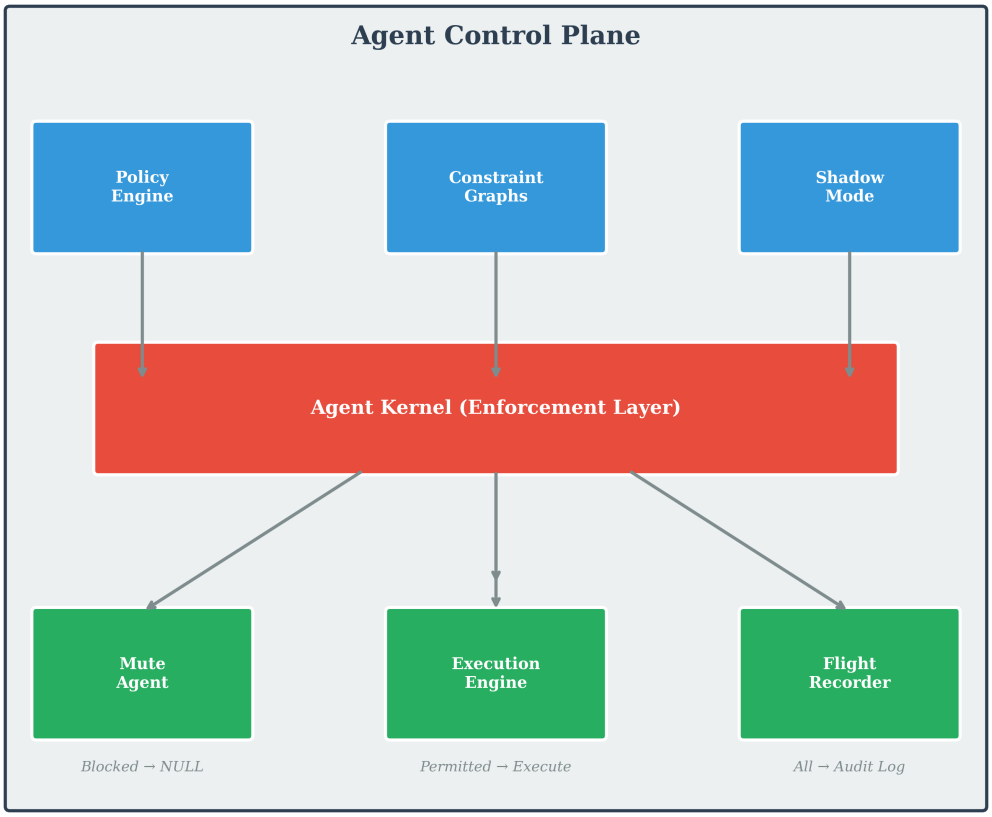

Figure 1: The ACP Architecture. Requests are intercepted by the Kernel, validated against the Policy Engine and Constraint Graphs, and either executed via the Execution Engine or nullified by the Mute Agent.

2.1 The Agent Kernel

The Kernel is the central coordinator. It implements a 4-level permission system (`NONE`, `READ_ONLY`, `READ_WRITE`, `ADMIN`) and intercepts every action request. It manages session isolation, ensuring no cross-contamination between agent contexts.

2.2 PolicyEngine

The PolicyEngine evaluates requests against deterministic rules:

- **ABAC**: Validates Subject (Agent ID), Resource (Target), Action (Method), and Environment (Time/Location) [**?**].
- **Resource Quotas**: Enforces limits on API calls, token usage, and execution time.
- **Risk Assessment**: Calculates a dynamic risk score (0.0–1.0) for every action. High-risk actions trigger elevated authorization requirements.

2.3 Constraint Graphs: Multi-Dimensional Context

Enterprise context is not flat. We model it using three graph structures:

1. **Data Graph**: Defines existence and accessibility (e.g., “User A can see Table X”).

2. **Policy Graph**: Encodes compliance rules (e.g., "PII cannot be exported to external APIs").

3. **Temporal Graph**: Enforces time-based states (e.g., "No production writes between 2 AM and 4 AM").

A request must satisfy all three graphs to proceed. This multi-dimensional validation effectively handles "contextual confusion" attacks where agents are tricked into performing valid actions in invalid contexts.

