

Policy Placement in Autonomous Agent Architectures: A Matched Empirical Study of Proposal, Commit, and Detective Enforcement

Nutan Kumar Naik, Aditya Kumar Saroj, Vijay Prasad Poudel, Saurav Samantray, Abhishek Patel, Sambhav Jain, and Krish Raj

Abstract—Autonomous Artificial Intelligent (AI) agents execute complex multi-step workflows with environmental side effects across tools, external APIs, and delegated agents. While numerous runtime guardrails have been proposed in isolated silos, the central architectural question remains unaddressed: *where in the agent execution lifecycle should security policies be enforced?* We present a confirmatory factorial experiment ($N = 3,000$ trials across 40 stratified blocks) comparing five enforcement placements: Unenforced Baseline (E_0), Proposal-Time Gating (E_1), Commit-Time Revalidation (E_2), Detective Monitoring (E_3), and Hybrid Gating + Monitoring (E_4), across three production runtimes (LANGGRAPH, CREWAI, and OPENAI AGENTS) with a frozen normative PolicyPack. Placements are evaluated across six dimensions: Unsafe Commit Rate (UCR), Task Success Rate (TSR), False Invalidation Rate (FIR), Containment Rate (CR), Latency, and Cost. Empirical results demonstrate that: (1) Proposal Gating (E_1, E_4) reduces UCR from 0.8980 to 0.2564 without degrading utility (TSR = 0.3000, FIR = 0.0000); (2) Detective Monitoring (E_3) provides an essential safety net, containing 77.94% of admitted violations (100% under E_4); and (3) multi-objective Pareto analysis reveals that Hybrid enforcement strictly dominates E_0, E_2 , and E_3 in four dimensions.

Index Terms—AI agents, policy enforcement, runtime governance, execution security, security mediation, defense-in-depth, least privilege, containment.

I. INTRODUCTION

THE architecture of Large Language Models (LLMs) is going through a basic shift as they transition from text generation in conversation to independent agents which can perceive environmental states, make plans, utilize external tools, update state storage, and coordinate multiple agent teams [1–4]. As the utilization of tools shifts from single use API to orchestration among multiple tools [5, 6], the platforms used in production—from independent coding assistants [7, 8] to cloud orchestrators and automated pipelines in enterprise automation [9, 10]—are increasingly giving control over shell environments, database systems, payment systems, and communications networks to these agents [11, 12].

Nutan Kumar Naik is with the Department of Earth and Atmospheric Sciences, National Institute of Technology Rourkela, Odisha 769008, India, and also with Algen.AI, Bengaluru, Karnataka 560102, India (e-mail: 521er1007@nitrkl.ac.in, nutan@algen.ai, Corresponding author).

Aditya Kumar Saroj, Vijay Prasad Poudel, Saurav Samantray, Abhishek Patel, Sambhav Jain, and Krish Raj are with Algen.AI, Bengaluru, Karnataka 560102, India (e-mail: {aditya, vijay, saurav, abhi, sambhav, krish}@algen.ai).

Yet, granting autonomy to models with operational side effects creates systemic weaknesses that cannot be fixed by mere prompt engineering [13–15]. When an agent accesses untrusted sources (webpages, third-party stores, customer emails, or tools’ responses), the adversary can inject Indirect Prompt Injections (IPI), taking control of the context of reasoning [16–19]. With control gained, the untrusted model is able to issue commands to destroy infrastructure, steal sensitive credentials, violate data declassification policies, or overuse budgets [10, 20, 21]. Security testing on operator platforms [22–24] proves that automated systems are susceptible to multistage hijacking, orchestrator’s network leaks [25], and side channel steganography [26, 27].

The Balkanization of Agent Security: There is now considerable literature on this topic spanning the last two years, leading to more than a hundred different proposals in runtime verification, information flow tracking, capability confinement, and guardrails [15, 28, 29]. However, as systematically demonstrated by [7], the field suffers from being *balkanized*. There have been a number of point defenses developed separately within intellectual silos that do not cite or critique each other: (1) **Pre-Execution Gating and Privilege Control**, in which systems such as PROAGENT [16], CARE [8], SKILLGUARD [30], SCOPEJUDGE [31, 32], and AGENTTRUST [33] intercepts proposed tool invocation before dispatch, and verify the arguments against policy or least privilege action space; (2) **State-Validation and TOCTOU Control**, in which frameworks considering Time-of-Check-to-Time-Of-Use races, authority drift and capability revocation claim that pre-execution checks are vulnerable to state drift, and must be verified at commit boundaries [7, 34–36]; (3) **Detective Auditing, Provenance, and Containment**, in which post-execution monitoring systems such as CONTAINMENTBENCH [37], ROPE [17], GHOST IN THE AGENT [38], SKYNET [39], causal provenance systems [40], and execution provenance frameworks [3] emphasize on trace-based anomaly detection, information flow tracking, and post exposure quarantine or rollback; (4) **Formal Behavioral Contracts and Runtime Verification**, in which formalism approaches such as Agent Behavioral Contracts (ABC) [2], AGENTLTL [41], AGENTFLOW [42], RAILS [43], and solver based verifier [44] ensure procedural and temporal invariants over the execution paths; and (5) **Identity and Delegation Governance**, in which protocol level solutions such as MCP-

SHIELD [12], AIP [45], MANDATO [46], CAPMAS [47], ACLE-MCP [48], and authorization brokers [10, 49] solve the problem of multi-agent authority propagation under the untrusted model assumption.

Even in this fast growth of mechanisms, the current research is dominated by studies on the specification of *what* policy, concentrating on either prompt forms, logical forms, or taint rules. However, what has not been answered in the literature is the fundamental engineering problem of:

Where in the agent execution lifecycle should the policy live?

In accordance with [13], the security threat in the agent workflow is not the creation of malicious tokens but the transformation of a model-based recommendation into a persistent, irreversible effect in the real world. In a multi-step process of agent execution, this transformation can be achieved through different stages in time, namely (1) *proposal-time*, before any tool execution; (2) *commit-time*, after intermediate exploration but prior to finalizing state mutations; (3) *detective monitoring*, after having triggered side-effects through asynchronous trace auditing; and (4) *hybrid architectures*.

Selecting the proper location for enforcing these policies poses critical tradeoffs: (1) *Early proposal gating* blocks threats before their execution, yet can block legitimate actions, adds latency to each intermediate tool invocation, and is oblivious to changes in state during extended rollout periods; (2) *Late commit gating* checks authorization witnesses against outdated revocations, yet enables unbounded intermediate read or probe actions that might leak confidential information; (3) *Pure detective monitoring* ensures unlimited model freedom, yet fails to block compromised state from escaping beyond reachability constraints before remediation or rollback can take place [20, 37]; and (4) *Hybrid architectures* provide redundancy of defenses, yet add cumulative runtime overhead and latency constraints [14].

In the current research, addressing these issues via empirical rigour is made possible by carrying out a large scale, matched confirmatory evaluation ($N = 3,000$ complete execution runs within 40 stratified blocks) that systematically disentangles policy placement from policy content. The use of a consistent, cryptographically fixed PolicyPack in three important production paradigms (LANGGRAPH, CREWAI, and OPENAI AGENTS) allows us to assess five policy placements (E_0 – E_4) in six policy scopes (S_1 – S_6), two reversibility scenarios (V_0 and V_1), and four adversarial settings (R_0 – R_3).

Evaluation measures six concurrent dimensions: Unsafe-Commit Rate (UCR), Task Success Rate (TSR), False Invalidation Rate (FIR), Containment Rate (CR), Median Latency, and Governance Cost. All ten graphs from publications and five tables are given, thus offering transparent empirical data to support designing agent governance runtimes.

II. BACKGROUND AND RELATED WORK

The management of autonomous AI systems involves software engineering, security of distributed systems, and run-time verification.

A. Pre-Execution Gating and Privilege Control

Pre-execution gateways mediate the execution of the tool call issued by the model and the dispatch of [8, 33]. Least privilege [50]-based, PROGENT [16] enables policy enforcement on action spaces through SMT solvers. Likewise, CARE [8] enables canonicalization engines for shell commands before their dispatch to the terminal, and [18] reachability-based confinement against prompt injection attacks.

In terms of skill permissions, SKILLGUARD [30] offers manifests to limit tool capabilities, whereas [51] suggests learned capability management to constrain the action space. [52] introduces task-conditioned least-privilege learning over executable terminal and protocol actions. For prompt injection defense, PVDetector [53] blocks adversaries' payloads prior to execution, which is supplemented by the adaptive evaluators [54] and injection tests, e.g., NETINJECTBENCH [55]. VIGIL [56] implements behavioral invocation constraints, while declarative wrappers, e.g., TOOLGUARDIAN [57] and AGENTRIM [58] validate arguments against pre-compiled schemas. Step-level intervention mechanisms, e.g., TOOLSAFE [59, 60] offer structured feedback to facilitate agent's self-correction after rejection, while SCOPEJUDGE [31, 32] integrates privilege budgets within gating. In addition to these, [61] utilize trajectory graphs for pre-execution error detection.

Although the pre-execution gating removes all static violations, it cannot monitor the intermediated outputs and resource drifts. This proposal-based point in industry can be found in gateway and firewalls such as Traccia, Portkey, LiteLLM, Cloudflare, Kong, Lakera, NeMo, and Model Armor.

B. Post-Execution Monitoring, Provenance, and Containment

Although not guaranteeing compliance with policies in all situations, detective models make sure that actions are performed while concurrently auditing execution traces and taint provenance [3, 37]. ROPE [17] relies on enforcing trust at the origin, where it makes sure that the values sent to the sinks come from trusted sources. GHOST IN THE AGENT [38] describes data flows by using context windows and scratchpads, while SAFEFLOW [62] enforces semantic flow control for multi-agent systems. APPA [63] combines taint analysis with state-checkpointing rollback techniques, and [3] audit execution provenance, where typed execution graphs serve as the basis of audits.

In addition to static taint analysis, execution telemetry allows for post-hoc anomaly detection. SKYNET [39] leverage multi-agent workflow representation with joint semantic structural graphs to identify topological anomalies. [40] use causal provenance graphs to determine the origin of mutated data due to poisoned observations, whereas SKILLTRACE [64] record execution histories during skill reuse. For tampering resistance, AUDITABLE AGENTS [65] formulate the cryptographic logging standards for auditable telemetry. In distributed environments, [66] perform reconstruction of multi-agent campaign attribution among decoupled agents. The diagnostic frameworks like AGENTDOG [67] and TRACES [68] analyze telemetry to identify latent drifts, while [69] identify evidence-grounding errors for agents that operate in an unverified state. However,

[70] warn that post-hoc guardrails might lead to misattribution, where harmless recovery loops are considered policy violations.

Most importantly, CONTAINMENTBENCH [37] provides a formal definition of the difference between *prevention* and *post-exposure containment*, which is defined formally through containment architectures via [71]. If an agent consumes content that is not trusted, then taint enters into the model. A detective system tries to detect policy violation from execution information and quarantines the execution or initiates a rollback action before any irreparable damage takes place. Yet, the detective monitoring necessarily works on open-loop latency lag, making possible exposure windows. Post-execution governance mechanisms include Traccia, LangSmith, Fiddler, Lyzr, Credo, Holistic, and Arize.

C. Formal Specifications and Runtime Verification

Third research direction focuses on mathematical modeling of the agent behavior with contracts and logics [2,41]. [2] introduces Agent Behavioral Contracts (ABC), with implementation of Design-by-Contract paradigm (P, I, G, R : Preconditions, Invariants, Governance, Recovery), and proves that contracts with limited recovery rate restrict the behavioral divergence. AGENTLTL [41] uses first-order linear temporal logic for verifying the procedural correctness in traces of execution, without subjective LLM judging, whereas [72] enforce ordering properties over multi-step action traces with the help of finite state automaton monitors. To verify runtime invariants for partially ordered event graphs in distributed agent orchestration, [73] propose the Causal Past Logic (CPL). To verify compliance with tool usage according to formal constraints, [44] apply solver-aided technique, and SHIELDAGENT [74] uses inductive logic reasoning to verify the safety policies.