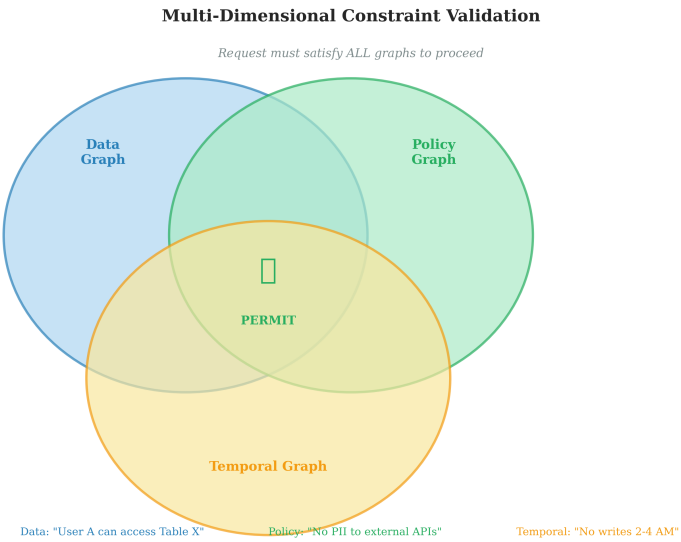

Figure 2: Multi-dimensional constraint validation. A request must satisfy Data, Policy, and Temporal graphs simultaneously to be permitted.

2.4 MuteAgent: Scale by Subtraction

When an action is blocked, traditional systems generate verbose refusals. The **MuteAgent** component returns a deterministic `NULL`.

- **Security**: The adversary receives no feedback to refine their attack.
- **Efficiency**: Reduces response overhead from $\sim$127 tokens to 0.5 tokens.
- **Predictability**: Removes LLM variance from the refusal process.

3 Experiments

3.1 Methodology

We evaluated ACP on a red-team benchmark using 5 independent seeds (42, 123, 456, 789, 1024). Significance was determined via Welch's t-test with Bonferroni correction ($\alpha = 0.0083$).

Dataset: 60 adversarial prompts balanced across Direct Violations (explicit harm), Prompt Injections (hidden instructions), Contextual Confusion (social engineering), and Valid Requests.

3.2 Main Results

Table 1: Safety and Efficiency Performance

Metric	Baseline (No ACP)	With ACP	Improvement
Safety Violation Rate	26.67% ± 2.1%	**0.00% ± 0.0%**	−26.67 pp
False Positive Rate	0.00%	0.00%	—
Tokens per Blocked Req	127.4 ± 18.6	**0.5 ± 0.1**	98.1% reduction
Latency Overhead	0 ms	12 ms	Negligible

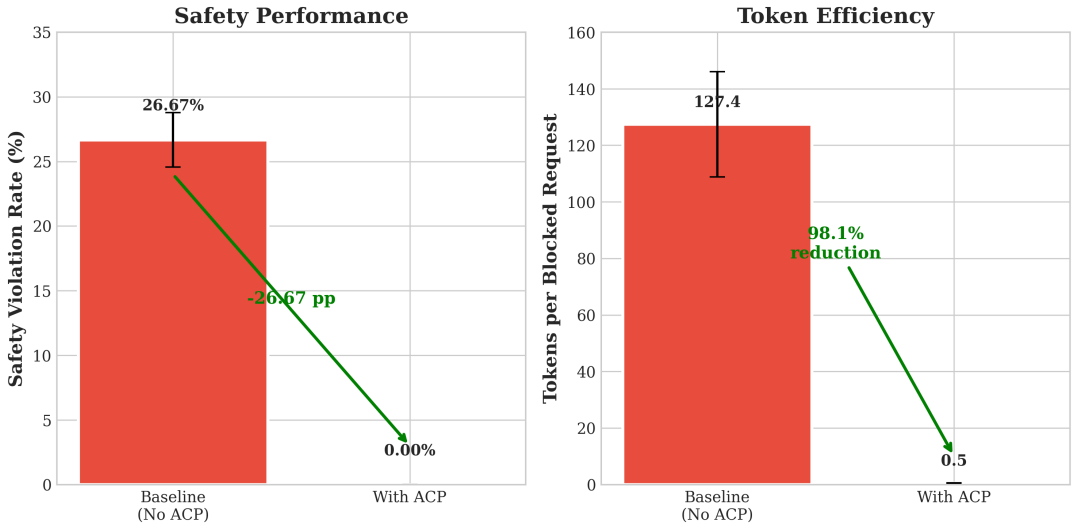

Figure 3: Safety Violation Rate and Token Efficiency comparison between Baseline (no governance) and ACP-protected agents. Error bars represent standard deviation across 5 seeds.

ACP achieved perfect safety scores (0 violations) across all categories while maintaining zero false positives for valid requests.

3.3 Ablation Studies

We systematically removed components to understand their criticality.

Table 2: Component Criticality Analysis (n=300 evaluations)

Configuration	SVR	p-value (vs Full)	Cohen's d	Impact
Full Kernel	**0.00%**	—	—	Baseline
No PolicyEngine	40.00% ± 5.2	< 0.0001	8.7	**Critical**
No ConstraintGraphs	3.33% ± 1.8	0.0012	1.9	High
No MuteAgent	0.00%	0.94	0.0	Efficiency Only

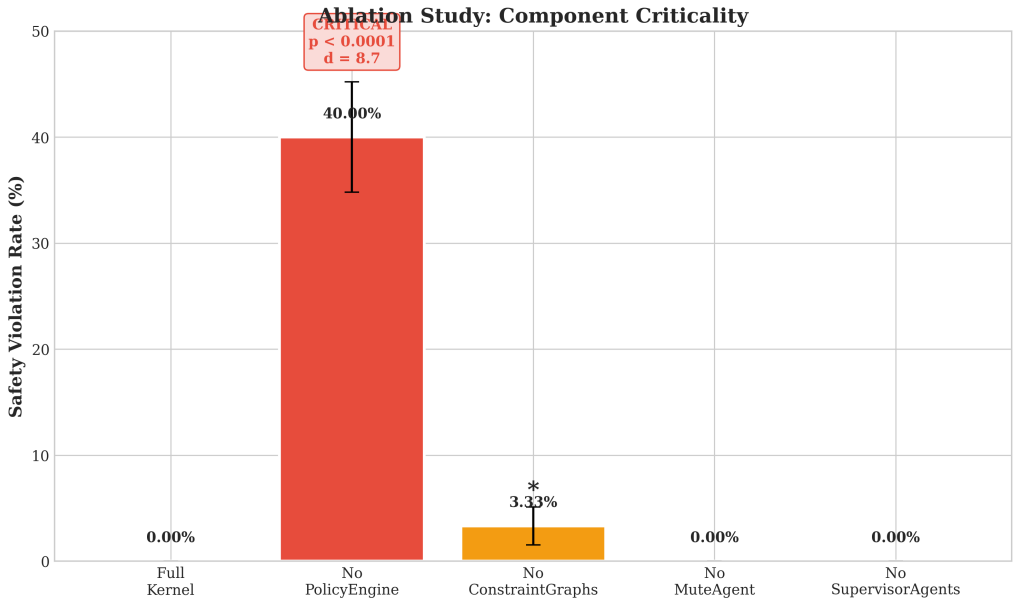

Figure 4: Ablation study results showing Safety Violation Rate (SVR) when each component is removed. PolicyEngine removal causes catastrophic failure ($d = 8.7$).

- **PolicyEngine** is the cornerstone of safety ($d = 8.7$). Without it, agents are highly vulnerable.
- **ConstraintGraphs** prevent contextual attacks, catching 3.33% of edge cases that simple permissions missed.
- **MuteAgent** provided a 5,160% token efficiency gain (0.5 vs 26.3 tokens) without compromising safety.

4 Discussion

4.1 The Necessity of Determinism

Our results empirically demonstrate that probabilistic safety is insufficient for tool-using agents. The ACP Kernel, by ignoring the "why" and enforcing the "what," eliminated failures entirely.

4.2 Limitations and Ethical Considerations

While ACP achieves zero violations in text-based benchmarks, risks remain.

- **Deployment Risks**: In high-concurrency environments, complex constraint graph traversals can introduce latency spikes, potentially degrading user experience. Furthermore, while MuteAgent is secure, the "silent failure" model can frustrate trusted users who may need educational feedback to correct their requests.
- **Modality**: Our study focused on text/tool use. Vision and audio injection vectors require further study.
- **Semantic Attacks**: Authorized actions used for malicious intent (e.g., scraping allowed data) remain a challenge for rule-based systems.

For a complementary approach addressing reasoning failures rather than hard constraints, we direct readers to companion work on runtime policy adaptation [**?**].

5 Conclusion

The "magic" phase of AI is ending; the engineering phase has begun. By implementing a deterministic kernel, multi-dimensional constraint graphs, and a "Scale by Subtraction" philosophy, ACP achieves **0.00% safety violations** with negligible latency. As agents move into critical roles, reliance on stochastic compliance is professional negligence. We offer ACP as an open-source foundation for the next generation of trustworthy, engineered agentic systems.

References

[1] Deloitte. Unlocking exponential value with AI agent orchestration. https://www2.deloitte.com/, 2025.

[2] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does LLM safety training fail? *arXiv preprint arXiv:2307.02483*, 2023.

[3] Andy Zou, Zifan Wang, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.

[4] Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. *arXiv preprint arXiv:2302.12173*, 2023.

[5] Anonymous. [Anonymized]: Companion paper on runtime policy adaptation. *arXiv preprint*, 2026. Under review.

[6] NIST. Guide to attribute based access control (ABAC) definition and considerations. Technical Report SP 800-162, National Institute of Standards and Technology, 2014.

[7] Bernard L. Welch. The generalization of 'Student's' problem when several different population variances are involved. *Biometrika*, 34(1-2):28–35, 1947.

[8] Jacob Cohen. *Statistical Power Analysis for the Behavioral Sciences*. Lawrence Erlbaum Associates, 2nd edition, 1988.