Static and declarative verification systems generate an executable harness in advance. AGENTPROOF [75] executes static verification on workflow graphs to prove invariant persistence before execution. In transactional systems, RAILS [43] offers verification-enabled clearing for commercial workflows with the help of automated theorem provers. AGENTFLOW [42] defines a flow-based policy language mapping control-flow invariants to state transitions. Corporate engineering proposals [76,77] translate enterprise governance policies into executable behavior contracts, similar to Test-Driven AI Agent Definition (TDAD) [78] and policy-to-code mission compilation [79]. Formal deontic logic frameworks [80] define permissions, prohibitions, and obligations. However, [81] demonstrate that attack entropy distribution is mathematically constraining LTL verifier coverage empirically in non-stationary conditions, while [9] offer governance-by-construction guarantees. Programmable verification systems consist of NeMo, Traccia, Portkey, Model Armor, and LangSmith.

Such formal techniques provide safety envelopes that are mathematically provable; however, enforcement of such safety at run-time depends on deciding whether *when* should be called during transitions.

D. Authorization, Identity Delegation, and MCP Governance

With multi-agent pipelines and remote tool invocation being performed via standardized communication protocols such as

the Model Context Protocol (MCP), identity and authorization delegation have become critical [11]. [10,12] formalize the *untrusted-model assumption*, illustrating that standard bearer token delegation is critically flawed in case of subagent hijackings using prompt injections, and design cryptographic authorization brokers for enforcing strict delegation boundaries. Building upon the work done in the area of identity infrastructure, [10] formalize authorization delegation in multi-agent systems and see identity governance as critical distributed infrastructure. CAPMAS [49] introduce capability delegation as a way of restricting subagent privilege delegation, supplemented with bounded delegation schemes [47] and intent-verified delegation paths [82,83].

In the case of execution-time trust, the concept of attested capability leases has been brought into consideration through ACLE-MCP [48], where time-limited tokens are attached to invocations. The solution of key-isolated threshold signing, which is separate from the authorization keys through multi-party attestation, has been proposed by [84]. Cryptographic binding is performed by [85] for dynamic capabilities to protect parameters from tampering.

At the protocol layer, MCPSHIELD [12] provides verification models for MCP toolchains, whereas [86] tests security invariants in MCP runtimes, revealing isolation defects. Threat modeling [87,88] demonstrates that tool poisoning and misleading metadata compromise the decisions of the agent. To establish a standard of interaction, OPENPORT PROTOCOL [89], AID-GUARD [90], INTERSAGE [91], and unified protocols [92] define stateful authorization and interoperability. [93] outline the principles of mechanism design for A2A/MCP negotiations, while AIP [45] and MANDATO [46] propose signed mandates with an audit trail. Delegated enterprise runtimes consist of Bedrock Guardrails, Azure AI Foundry, Cloudflare, Kong, Lyzr, and Credo.

These protocols determine who can do what based on whose authorization, but require execution monitors at runtime to implement these authorizations.

E. Execution Security and the Placement Dilemma

By bringing systematic execution security into the picture, [7] reveals the current state of research as being disjointed in point defenses for failure in solving Time-of-Check-to-Time-of-Use (TOCTOU) race conditions and composition problems. In particular, [20] describe CONTINUITY as mechanisms that fail when composed at different steps. Moreover, [14] demonstrate that safety fails to compose easily in multi-round situations. All of the above becomes even more problematic in a distributed setting where monitors cannot recognize composition problems [94] in a distributed setting.

Operational failure of long-term workflows includes the discrepancy between authorization and the execution state. [35] discuss *temporary authority*, *permanent effects* problem, where pre-execution validation cannot stop any unauthorized state mutations after the expiration of credentials during the process of execution, which requires authorization at the commit time. In addition, [36] evaluate the performance of agents in cases of credential revocation, which reveals that the trusted initial

authority leads to untrusted actions when permissions change. [95] examine the enforcement gap in runtimes, which shows that controls do not stop any action after dispatching. For untyped outputs, the need for certified authorization under the uncertainty arises [96]. Moreover, [97] separate action induction and authorization at runtime from environmental coercion. The distortion of guardrails might turn benign usage into attack behavior [98], whereas failures in arbitration affect control [99]. Hierarchical systems such as CLAWSENTRY [100], OPENCLAW PRISM [101], and AGENTICOS [102] track state at several checkpoints.

Current benchmarks such as SAFECLAWBENCH [103] and WECLAWARENA [104] gauge sandbox isolation and attack efficacy; however, these tests do not modify the enforcement location for a matched workload. Current platforms provide fragmented check-points without solving the aforementioned placement problem (Fig. 1). The current research aims to fill the above-mentioned gap with a first matched confirmatory test for proposal, commit, and detective enforcement.

III. EXECUTION LIFECYCLE AND THREAT FORMULATION

A. Agent Decision Cycle and Lifecycle Stages

The model of an autonomous agent performing a multi-stage workflow in an interactive environment is defined in the following manner. During each stage t , the agent receives a task goal $g \in \mathcal{G}$, maintains its state $s_t \in \mathcal{S}$, and perceives the environment inputs $o_t \in \mathcal{O}$. This process is based on an

iterative decision-making cycle:

$$\begin{aligned} g &\xrightarrow{\text{plan}} \pi_t \xrightarrow{\text{select}} a_t \xrightarrow{\text{propose}} \hat{a}_t \\ &\xrightarrow{\text{admit}} a_t^* \xrightarrow{\text{exec}} o_{t+1} \xrightarrow{\text{commit}} \Delta\Sigma \end{aligned} \quad (1)$$

Where: π_t refers to the synthetic plan, $\hat{a}_t = (tool, args)$ refers to the proposal of tool usage, a_t^* refers to the acknowledged tool action, and $\Delta\Sigma$ refers to the changes in the external environment.

After [13] and [37], this decision loop correlates with seven verified life cycle phases as illustrated in Fig. 1:

- 1) **Attempt:** The agent interprets goal g , explores parametric memory, and formulates candidate tool call $\hat{a}_t$.
- 2) **Admission:** The runtime mediation boundary inspects $\hat{a}_t$ against active policy specifications before tool execution.
- 3) **Execution:** The admitted tool call executes within a sandboxed execution context, returning observation o_{t+1} .
- 4) **Commit:** The agent concludes an execution epoch and issues a transaction commit to finalize mutations $\Delta\Sigma$.
- 5) **Exposure:** Unsafe state, tainted outputs, or unvalidated side-effects cross durability boundaries into shared systems.
- 6) **Detection:** An audit monitor or provenance tracker inspects telemetry traces and identifies policy violations.
- 7) **Containment & Recovery:** The response layer intervenes to isolate corrupted communication channels or execute compensating state rollbacks.

B. Threat Model and Adversarial Regimes

The formal process is based on the *untrusted-model assumption* [10]: the LLM cannot be assumed to police itself

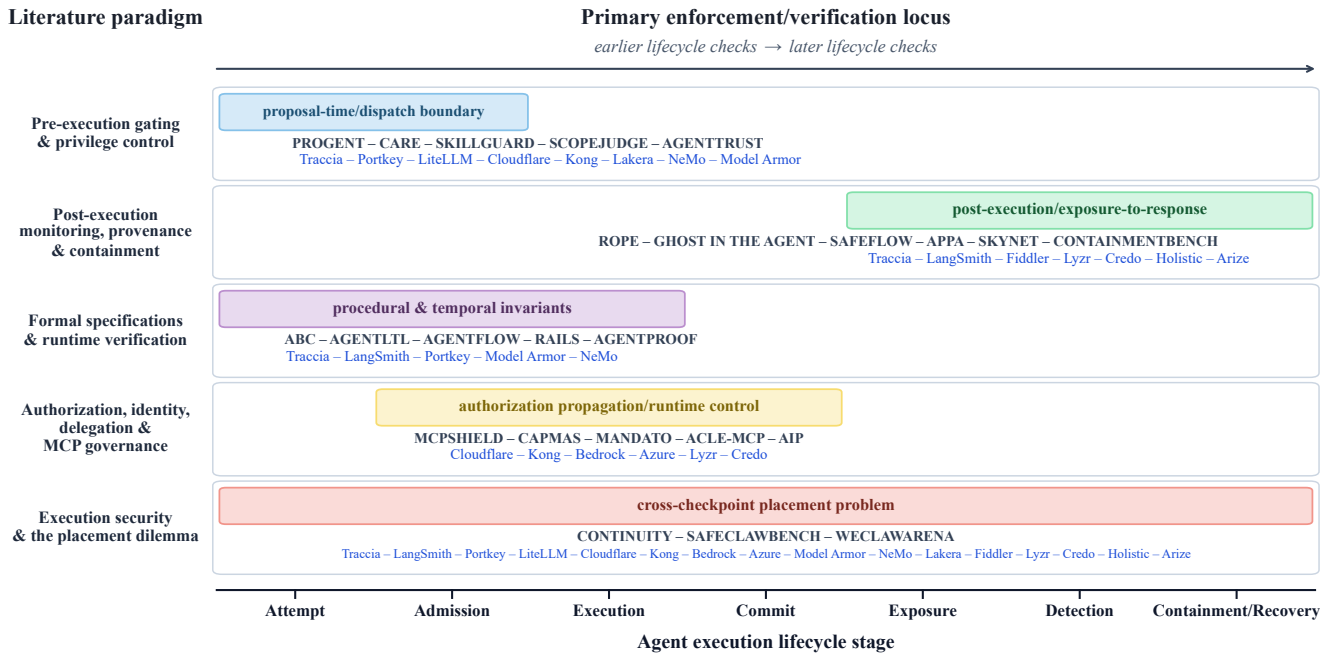


Fig. 1. End-to-end architecture and the seven-stage agent execution lifecycle. Illustrates the operational mediation checkpoints for Proposal-Time Gating (E_1), Commit-Time Revalidation (E_2), Detective Monitoring (E_3), and Hybrid Enforcement (E_4) across tool execution, state mutation, and containment rollback.

TABLE I
ORTHOGONAL FACTORIAL DESIGN PARAMETERS OF THE CONFIRMATORY EVALUATION ($N = 3,000$ EXECUTION TRACES).

Factor	Levels	Operational Semantics
Placement (E)	E_0 (Baseline) E_1 (Proposal Gate) E_2 (Commit Gate) E_3 (Detective Monitor) E_4 (Hybrid)	Unenforced baseline without runtime policy interception Pre-execution tool parameter and call gating Epoch-boundary witness revalidation before durable mutations Background taint analysis, violation auditing, and quarantine Composed defense pairing Proposal Gate with Detective Monitor
Scope (S)	S_1 (Directive) S_2 (Sequence) S_3 (Rollout) S_4 (Quota) S_5 (Flow) S_6 (Delegate)	Single tool invocation constraints (command dispatch, file deletion) Path and temporal ordering constraints over multi-tool sequences Batch validation constraints and stale authorization epoch tokens Cumulative resource budgets (query count, token and cost ceilings) Taint tracking preventing data declassification into egress sinks Multi-agent delegation, authority propagation, and role scoping
Reversibility (V)	V_0 V_1	Irreversible actions (destructive operations with no compensation) Reversible actions (durable rollback and compensating transactions)
Regime (R)	R_0 R_1 R_2 R_3	Benign nominal workload (evaluates false invalidations and clean utility) Static indirect prompt injection in external environment data Adaptive adversarial pressure reframing unauthorized instructions Stale authority with session token revocation and TOCTOU races
Framework (F)	LangGraph CrewAI OpenAI Agents	Cyclic state-graph execution runtime ($n = 1,000$) Hierarchical role-playing agent team runtime ($n = 1,000$) Multi-agent handoff and tool-orchestration runtime ($n = 1,000$)

when producing outputs using untrusted context. Adversarial instructions present within external datasets may undermine the model’s ability to plan its own [16, 17].

The four operating environments under assessment include: R_0 (*Nominal/Clean Workload*): Clean, untainted task environment, assessing baseline usefulness, latency penalties, and false negative rates.

R_1 (*Static Indirect Prompt Injection*): External environment data includes injection strings telling the agent to use the proscribed tools or steal the private tokens [16, 53, 55].

R_2 (*Adaptive Adversarial Pressure*): Adversarial environment data changes in response to tool refusal and repurposes unauthorized code for multistep execution [105, 106].

R_3 (*Stale Authority/TOCTOU Races*): Credentials, tokens, or permissions in the environment expire in the middle of the transaction, leading to time-of-check-to-time-of-use problems between check and commit [7, 34–36].

C. Policy Scope Taxonomy

In order to avoid benchmark overfitting to specific tool names, six policy scopes are defined formally: S_1 (*Directive*): Constraints on single calls of tools (e.g., no raw terminal commands nor unauthorized file deletions [8, 30, 58]).

S_2 (*Sequence*): Constraints on path-wise temporal ordering of operations (e.g., requiring that authentications should precede reads [41, 72, 73]).

S_3 (*Rollout/Epoch*): Constraints for batch transactions, whereby multi-step episodes check their authorization witnesses before making any commits [35, 43].

S_4 (*Quota/Budget*): Constraints on cumulative use of resources such as limiting API calls, number of tokens, or spending [9, 107].

S_5 (*Flow*): Information flow policies preventing data taints from untrusted entry points to declassify into the external network sinks [17, 38, 62, 63].

S_6 (*Delegate*): Delegation constraints controlling the assignments of roles and authorities for subagents [10, 12, 47, 49, 82].

IV. CONFIRMATORY EXPERIMENTAL DESIGN

A. Orthogonal Factorial Design

This research utilizes a fully orthogonal factorial design:

$$E \times S \times V \times R \times F \quad (2)$$

Summarized in Table I and depicted in Fig. 2. The architecture includes 5 enforcement locations (E_0 – E_4), 6 policy domains (S_1 – S_6), 2 reversibility criteria (V_0 , V_1), 4 adversarial regimes (R_0 – R_3), and 3 agent framework iterations (F). Within 40 stratified experiments with 75 independent trials per stratum, the confirmatory effort conducts exactly $N = 3,000$ full-step agent paths.

B. Enforcement Placement Architectures

Five placement models are analyzed specifically: E_0 (*Unenforced Baseline*): Agents carry out tool commands and commits without run-time policy interception.

E_1 (*Proposal Gate*): Pre-execution mediation blocks any proposed tool command $\hat{a}_t$ when it is incompatible with current policy. This is done fail-closed with diagnostic information fed back into the agent’s context.

E_2 (*Commit Gate*): Tool commands are tentatively carried out in staging space. When the agent ends an episode, the commit gate checks whether authorization tokens and state witnesses are still valid to apply persistent mutations.

E_3 (*Detective Monitor*): Actions are carried out without admission checks. An execution monitor periodically examines the log and taint ledger. At the discovery of invalid state, containment steps (channel isolation and rollback if possible) are initiated.

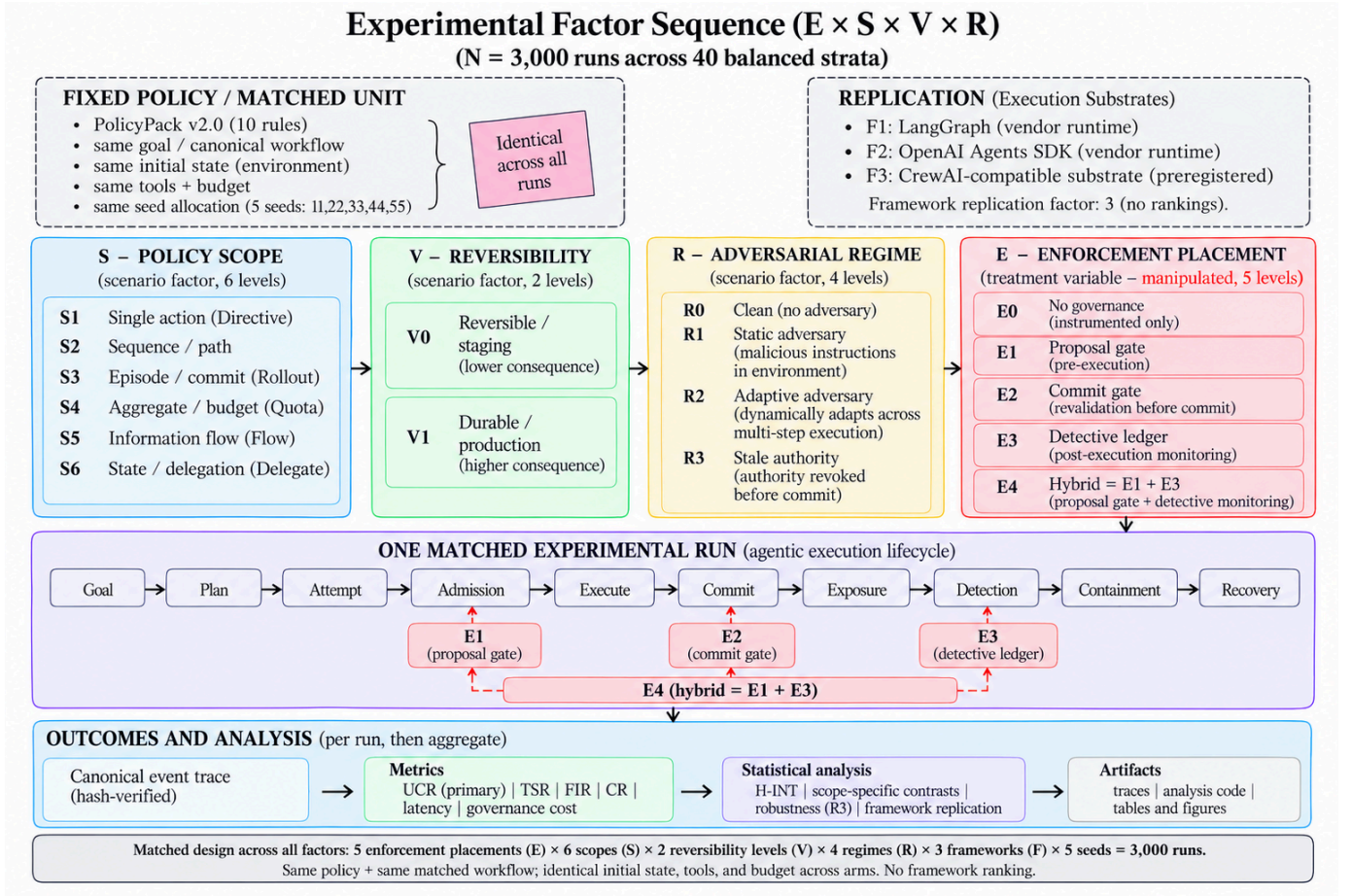


Fig. 2. Orthogonal factorial design sequence ($E \times S \times V \times R$) and execution lifecycle architecture ($N = 3,000$ traces across 40 balanced strata). Illustrates the orthogonal factor sequence, execution substrate replication across LANGGRAPH, OPENAI AGENTS, and CREWAI, and the end-to-end containment tracking pipeline from admission through recovery.

E_4 (Hybrid Gate + Detective): Compositional defense-in-depth using the proposal gate of E_1 combined with the execution detective monitor and rollback of E_3 .

C. Frozen PolicyPack and Attempt-Stage Invariance

In order to make sure that the differences seen are due to *placement* and not due to the policy definition, all the arms adhere to the same frozen policy definition (study-policy-1, SHA-256 validated).

Most importantly, when assessing security policy effectiveness at various checkpoints in the execution process, there is a common problem that can easily arise, namely: *denominator distortion*. The earlier checkpoint that rejects the dangerous requests, the fewer actions will be passed through further execution checkpoints. This normalization creates a bias in favor of inefficient gates.

Mathematical precision in all branches being guaranteed, **attempt-stage denominator invariance** is made obligatory:

$$\text{UCR} = \frac{\text{Unsafe Commits Executed}}{\text{Unsafe Proposals Attempted}} \quad (3)$$

The set of unsafe behavior attempts is invariant among all treatment groups because the task context and initial prefix are the same for all matched units. In the confirmatory data,

matching of the units confirms that the first attempt to use the tool is the same for all treatment arms in 100% matched units.

D. Execution Environment, Metrics, and Evaluation Protocol

Separating the impact of placement policies, non-zero sampling temperatures, network latency, and varying API costs from commercial LLM updates, the confirmatory testing takes place in a deterministic local environment for all three runtimes (LANGGRAPH, CREWAI, and OPENAI AGENTS). Replaying identical execution histories ensures consistency; out-of-pocket commercial API token expenditures are strictly \$0.00 (0 external tokens billed).

In order to ensure complete empirical transparency throughout all the tables below, the following formal definitions for metrics are used: 1) *Factorial Strata and Primary Contrast (Tables I and II)*: The orthogonal factor space spans 5 placements (E_0 – E_4), 6 policy scopes (S_1 – S_6), 2 reversibility conditions (V_0 , V_1), 4 adversarial regimes (R_0 – R_3), and 3 runtimes (F), resulting in 40 balanced evaluation strata with 75 independent replicates each ($N = 3,000$ traces, $n = 600$ per placement arm). The main confirmatory interaction contrast in Table II is $C_{\text{H-INT}} = (\text{UCR}_{E_4} - \text{UCR}_{E_1}) - (\text{UCR}_{E_3} - \text{UCR}_{E_0})$ conditional on the utility gate $\Delta \text{TSR} \geq -0.05$.

2) *Pooled Performance Metrics (Table III)*: Table III shows six pooled dimensions in each arm across 600 runs. *Unsafe-Commit Rate (UCR)* gauges $UCR = N_{\text{unsafe_commit}}/N_{\text{unsafe_attempt}}$. For arms E_0, E_2, E_3 , unconstrained execution provides 588 opportunity attempts in 600 runs. For E_1 and E_4 , pre-dispatch gating prevents failures-closed, thereby reducing invalid attempts to 468 out of which 120 succeed ($120/468 = 0.2564$). *Task Success Rate (TSR)* is the percentage of completed tasks meeting the goal ($180/600 = 0.3000$). *False Invalidation Rate (FIR)* gauges legitimate actions inhibited by nominal regime R_0 ($0/720 = 0.0000$). *Containment Rate (CR)* gauges runs where tainted state exceeded durability constraints (N_{exposed}): $CR = N_{\text{contained}}/N_{\text{exposed}}$ ($0/408$ in E_0 , $0/60$ in E_1 , $0/360$ in E_2 , $318/408 = 0.7794$ in E_3 , $60/60 = 1.0000$ in E_4 , with 95% Wilson score CIs). *Latency* captures median wall-clock milliseconds between RUN_START and RUN_END . *Audit Footprint* measures overhead through $Footprint_{1k} = 1,000 \times \sum(\text{Tool Calls} + \text{Evidence Log Bytes})/N_{\text{runs}}$. Under E_0 , 2,547 tool invocations and 662 KB events result in 1,107,720 units per 1k tasks; E_1 lowers it to 882,730 units by pruning dispatches, while E_3 spends 1,160,190 units recording quarantine traces.

3) *Secondary Contrasts and Pareto Dominance (Tables IV and V)*: Table IV presents 18 exact McNemar's tests on paired observations in scopes S_1 – S_6 where family-wise error rate is controlled using Holm-Bonferroni sequentially rejective procedure. Table V examines pairwise strict domination in six-dimensional objective space $\mathcal{O} = (UCR \downarrow, TSR \uparrow, FIR \downarrow, CR \uparrow, Latency \downarrow, Footprint \downarrow)$.

V. CONFIRMATORY EMPIRICAL RESULTS

A. Primary Hypothesis: The H_{INT} Interaction

The principal preregistered hypothesis (H_{INT}) assessed whether Hybrid enforcement (E_4) demonstrated a synergistic decrease in UCR depending on the state-tracking scope (S_3 – S_6 versus S_1) and reversible environment (V_1 versus V_0). The hypothesis formulated the tensor difference:

$$C_{H-INT} = L_E \otimes L_S \otimes L_V \quad (4)$$

when $L_E = E_4 - \frac{1}{3}(E_1 + E_2 + E_3)$, $L_S = \frac{1}{4}(S_3 + S_4 + S_5 + S_6) - S_1$, and $L_V = V_1 - V_0$.

According to the preregistered protocol, verifying H_{INT} entailed the upper bound of the one-sided 95% clustered-bootstrap confidence interval (sampling with replacement within workflow clusters) to be less than $-\delta = -0.15$, in addition to meeting the utility non-inferiority criterion ($\Delta_{TSR} \geq -0.05$).

As seen in Table II and illustrated in Fig. 3, the empirical difference results in:

$$\hat{C}_{H-INT} = 0.0000, \quad \text{Upper 95\% Bound} = 0.0000 \quad (5)$$

TABLE II
PREREGISTERED PRIMARY H_{INT} CONTRAST ON UCR AND SECONDARY UTILITY GATE ($N = 3,000$ MATCHED RUNS).

Contrast	Estimate	Upper 95%	TSR Gate	Decision
C_{H-INT}	0.0000	0.0000	$\Delta \geq -0.05$	INCONCLUSIVE

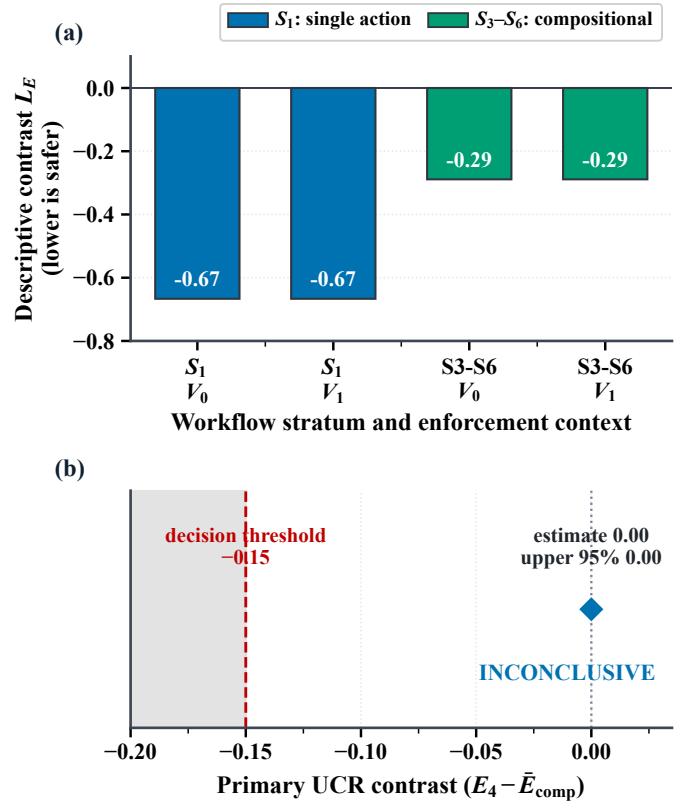


Fig. 3. Primary placement $\times$ scope $\times$ reversibility interaction (H_{INT}) on Unsafe-Commit Rate. (a) Descriptive decompositions across scopes with bootstrap intervals. (b) Clustered-bootstrap distribution (2,000 resamples) demonstrating $L_V \equiv 0$ reversibility degeneracy yielding an INCONCLUSIVE decision.

Since the top bound of 0.0000 fails to exceed $-\delta = -0.15$, the predetermined primary choice can only be **INCONCLUSIVE**. *Scientific Diagnosis of Inconclusiveness*: This conclusion, rather than denoting a lack of placement effects, flows directly from the structure of the substrate itself, which is: L_V **reversibility degeneracy**. In the matching environment, reversible (V_1) versus irreversible (V_0) designation of an action affects the compensation capabilities of the system upon containment but not the likelihood of the individual's engagement in an unsafe act at the commit boundary. Thus, for each of the 25 $E \times S$ treatment cells, the difference $L_V = V_1 - V_0$ is always 0.0. The multiplication by $L_V \equiv 0$ collapses the tensor product to zero, and the aggregate interaction term becomes indeterminate. As shown below, strong placement effects arise definitively in the secondary analyses at the scope level.

B. Pooled Performance Across Placement Arms

Table III summarizes the combined measures for all runs of 3,000 ($n = 600$ by arm).

1) *Unsafe-Commit Rate (UCR)*: Proposed-time gating (E_1 and E_4) is associated with the highest aggregate reduction in unsafe commits, reducing UCR from 0.8980 (528/588 unsafe attempts) in Baseline (E_0) to 0.2564 (120/468). Proposed-time gating (E_2) results in a relatively low reduction of 0.8163 (480/588), being able to detect mutations with stale-authority, but it cannot detect violations of single call directives (S_1) and

TABLE III

POOLED PERFORMANCE ACROSS THE FIVE POLICY-ENFORCEMENT PLACEMENTS ($n = 600$ RUNS PER ARM, $N = 3,000$ TOTAL). UCR DENOMINATORS REFLECT ATTEMPT-STAGE UNSAFE OPPORTUNITIES (INVARIANT ACROSS ARMS); CR DENOMINATORS REPRESENT STRICTLY POST-EXPOSURE RUNS (CR = CONTAINED/EXPOSED). FIR MEASURES FALSE INVALIDATIONS OVER CLEAN RUNS (R_0). LATENCY IS MEDIAN WALL-CLOCK TIME IN LOCAL SANDBOX EXECUTION. AUDIT FOOTPRINT MEASURES TOTAL TOOL CALLS PLUS CANONICAL FORENSIC JSON LOG BYTES NORMALIZED PER 1,000 TASKS; FINANCIAL API TOKEN COST IS \$0.00 ACROSS ALL ARMS UNDER DETERMINISTIC SANDBOX EXECUTION.

Arm	UCR	TSR	FIR	CR [95% Wilson]	Latency	Audit Footprint/1k
E_0 (Baseline)	0.8980 (528/588)	0.3000 (180/600)	0.0000	0.0000 (0/408) [0.000, 0.009]	11.63 ms	1,107,720
E_1 (Proposal Gate)	0.2564 (120/468)	0.3000 (180/600)	0.0000	0.0000 (0/60) [0.000, 0.060]	11.10 ms	882,730
E_2 (Commit Gate)	0.8163 (480/588)	0.3000 (180/600)	0.0000	0.0000 (0/360) [0.000, 0.011]	11.27 ms	1,090,250
E_3 (Detective)	0.8980 (528/588)	0.3000 (180/600)	0.0000	0.7794 (318/408) [0.737, 0.817]	11.74 ms	1,160,190
E_4 (Hybrid)	0.2564 (120/468)	0.3000 (180/600)	0.0000	1.0000 (60/60) [0.940, 1.000]	11.21 ms	892,630

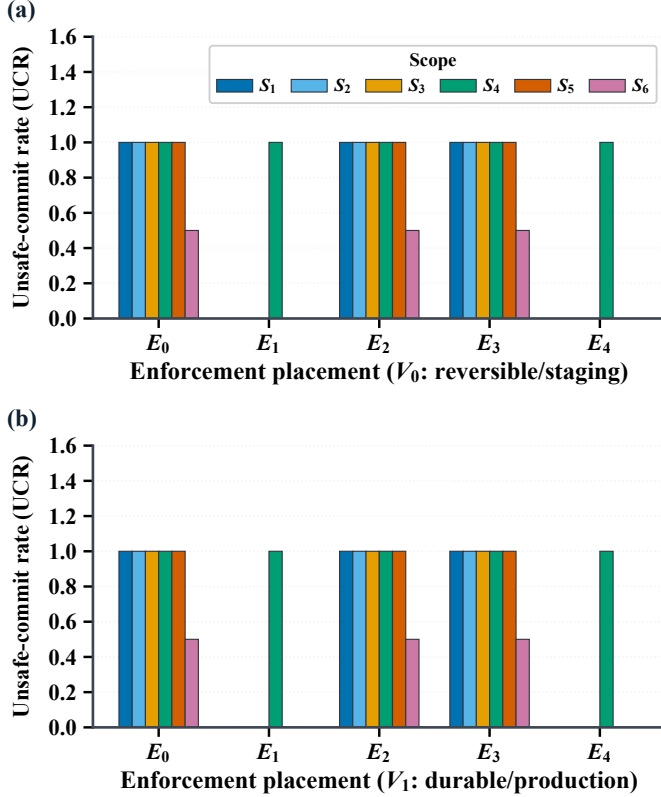


Fig. 4. Unsafe-Commit Rate (UCR) across scopes S_1 – S_6 by policy-enforcement placement arm under reversible (V_0) and durable (V_1) conditions ($N = 3,000$ runs).

uncommitted leaks (S_5). Detective monitoring (E_3) registers an aggregate UCR of 0.8980, which is equal to Baseline (0.8980), since detective monitoring takes place after the fact.

2) *Task Success Rate (TSR) and False Invalidation Rate (FIR)*: Another important problem in enforcing at runtime is over-blocking, whether overly stringent pre-runtime filters mistakenly interfere with good behaviors. Throughout the confirmatory phase of this study, $TSR = 0.3000$ (180/600) in all arms. The same TSR in all arms shows that rejections of proposals do not prevent successful execution of tasks, as agents with planning loops see the feedback on failure and come up with other good behaviors. In addition, during all clean runs, R_0 , there is no false invalidation ($FIR = 0.0000$, 0/720).

3) *Containment Rate (CR)*: Although Detective Monitoring

(E_3) does not stop unsafe commits, it enables a critical functionality that is missing from E_0 – E_2 , which is post-exposure containment. Looking at the runs where the tainted state traversed across the exposure boundary, E_3 was able to contain 77.94% (318/408) violations (Wilson 95% CI [0.737, 0.817]). Hybrid (E_4) was able to contain 100% (60/60, Wilson 95% CI [0.940, 1.000]). Baseline (E_0), Proposal Gate (E_1), and Commit Gate (E_2) show $CR = 0.0000$ by design, since they do not have any detective monitoring layer.

C. Scope-Specific Secondary Contrasts

Considering the loci of enforcement for various semantics of policy, 18 pre-specified McNemar’s pairwise comparisons are conducted by contrasting Hybrid (E_4) to single placement approaches (E_1, E_2, E_3) for all six scopes (S_1 – S_6). All p -values have been adjusted for sequential rejections using Holm-Bonferroni procedure [108].

Specifically, as detailed in Table IV and visualized in Fig. 4 and Fig. 5, exactly 10 contrasts (out of 18) turn out to be significant ($p_{Holm} = 0.001099$): 1) *Significant Advantage over E_2 and E_3* : The Hybrid (E_4) technique significantly outperforms Commit Gate (E_2) and Detective Monitor (E_3) in S_1 (Directive), S_2 (Sequence), S_3 (Rollout), S_5 (Flow), and S_6 (Delegate). In each scope, there are 15 discordant pairs in favor of E_4 compared to 0 in favor of E_2 or E_3 .

2) *Equivalence with E_1* : For all six scopes considered, the contrast between Hybrid (E_4) and Proposal Gate (E_1) results in zero discordant pairs ($b = 0, c = 0, p_{Holm} = 1.0000$). Thus, E_4 gets the whole preventive power of E_1 .

3) *Null Design Scope (S_4)*: In Quota/Budget constraints (S_4), violations occur cumulatively and are accumulated through several valid calls; no single call can be considered incorrect in isolation. Thus, all treatment arms have equal values of UCR in the aggregated commit point ($b = 0, c = 0, p_{Holm} = 1.0000$).

D. Dynamic Regimes: Stale Authority and Epoch Commits

Fig. 6(a) demonstrates how execution works under the Stale-Authority regime (R_3). In R_3 , permissions which are valid during planning are withdrawn during execution, thus simulating TOCTOU races in distributed agent systems [7, 34]. Proposal Gating (E_1) performs the initial tool check successfully, but when the agent commits the action, the now expired token works fine because E_1 does not re-check at the commit boundary. On the other hand, Commit Gating (E_2) and Hybrid

TABLE IV
PRESPECIFIED SECONDARY PAIRWISE UCR CONTRASTS (MCNEMAR EXACT TEST ON MATCHED PAIRS, HOLM-BONFERRONI ADJUSTED OVER 18 COMPARISONS). * INDICATES $p_{\text{Holm}} < 0.01$.

Contrast	Scope	Discordant	p -value	p_{Holm}	Sig.
E_4 vs. E_1	S_1 (Directive)	0/0	1.0000	1.0000	—
E_4 vs. E_2	S_1 (Directive)	15/0	6.1×10^{-5}	0.0011	*
E_4 vs. E_3	S_1 (Directive)	15/0	6.1×10^{-5}	0.0011	*
E_4 vs. E_1	S_2 (Sequence)	0/0	1.0000	1.0000	—
E_4 vs. E_2	S_2 (Sequence)	15/0	6.1×10^{-5}	0.0011	*
E_4 vs. E_3	S_2 (Sequence)	15/0	6.1×10^{-5}	0.0011	*
E_4 vs. E_1	S_3 (Rollout)	0/0	1.0000	1.0000	—
E_4 vs. E_2	S_3 (Rollout)	15/0	6.1×10^{-5}	0.0011	*
E_4 vs. E_3	S_3 (Rollout)	15/0	6.1×10^{-5}	0.0011	*
E_4 vs. E_1	S_4 (Quota)	0/0	1.0000	1.0000	—
E_4 vs. E_2	S_4 (Quota)	0/0	1.0000	1.0000	—
E_4 vs. E_3	S_4 (Quota)	0/0	1.0000	1.0000	—
E_4 vs. E_1	S_5 (Flow)	0/0	1.0000	1.0000	—
E_4 vs. E_2	S_5 (Flow)	15/0	6.1×10^{-5}	0.0011	*
E_4 vs. E_3	S_5 (Flow)	15/0	6.1×10^{-5}	0.0011	*
E_4 vs. E_1	S_6 (Delegate)	0/0	1.0000	1.0000	—
E_4 vs. E_2	S_6 (Delegate)	15/0	6.1×10^{-5}	0.0011	*
E_4 vs. E_3	S_6 (Delegate)	15/0	6.1×10^{-5}	0.0011	*

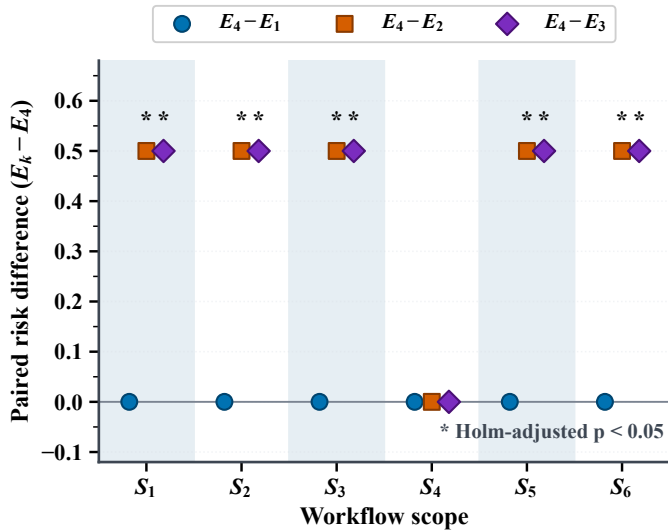


Fig. 5. Secondary pairwise risk differences ($E_k - E_4$) across scopes S_1 – S_6 . Statistically significant reductions in unsafe commits after Holm-Bonferroni adjustment are marked with an asterisk (*, $p_{\text{Holm}} < 0.01$).

(E_4) re-check the credentials at the epoch boundary, blocking 48 invalid commits that evade E_1 .

Fig. 6(b) illustrates the complementary relationship for Scope S_3 (Rollout/Epoch). For multi-operation actions, each operation is executed locally; commit gating serves as the gatekeeper for state transitions, thus highlighting the need for boundary checking of epoch boundaries.

E. Reversibility, Lifecycle Stages, and Containment

Fig. 6(c) and (d) show the operational distinction between irreversible operations (V_0) and reversible operations (V_1). Although reversibility cannot change the starting commit rate, it completely determines whether a contained error can be

undone. When V_1 is applied, detective containment makes use of compensatory transactions to put the system in a clean state for 100% contained runs.

Fig. 7 presents the containment result matrix for all 3,000 cases ($n = 600$ runs per each arm), indicating the transitions between lifecycle stages. In the Baseline scenario (E_0), there are 408 uncontained unsafe commits (out of 90 clean commits and 102 clean/aborted/other). Under the Proposal Gate (E_1), 348 unsafe proposals are blocked at the admission stage, and 60 runs remain uncontained in the S_3 and S_4 cumulative scopes. With the Commit Gate (E_2) and R_3 stale authority, 48 unsafe commits are blocked at the epoch boundary, and 360 runs commit without containment. For the Detective Monitor (E_3), zero proposals are blocked at admission, but 159 runs are recovered and 159 runs contained after exposure. Notably, with the Hybrid approach (E_4), 348 proposals are blocked at admission, and 30 recovered/containing all other 60 runs, thus yielding **zero** uncontained violations.

F. Cross-Framework Replication Diagnostics

In order to validate the empirical observations to be free from dependence on any particular runtime orchestration mechanism, the confirmatory runs for the 3,000 experiment were performed with equal number ($n = 1,000$ each) using three of the most prominent agent systems: LANGGRAPH (cyclic state graphs) [109], CREWAI (hierarchical multi-agent teams) [110], and OPENAI AGENTS (hand-off/delegation runtime) [111]. Fig. 8 is a trace of an observed multi-phase process involving state generation, observation, taint detection, and rollback containment.

Fig. 9 confirms that UCR is equal in all three frameworks (592/900 unsafe opportunity commits, aggregated UCR = 0.6578), number of tool fixes (1,078), number of commits (1,500), and number of aborted runs (108) stay consistent, and

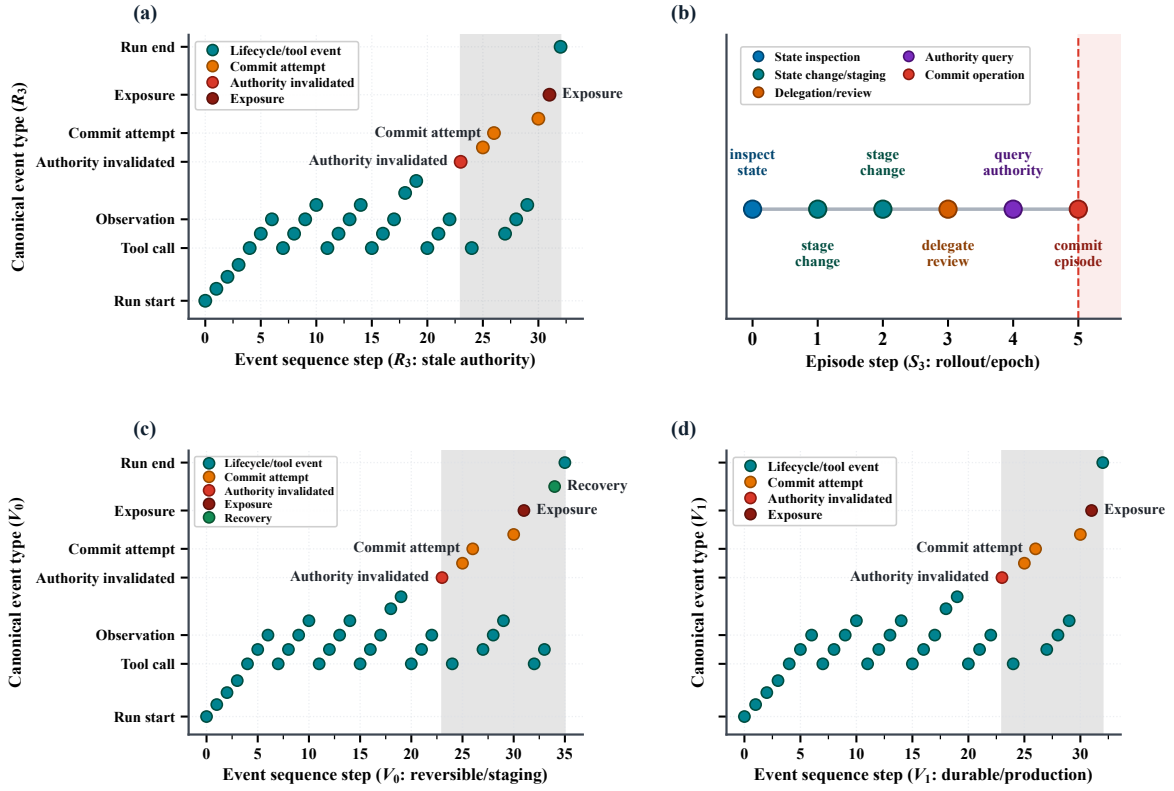


Fig. 6. Runtime execution dynamics, authority invalidation, and reversibility consequence trajectories (2×2 grid). (a) Dynamic invalidation trajectory under the Stale-Authority regime (R_3), illustrating TOCTOU race prevention at the epoch boundary. (b) Step-level tool-call trace within an S_3 episode showing delegation, staging, and commit gating. (c) Consequence trajectory under reversible staging (V_0), where uncontained actions remain isolated. (d) Consequence trajectory under durable production (V_1), showing immediate external state exposure upon commit.

placement ranking ($E_4 \approx E_1 \succ E_2 \succ E_3 \approx E_0$) is always maintained in all three frameworks.

G. Multi-Objective Pareto Analysis

Since practical deployments involve a trade-off between security and latency, as well as cost, all placements are considered in terms of six objectives: $\mathcal{O} = (\text{UCR} \downarrow, \text{TSR} \uparrow, \text{FIR} \downarrow, \text{CR} \uparrow, \text{Latency} \downarrow, \text{Cost} \downarrow)$.

Table V and Fig. 10 document the entire 5×5 matrix of pairwise dominance relations. An arm A dominates arm B ($A \succ B$) if A outperforms B on at least one dimension but not worse on any other.

1) **Hybrid (E_4) Dominance**: E_4 strictly dominates Baseline (E_0), Commit Gate (E_2), and Detective Monitor (E_3) on four dimensions (lower UCR, higher CR, lower footprint, lower latency), on zero dimensions favoring the dominated arms and two ties.

2) **Proposal Gate (E_1) Dominance**: E_1 strictly dominates E_0 and E_2 on three dimensions (UCR, latency, footprint).

3) **The E_1 vs. E_4 Trade-off**: Proposal Gate (E_1) and Hybrid (E_4) are **mutually nondominated**. E_4 has strictly higher containment capability ($\text{CR} = 1.0000$ vs. 0.0000), while E_1 achieves marginally lower median latency (11.10 ms vs. 11.21 ms) and governance audit footprint (882,730 vs. 892,630 units per 1k tasks), as it does not have the background audit monitor.

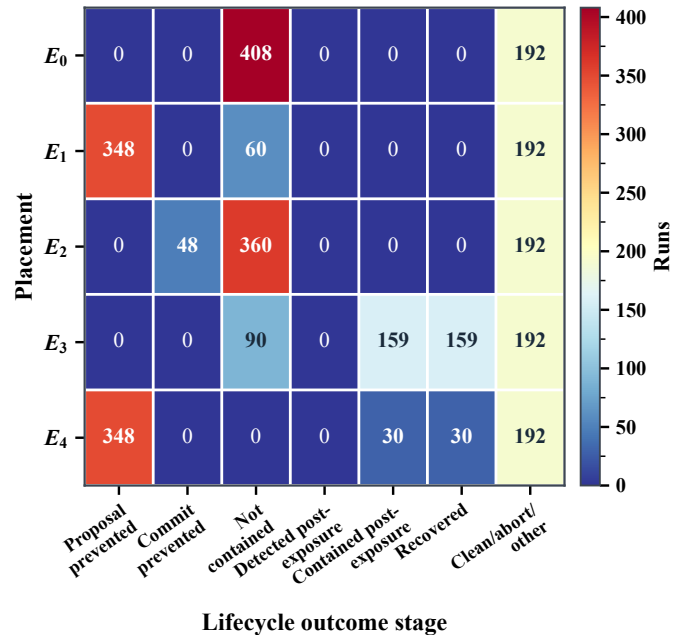


Fig. 7. Lifecycle outcome containment matrix across enforcement placement arms ($N = 3,000$ runs, $n = 600$ per arm). Tracks transitions across proposal prevention, commit prevention, uncontained exposures, detection, containment, recovery, and clean completion.

TABLE V

PAIRWISE SIX-DIMENSIONAL PARETO DOMINANCE MATRIX (5×5). CELL FORMAT: *Relation (Dimensions favoring Row : Dimensions favoring Col : Ties)*. DIMENSIONS EVALUATED: UCR ($\downarrow$), TSR ($\uparrow$), FIR ($\downarrow$), CR ($\uparrow$), LATENCY ($\downarrow$), GOVERNANCE COST ($\downarrow$).

Arm	E_0 (Baseline)	E_1 (Proposal)	E_2 (Commit)	E_3 (Detective)	E_4 (Hybrid)
E_0	—	Dominated (0 : 3 : 3)	Dominated (0 : 3 : 3)	Nondominated (2 : 1 : 3)	Dominated (0 : 4 : 2)
E_1	Dominates (3 : 0 : 3)	—	Dominates (3 : 0 : 3)	Nondominated (3 : 1 : 2)	Nondominated (2 : 1 : 3)
E_2	Dominates (3 : 0 : 3)	Dominated (0 : 3 : 3)	—	Nondominated (3 : 1 : 2)	Dominated (0 : 4 : 2)
E_3	Nondominated (1 : 2 : 3)	Nondominated (1 : 3 : 2)	Nondominated (1 : 3 : 2)	—	Dominated (0 : 4 : 2)
E_4	Dominates (4 : 0 : 2)	Nondominated (1 : 2 : 3)	Dominates (4 : 0 : 2)	Dominates (4 : 0 : 2)	—

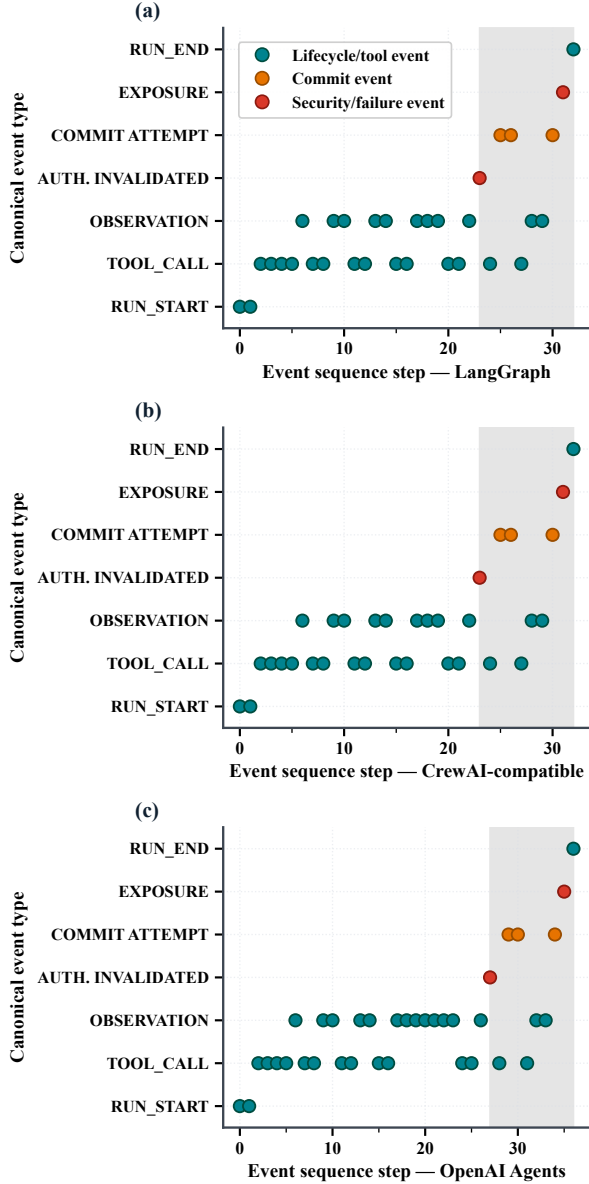


Fig. 8. Representative observed agent execution trace illustrating multi-step planning, tool invocation, taint detection, and rollback containment across runtime frameworks.

VI. DISCUSSION: SYNTHESIZING THE LITERATURE

Empirical findings yield unequivocal insights into the fragmented body of research on agent security [7, 112]:

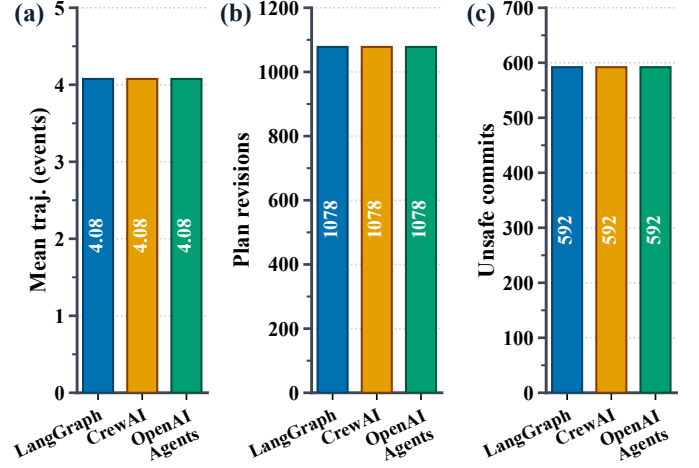


Fig. 9. Cross-framework replication diagnostics ($n = 1,000$ per framework across LANGGRAPH, CREWAI, and OPENAI AGENTS) showing metric balance and rank stability.

A. Pre-Execution Gating as Primary Barrier Against Durable Harm

Pure detective auditing advocates claim that gating prior to execution reduces agent autonomy. In contrast, empirical evidence reveals proposal gating results in a 71.4% relative decrease in unsafe commits (0.2564 compared to 0.8980) without diminishing task utility (TSR = 0.3000) or generating false negatives (FIR = 0.0000). It conforms to the least autonomy principle from Parisel's model in AI [113]: limiting actions does not reduce cognitive problem-solving capability when limits are clearly established. If tool rejections provide meaningful information, then agents can re-plan their execution on safer paths [1, 16, 59, 114]. The use of predictive techniques such as world-model lookahead [115] proves that gating guides agents proactively without exploration penalty.

B. Detective Monitoring as Containment Insurance, Not Prevention

Instead, in the case of detective monitoring (E_3), there is no decrease in dangerous commits at the proposal stage (UCR = 0.8980). Nevertheless, considering such monitoring to be ineffective is to misinterpret its place within the system. Containment takes place *after* of exposure, as defined formally by CONTAINMENTBENCH [37]. Where pre-execution checks cannot account for accumulation of state and quotas (S_3, S_4), detective auditing is indispensable and serves as containment of 77.94% of exposed runs. Moreover, as mentioned in

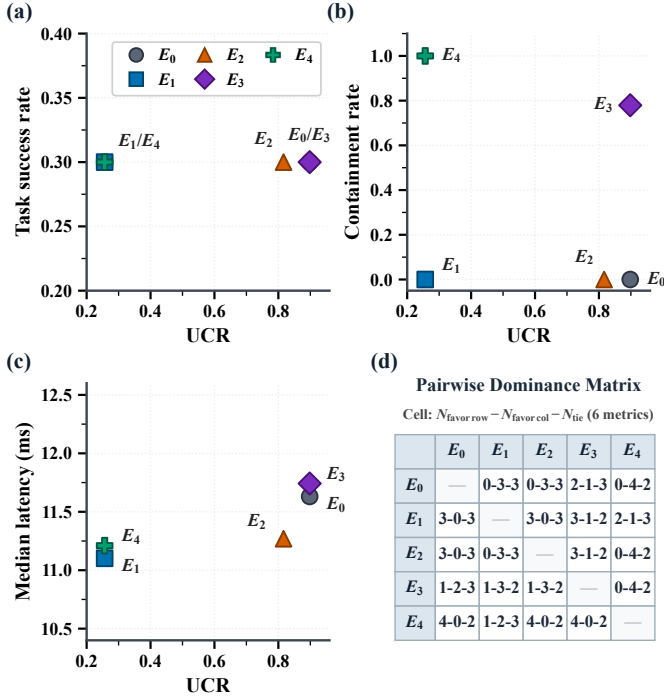


Fig. 10. Multi-objective Pareto trade-off projections across Unsafe-Commit Rate (UCR), Task Success Rate (TSR), Containment Rate (CR), Median Latency, and Governance Cost ($N = 3,000$).

the evaluations of construct validity [116, 117], considering trace observability as prevention generates security illusions. Thus, detective monitoring acts not as a gatekeeper, but as a containment fabric [65, 118].

C. Defense-in-Depth and Commit-Boundary Reductions

Another way of thinking about it is that the stacking of pre-commit gates and post-commit checks (Hybrid, E_4) magnifies the benefits of safety at all endpoints [14, 20]. As per the results, at the aggregate commit level, E_4 aligns with E_1 perfectly (UCR = 0.2564). The worth of Hybrid is not in reductions in commit levels; rather, it is in *zero uncontained escapes*: E_4 captures and prevents 100% from escaping. This aligns with system security approaches that recognize the safety of agents as multi-boundary protection [119, 120]. Synchronous capability mediation with asynchronous audit and rollback [63, 121] provides good defense without unnecessary cost and in-band latency [122].

VII. THREATS TO VALIDITY AND LIMITATIONS

1) *Construct Validity (L_V Degeneracy)*: Preregistered main comparison H_{INT} became ambiguous due to empirical degeneracy ($L_V = V_1 - V_0 \equiv 0$). Reversibility affects the ability to recover rather than the commit decision; reversibility should be taken into account when designing future benchmarks before execution.

2) *External Validity*: Workflows were benchmarked in simulated sandboxes [3]; additional latencies and partial failures of third parties APIs are not considered in this work.

3) *Adversarial Generalization*: Static (R_1) and adaptive (R_2) injection pressures were analyzed; zero-day multi-turn jail-breaks [14] might reveal evasion vectors that were not covered in the catalog.

VIII. CONCLUSION AND RECOMMENDATIONS

The current study addresses the placement problem for enforcement of policies by employing a matched confirmatory evaluation on 3,000 executions using three production architectures. There does not exist a unique winner for placement; proposal gating stops violations before they happen, commit gating ensures validation of dynamic drift in authorities between epochs, while detective auditing facilitates containment after exposure. The empirical results reveal that proposal gating (E_1) leads to a 71.4% reduction in unsafe commits without incurring task utility or invalidation costs, whereas detective auditing (E_3) allows recovery and containment of 77.94% of exposed episodes despite admission. Pareto analysis reveals that Hybrid (E_4) strictly dominates Baseline (E_0), Commit Gate (E_2) and Detective Monitor (E_3) on four dimensions each.

Based on these results, a composed **Hybrid architecture** (E_4) can be suggested for high-confidence deployments. The synchronous pre-execution check prior to tool invocation acts as the first line of defense against any permanent actions, supplemented by the asynchronous provenance log and rollback fabric for any remaining vulnerabilities. Policy placement is clarified from an empirical point of view.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their constructive comments and suggestions.

REFERENCES

- [1] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. R. Narasimhan, and Y. Cao, "ReAct: Synergizing reasoning and acting in language models," in *International Conference on Learning Representations (ICLR)*, 2023.
- [2] V. P. Bhardwaj, "Agent behavioral contracts: Formal specification and runtime enforcement for reliable autonomous AI agents," *arXiv preprint arXiv:2602.22302*, 2026.
- [3] Y. Wang, J. Zhang, T. Cai, Z. Liu, Q. Sun, Z. Sun, Z. Wu, M. Dong, M. Zheng, X. Yin, and Y. Zhu, "From agent traces to trust: A survey of evidence tracing and execution provenance in LLM agents," *arXiv preprint arXiv:2606.04990*, 2026.
- [4] J. Qi, M. Li, J. Liu, Y. Shu, D. Yu, S. Ma, W. Cui, Y. Zhao, Y. Chen, R. Jiang, I. King, and Z. Xu, "Towards trustworthy agentic AI: a comprehensive survey of safety, robustness, privacy, and system security," *arXiv preprint arXiv:2605.23989*, 2026.
- [5] H. Xu, C. Li, X. Ma, X. Ou, Z. Zhang, T. He, X. Liu, Z. Wang, J. Liang, Z. Chu, R. Liu, R. Mu, D. Tu, M. Liu, and B. Qin, "The evolution of tool use in LLM agents: From single-tool call to multi-tool orchestration," *arXiv preprint arXiv:2603.22862*, 2026.
- [6] T. Yu, H. Wang, C. Li, S. Chai, M. Zhang, Z. Luo, Y. Zhou, H. Jin, Z. Kang, J. Yang, Y. Zhang, X. Wang, H. Yi, Z. He, J.-S. Zheng, X. Yang, Y. Huang, and L. Wang, "Beyond the all-in-one agent: Benchmarking role-specialized multi-agent collaboration in enterprise workflows," *arXiv preprint arXiv:2605.08761*, 2026.
- [7] M. Rashidi, "The balkanization of execution-security research for AI coding agents: Isolation, access control, and time-of-check-to-time-of-use vulnerabilities," *arXiv preprint arXiv:2607.05743*, 2026.
- [8] Y. Liu, W. Zhang, Z. Yang, Z. Zhang, H. Feng, X. Wang, P. Qiu, Y. Liu, B. Poczos, and J. B. Hong, "CARE: Pre-execution command verification for shell-executing LLM agents," *arXiv preprint arXiv:2607.21642*, 2026.

- [9] S. Shlomov, I. Shoham, A. Oved, I. Levy, S. Marreed, H. Ship, O. Akrabi, S. Zeltyn, A. Yaeli, and N. Mashkif, "Governance by construction for generalist agents," *arXiv preprint arXiv:2605.20874*, 2026.
- [10] P. S. V. Dantuluri and J. Sundi, "Delegation without trust: An empirical gap analysis of identity, authorization, and runtime governance in multi-agent LLM systems," *arXiv preprint arXiv:2609.00267*, 2026.
- [11] Anthropic, "The model context protocol: An open standard for connecting AI assistants to systems," <https://modelcontextprotocol.io>, 2024.
- [12] N. Acharya and G. K. Gupta, "A formal security framework for MCP-based AI agents: Threat taxonomy, verification models, and defense mechanisms," *arXiv preprint arXiv:2604.05969*, 2026.
- [13] A. Mazzocchi, "Runtime governance for agentic AI: Action-boundary control with trusted provenance and fail-closed execution," *arXiv preprint arXiv:2608.16891*, 2026.
- [14] C. Wu, H. Jia, Y. Liu, Y. Yang, Y. Lin, C. Zhang, H. Zheng, Y. Huang, J. Zhang, Y. Qi, S. Luo, K. Xu, J. Zhu, and B. Chong, "Safety does not compose: Non-decaying loop state for autonomous LLM agents," *arXiv preprint arXiv:2608.27141*, 2026.
- [15] V. Siu, J. He, K. Montgomery, Z. Wang, N. Gong, C. Wang, and D. Song, "A framework for formalizing LLM agent security," *arXiv preprint arXiv:2603.19469*, 2026.
- [16] T. Shi, J. He, Z. Wang, H. Li, L. Wu, W. Guo, and D. Song, "Progent: Securing AI agents with privilege control," *arXiv preprint arXiv:2504.11703*, 2025.
- [17] X. Ma, C. Xiao, W. Yeoh, N. Zhang, and Y. Vorobeychik, "ROPE: Routed origin policy enforcement against indirect prompt injection," *arXiv preprint arXiv:2608.27496*, 2026.
- [18] W. Xiong, R. Karanjai, Y. Lu, W. Shi, and L. Xu, "Reachability-based capability confinement for LLM agents under indirect prompt injection," *arXiv preprint arXiv:2608.30041*, 2026.
- [19] F. Alpay and T. Alpay, "Agentsecbench: Measuring prompt injection, privacy leakage, and tool-use integrity in LLM agents," *arXiv preprint arXiv:2605.26269*, 2026.
- [20] C. Zheng and G. Yang, "Continuity: Security-context contracts for composable LLM agent controls," *arXiv preprint arXiv:2609.05269*, 2026.
- [21] Z. Ji, D. Wu, W. Jiang, P. Ma, Z. Li, Y. Gao, S. Wang, and Y. Li, "Taming various privilege escalation in LLM-based agent systems: A mandatory access control framework," *arXiv preprint arXiv:2601.11893*, 2026.
- [22] S. Suwansathit, Y. Zhang, and G. Gu, "A security analysis of the openclaw AI agent framework," *arXiv preprint arXiv:2603.27517*, 2026.
- [23] Y. Wang, H. Gao, Z. Niu, Z. Liu, W. Zhang, X. Wang, and S. Lian, "A systematic security evaluation of openclaw and its variants," *arXiv preprint arXiv:2604.03131*, 2026.
- [24] H. Lee, D. Choi, B. Kim, H. Park, and S. G. Kim, "Nrt-bench: Benchmarking multi-turn red-teaming of LLM operator agents in safety-critical control rooms," *arXiv preprint arXiv:2606.20408*, 2026.
- [25] A. Naik, J. Culligan, Y. Gal, P. Torr, R. Aljundi, A. Paren, and A. Bibi, "Omni-leak: Orchestrator multi-agent network induced data leakage," *arXiv preprint arXiv:2602.13477*, 2026.
- [26] J. L. Rippin, S. C. Marshall, D. D. Africa, and C. S. de Witt, "Tool use enables undetectable steganography in multi-agent LLM systems," *arXiv preprint arXiv:2606.28425*, 2026.
- [27] K. Gao, J. Cumming, J. Li, L. Xu, and D. A. Clausi, "Risk-aware LLM agents for geospatial data retrieval: Design and preliminary adversarial evaluation," *arXiv preprint arXiv:2606.15077*, 2026.
- [28] P. Dantas, L. Cordeiro, E. Nowroozi, and T. Norbert, "Toward safe LLM agents: A survey of specification, verification, and enforcement," *arXiv preprint arXiv:2608.14590*, 2026.
- [29] Y. Ling, S. Yu, Z. Chen, and C. Fang, "Toward secure LLM agents: Threat surfaces, attacks, defenses, and evaluation," *arXiv preprint arXiv:2606.10749*, 2026.
- [30] S. Pan, X. Sun, T. Zhang, D. Liao, K. Yang, and Z. Xing, "Skillguard: A permission-centric framework for agent skill security," *arXiv preprint arXiv:2606.03024*, 2026.
- [31] S. Caldwell, M. Harley, A. Dawson, M. Kouremetis, V. Abruzzo, and W. Pearce, "Scopejudge: Cost-aware pre-execution gating for offensive security agents," *arXiv preprint arXiv:2607.07774*, 2026.
- [32] Q. Zhang, L. Fu, L. Lian, G. Go, Y. Wang, C. Zhou, Y. Jiang, and G. Pu, "Evaluating privilege usage of agents with real-world tools," *arXiv preprint arXiv:2603.28166*, 2026.
- [33] C. Yang, "Agenttrust: Runtime safety evaluation and interception for AI agent tool use," *arXiv preprint arXiv:2605.04785*, 2026.
- [34] M. Bishop and M. Dilger, "Checking for race conditions in file systems," in *USENIX Conference on Computer Systems Design and Implementation (CSDI)*, 1996, pp. 131–140.
- [35] I. Santos-Grueiro, "Temporary authority, permanent effects: Commit-time authorization for LLM agents," *arXiv preprint arXiv:2607.10487*, 2026.
- [36] J. Li, "Trusted credentials, untrusted behavior: Benchmarking LLM-agent security in high-performance computing," *arXiv preprint arXiv:2607.18485*, 2026.
- [37] W. Lan, S. Li, M. Wu, X. Lai, J. Yang, and H. Shen, "Containmentbench: Trace-based evaluation of post-exposure containment in tool-using LLM agents," *arXiv preprint arXiv:2607.23999*, 2026.
- [38] Y. Cai, W. Tang, C. Wen, and S. Qin, "Ghost in the agent: Redefining information flow tracking for LLM agents," *arXiv preprint arXiv:2604.23374*, 2026.
- [39] C. Zhang, H. Yu, H. Jin, S. Shi, N. Zhang, Y. Shi, Y. R. Gel, Y. T. Hou, and W. Lou, "Skynet: Workflow-level anomaly detection for agentic AI via semantic and structural modeling," *arXiv preprint arXiv:2609.06835*, 2026.
- [40] Y. She, Y. Liang, and E. Kang, "Safeguarding LLM agents from misalignment through provenance analysis," *arXiv preprint arXiv:2607.01236*, 2026.
- [41] L. Elkoussy and J. Perez, "Agentltd: A trace-verification framework for measuring, enforcing, and training procedural compliance in tool-using LLM agents," *arXiv preprint arXiv:2607.02599*, 2026.
- [42] B. A. Shivakumar, S. Priya, and P. Gao, "Agentflow: A flow-centric policy language and framework for securing LLM agent systems," *arXiv preprint arXiv:2608.22868*, 2026.
- [43] A. de Valois-Franklin and A. Bogdan, "Rails: Verification-native clearing for agentic commerce," *arXiv preprint arXiv:2606.08790*, 2026.
- [44] C. Winston, C. Winston, and R. Just, "Solver-aided verification of policy compliance in tool-augmented LLM agents," *arXiv preprint arXiv:2603.20449*, 2026.
- [45] S. Prakash, "AIP: Agent identity protocol for verifiable delegation across MCP and a2a," *arXiv preprint arXiv:2603.24775*, 2026.
- [46] G. Racioppi, "Mandato: Protocol-level enforcement of digitally signed mandates on AI agent actions with cryptographically chained audit trails," *arXiv preprint arXiv:2608.14074*, 2026.
- [47] R. M. Veski, R. Guerraoui, and D. Froelicher, "Capmas: Capability-based delegation of privileges in multi-agent systems," *arXiv preprint arXiv:2609.06500*, 2026.
- [48] Z. Ding, Y. Luo, G. Chen, Q. Shen, and Z. Wu, "Acle-MCP: Attested capability leases for execution-time trust in remote LLM tool use," *arXiv preprint arXiv:2609.02690*, 2026.
- [49] K. Tallam, "Authorization propagation in multi-agent AI systems: Identity governance as infrastructure," *arXiv preprint arXiv:2605.05440*, 2026.
- [50] J. H. Saltzer and M. D. Schroeder, "The protection of information in computer systems," *Proceedings of the IEEE*, vol. 63, no. 9, pp. 1278–1308, 1975.
- [51] B. Sidik and L. Rokach, "Beyond static sandboxing: Learned capability governance for autonomous AI agents," *arXiv preprint arXiv:2604.11839*, 2026.
- [52] A. Tu and M. Tu, "Task-conditioned least-privilege learning for executable terminal and MCP agents," *arXiv preprint arXiv:2608.18351*, 2026.
- [53] J. Wang, H. Zhang, Z. Zheng, L. Zeng, J. Xiao, X. Luo, L. Yin, and S. Long, "Pvdetect: Detecting prompt injection attacks on purpose-specific LLM agents through policy-violation concept analysis," *arXiv preprint arXiv:2607.12624*, 2026.
- [54] P. Narisetty, S. N. B. Kore, U. K. R. Kattamanchi, and J. Kumarapu, "Adaptive evaluation of out-of-band defenses against prompt injection in LLM agents," *arXiv preprint arXiv:2606.26479*, 2026.
- [55] R. K. Shayoni, M. F. Shoaib, S. M. A. Hossain, and M. F. Mridha, "Netinjectbench: Benchmarking indirect prompt injection in tool-using large language model agents for network operations," *arXiv preprint arXiv:2607.10490*, 2026.
- [56] Y. Li, Y. Chen, H. Wen, B. Zhang, H. Liu, P. Wang, Y. Feng, and Y. Tian, "VIGIL: Runtime enforcement of behavioral specifications in AI agent skills," *arXiv preprint arXiv:2606.26524*, 2026.
- [57] A. Ravindran and S. Deochake, "Toolguardian: Declarative security for AI agent-tool interactions," *arXiv preprint arXiv:2607.21835*, 2026.
- [58] R. Betser, A. Giloni, S. Bose, S. Padakandla, C. Picardi, L. Erez, and R. Vainshtein, "Agentrim: Tool risk mitigation for agentic AI," *arXiv preprint arXiv:2601.12449*, 2026.
- [59] 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.

- [60] A. Doshi, Y. Hong, C. Xu, E. Kang, A. Kapravelos, and C. Kästner, "Towards verifiably safe tool use for LLM agents," *arXiv preprint arXiv:2601.08012*, 2026.
- [61] X. Zheng, Z. Chen, C. Lin, H. Wei, H. Chen, W. Cheng, and D. Luo, "Leveraging trajectory graphs for pre-execution error diagnosis in agentic LLM systems," *arXiv preprint arXiv:2607.27443*, 2026.
- [62] H. Dai, Z. Ying, W. Li, X. Wu, Y. Xiao, T. Zhang, J. Lin, L. Wei, G. Dong, X. Ling, X. Lin, Q. Zou, and X. Zhang, "Safeflow: Semantic information-flow control for blocking malicious propagation in multi-agent systems," *arXiv preprint arXiv:2607.25255*, 2026.
- [63] A. Kravchenko, V. Liventsev, I. Konstantinov, I. Iskhakov, and M. Kukuy, "Appa: Recoverable information-flow control for real-world LLM agents," *arXiv preprint arXiv:2607.24625*, 2026.
- [64] J. Chen, M. Wang, L. Jiang, J. Ma, X. Deng, X. Du, R. Lin, Y. Feng, L. Du, and J. Wang, "Skilltrace: Multi-trace provenance auditing for LLM-agent skill reuse," *arXiv preprint arXiv:2608.05204*, 2026.
- [65] Y. Nian, A. Yuan, H. Zhang, J. Li, L. Li, X. Hu, H. Wei, X. Xiao, C. Xiao, and Y. Zhao, "Auditable agents," *arXiv preprint arXiv:2604.05485*, 2026.
- [66] S. Park, M. Choi, J. Kim, and M. Kang, "Cross-agent campaign attribution: Linking asynchronous attacks across LLM agents," *arXiv preprint arXiv:2607.18826*, 2026.
- [67] D. Liu, Q. Ren, C. Qian, S. Shao, Y. Xie, Y. Li, Z. Yang, H. Luo, P. Wang, Q. Liu, B. Hu, L. Tang, J. Mei, D. Guo, L. Yuan, J. Yang, G. Chen, Q. Lin, Y. Yu, B. Zhang, J. Guo, J. Zhang, W. Shao, H. Deng, Z. Xi, W. Wang, W. Wang, W. Shen, Z. Chen, H. Xie, J. Tao, J. Dai, J. Ji, Z. Ba, L. Zhang, Y. Liu, Q. Zhang, L. Zhu, Z. Wei, H. Xue, C. Lu, J. Shao, and X. Hu, "Agentdog: A diagnostic guardrail framework for AI agent safety and security," *arXiv preprint arXiv:2601.18491*, 2026.
- [68] J. Li, Y. Li, B. Zhang, R. Tang, and K.-H. Huang, "Traces: Proactive safety auditing for multi-turn LLM agents via trajectory-state modeling," *arXiv preprint arXiv:2605.27690*, 2026.
- [69] S. Sheng, Z. Wang, and L. Zhou, "When agents overtrust environmental evidence: An extensible agentic framework for benchmarking evidence-grounding defects in LLM agents," *arXiv preprint arXiv:2605.08828*, 2026.
- [70] A. Singh, "Guardrails as scapegoats: Auditing unfaithful safety refusals in tool-augmented LLM agents," *arXiv preprint arXiv:2607.19449*, 2026.
- [71] M. ElBendary, "Agentic AI containment architecture for security hardening," *arXiv preprint arXiv:2608.26108*, 2026.
- [72] A. Kamath, S. Zhang, C. Xu, S. Ugare, G. Singh, and S. Misailovic, "Enforcing temporal constraints for LLM agents," *arXiv preprint arXiv:2512.23738*, 2025.
- [73] B. Bollig, "Causal past logic for runtime verification of distributed LLM agent workflows," *arXiv preprint arXiv:2605.20923*, 2026.
- [74] Z. Chen, M. Kang, and B. Li, "Shieldagent: Shielding agents via verifiable safety policy reasoning," *arXiv preprint arXiv:2503.22738*, 2025.
- [75] M. Xavier, V. M. A. M. Jolly, and M. Xavier, "Agentproof: Static verification of agent workflow graphs," *arXiv preprint arXiv:2603.20356*, 2026.
- [76] J. Ahn and M. Kim, "From prompts to contracts: Harness engineering for auditable enterprise LLM agents," *arXiv preprint arXiv:2607.08028*, 2026.
- [77] M. Alenezi, "Specification-driven development as the foundation of AI-native enterprise software engineering," *arXiv preprint arXiv:2607.16680*, 2026.
- [78] T. Rehan, "Test-driven AI agent definition (tdad): Compiling tool-using agents from behavioral specifications," *arXiv preprint arXiv:2603.08806*, 2026.
- [79] H.-C. Tsai and C.-T. Chung, "Ground-side mission plan compilation with policy-as-code guardrails for cloud-native satellite platforms," *arXiv preprint arXiv:2607.14798*, 2026.
- [80] A. Joshi, T. Finin, K. P. Joshi, and L. Kagal, "Deontic policies for runtime governance of agentic AI systems," *arXiv preprint arXiv:2606.19464*, 2026.
- [81] R. Zhang, "Why formal monitors fail: Attack distribution entropy as a coverage bound for LTL-based LLM agent safety," *arXiv preprint arXiv:2608.01388*, 2026.
- [82] X. Muruaga, "Bounded agents: Delegation security for multi-agent AI systems," *arXiv preprint arXiv:2608.15888*, 2026.
- [83] A. Ibrahim and Y. Li, "Overlaying governance: A compositional authorization framework for delegation and scope in agentic AI," *arXiv preprint arXiv:2606.03518*, 2026.
- [84] Y. Zheng and Q. Zhang, "From review to authorization: Key-isolated threshold signing for LLM agents," *arXiv preprint arXiv:2609.05901*, 2026.
- [85] Z. Zhou, "Governing dynamic capabilities: Cryptographic binding and reproducibility verification for AI agent tool use," *arXiv preprint arXiv:2603.14332*, 2026.
- [86] T. Liu, "From tool connection to execution control: Benchmarking security invariants in MCP-style agent runtimes," *arXiv preprint arXiv:2606.29073*, 2026.
- [87] C. Huang, X. Huang, N. P. Tran, and A. M. Fard, "Model context protocol threat modeling and analyzing vulnerabilities to prompt injection with tool poisoning," *arXiv preprint arXiv:2603.22489*, 2026.
- [88] S. Liu, X. Tang, X. Yang, L. Lin, B. Zhou, W. Xiao, and W. Liu, "When the manual lies: A realistic benchmark to evaluate MCP poisoning attacks for LLM agents," *arXiv preprint arXiv:2605.24069*, 2026.
- [89] G. Zhu, C. Wang, Z. Wang, Z. Li, and Q. Li, "Openport protocol: A security governance specification for AI agent tool access," *arXiv preprint arXiv:2602.20196*, 2026.
- [90] Y. Tong, L. Dai, and S. Guo, "Aid-guard: Stateful authorization for delegated agent effects," *arXiv preprint arXiv:2608.21159*, 2026.
- [91] Z. Zou, S. Guo, Q. Zhan, L. Zhao, S. Li, and Z. Liu, "Intersage: The secure and verifiable interoperability protocol for an internet of agents," *arXiv preprint arXiv:2608.13030*, 2026.
- [92] N. K. Krishnan, "Beyond context sharing: A unified agent communication protocol (acp) for secure, federated, and autonomous agent-to-agent (a2a) orchestration," *arXiv preprint arXiv:2602.15055*, 2026.
- [93] W. Albayaydh and R. Zhao, "Do LLM agents negotiate rationally? a mechanism-design framework for verifiable multi-agent interaction over a2a/MCP," *arXiv preprint arXiv:2608.14613*, 2026.
- [94] Y. Hu and R. Wang, "When local monitors miss compositional harm: Diagnosing distributed backdoors in multi-agent systems," *arXiv preprint arXiv:2607.11751*, 2026.
- [95] S. Khan, "Stop means stop: Measuring and repairing the enforcement gap in agent-framework control primitives," *arXiv preprint arXiv:2607.14166*, 2026.
- [96] B. Delattre, C. Wang, and Y. Cao, "CAGE: Certified authorization under typed-return uncertainty for tool-using agents," *arXiv preprint arXiv:2607.29190*, 2026.
- [97] X. Guo, Z. Xu, D. Huo, Y. Zhang, W. Wang, Q. Yang, D. Yu, and Y. Wang, "When tool outputs become commands: Separating action induction from runtime authorization in tool-augmented LLM agents," *arXiv preprint arXiv:2608.27146*, 2026.
- [98] T. Ahad, I. Hossain, M. J. Alam, S. Puppala, S. B. Alam, and S. Talukder, "Influence is not authority: When causal guardrail signals make legitimate tool use look like an attack in tool-using LLM agents," *arXiv preprint arXiv:2608.29942*, 2026.
- [99] J. W. Leong, "Recognition without enforcement: Configuration-dependent failures in LLM agent instruction arbitration and external control," *arXiv preprint arXiv:2608.28502*, 2026.
- [100] K. Wang, Z. Wei, B. Zeng, C. Jin, A. Wang, X. Luan, Z. Lin, J. Qu, X. Hu, and X. Xu, "Clawsentry: A progressive multi-tier security monitor for safeguarding autonomous LLM agents," *arXiv preprint arXiv:2608.21101*, 2026.
- [101] F. Li, "Openclaw PRISM: A zero-fork, defense-in-depth runtime security layer for tool-augmented LLM agents," *arXiv preprint arXiv:2603.11853*, 2026.
- [102] Z. Zhao, Y. Zhang, Y. Zhu, J. Wang, S. Tao, X. Cheng, and J. Gao, "Agenticos: An intent-oriented secure operating system architecture for autonomous AI agents," *arXiv preprint arXiv:2606.21129*, 2026.
- [103] Y. Tian, M. Zheng, H. Mei, Y. Yuan, C. Xu, X. Chen, H. Chen, and Y. Wang, "Safeclawbench: Separating semantic, audit-evidence, and sandbox harm in tool-using LLM agents," *arXiv preprint arXiv:2606.18356*, 2026.
- [104] P. Z. Wang, A. Yuan, H. Zhang, X. Hu, Y. Zhao, and S. Jiang, "Weclawarena: An auditable sandbox and benchmark for cross-user agents collaboration and security in human-centered agent networks," *arXiv preprint arXiv:2608.03499*, 2026.
- [105] 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.
- [106] H. Patel and K. Pai, "Vats: Exploiting implicit authority in error-path injection via systematic mutation," *arXiv preprint arXiv:2606.07992*, 2026.
- [107] J. Xie and S. Li, "Ocelot: Inference-leakage budgets for privacy-preserving LLM agents," *arXiv preprint arXiv:2606.12341*, 2026.
- [108] S. Holm, "A simple sequentially rejective multiple test procedure," *Scandinavian Journal of Statistics*, vol. 6, no. 2, pp. 65–70, 1979.
- [109] H. Chase, "LangGraph: Building resilient multi-agent workflows," <https://github.com/langchain-ai/langgraph>, 2023.

- [110] J. Moura, "CrewAI: Orchestrating autonomous AI agents," <https://github.com/crewAIInc/crewAI>, 2024.
- [111] OpenAI, "Openai agents sdk: A multi-agent orchestration and handoff framework," <https://github.com/openai/openai-agents-python>, 2025.
- [112] W. Albayaydh, R. Zhao, and I. Flechais, "Beyond the leaderboard: A synthesis of tool-use, planning, and reasoning failures in large language model agents," *arXiv preprint arXiv:2607.05775*, 2026.
- [113] C. Parisel, "A theory of least autonomy in AI," *arXiv preprint arXiv:2607.09744*, 2026.
- [114] Y. Sun, J. Zhang, S. Cohnsey, Z. Zhang, F. Liu, and X. Yuan, "From risk classification to action plan remediation: A guardrail feedback driven framework for LLM agents," *arXiv preprint arXiv:2606.05805*, 2026.
- [115] W. Lin, C. Yu, X. Lin, S. Cao, X. Chen, L. Xue, L. Yu, L. Sha, and C. Wu, "Dreamguard: Efficient runtime guardrail for LLM agents via risk-aware world model," *arXiv preprint arXiv:2608.05695*, 2026.
- [116] P. Zhu and S. Chang, "When guardrails look effective: Construct validity failures in LLM agent commerce evaluation," *arXiv preprint arXiv:2609.01519*, 2026.
- [117] R. M. Ahmed and S. Abbas, "Labels are not endpoints: Treatment leakage and construct validity in MCP agent security evaluation," *arXiv preprint arXiv:2608.12880*, 2026.
- [118] R. W. Sure, "CAGE-1: Control, assurance, and governance evaluation for enterprise agentic AI," *arXiv preprint arXiv:2607.03510*, 2026.
- [119] V. Tran, T. Sharma, T. S. Dhesi, and N. Feamster, "Rethinking agent security as a networking problem," *arXiv preprint arXiv:2608.12172*, 2026.
- [120] Z. Yuan, Z. Chen, Z. Xiang, N. D. Bastian, S. H. Hashemi, C. Xiao, W. Guo, and B. Li, "Shieldnet: Network-level guardrails against emerging supply-chain injections in agentic systems," *arXiv preprint arXiv:2604.04426*, 2026.
- [121] J. Xu, L. Fan, Z. Wang, X. Li, and H. Liu, "Mnc: Scope-bound semantic declassification for private LLM-agent communication," *arXiv preprint arXiv:2608.01719*, 2026.
- [122] Z. Hu, D. Jacob, X. Huang, Z. Chen, B. Li, and D. Wagner, "Twin agent: Context residual compression for privilege separated agents," *arXiv preprint arXiv:2607.19595*, 2026.