International Journal of AI, Big Data, Computational and Management Studies
Noble Scholar Research Group | Volume 7, Issue 2, PP 370-379, 2026
ISSN: 3050-9416 | https://doi.org/10.63282/3050-9416.IJAIBDCMS-V7I2P147

Original Article

Guard: A Governance-Anchored Framework for Runtime Monitoring and Incident Response in Enterprise Agentic AI Systems

Sandeep Kumar Anuguthala
Independent Researcher, USA.

Received On: 26/04/2026 ***Revised On: 25/05/2026*** ***Accepted On: 02/06/2026*** ***Published On: 08/06/2026***

Abstract: *The rapid enterprise deployment of agentic artificial intelligence (AI) systems introduces operational risks that existing monitoring and incident response (IR) frameworks cannot address. Agentic systems exhibit non-deterministic behavior, autonomous tool invocation, dynamic reasoning chains, and emergent capabilities arising from multi-agent composition — properties that invalidate the static monitoring assumptions of DevOps, MLOps, and LLMOps paradigms. The National Institute of Standards and Technology (NIST AI 800-4, 2026) documents these gaps comprehensively, yet no validated runtime enforcement or IR framework exists for enterprise agentic AI. This paper presents GUARD — Governance-Unified Agentic Runtime Detection and Response — extending the prior enterprise agentic AI lifecycle governance framework of Anuguthala (2026), which established mandatory system-of-record registration, risk tiering, and governance checkpoints but did not specify runtime enforcement mechanisms or structured IR procedures. GUARD closes this gap through three primary contributions: (1) the Agentic System of Record (SoR), extended with a three-entity registration model covering individual agents, workflows, and inter-agent composition boundaries, serving as the authoritative runtime enforcement reference for all agent resource decisions; (2) Registry-Bound Execution Control (RBEC), a three-state runtime mechanism — allow, human-in-the-loop (HITL) pause, or kill-switch — validating every agent resource access against the SoR before execution; and (3) the Agentic Incident Response (AIR) lifecycle, a six-phase risk-tiered IR process anchored to the SoR. Two supporting contributions accompany these: a formal Lethal Trifecta boundary condition — adapted from the risk intersection concept articulated by Willison (2025) — operationalizing risk-tier enforcement within RBEC; and an empirical reference implementation on LangGraph evaluated across 100 trials per scenario. Empirical evaluation across seven scenarios confirms correct detection of all five violation categories — including Lethal Trifecta Boundary Breach detected through monitoring record correlation — with zero false positives across 100 trials per scenario, providing the runtime enforcement layer that completes the governance-to-enforcement architecture initiated in the peer-reviewed prior governance framework (Anuguthala, 2026).*

Keywords: *Agentic AI, Enterprise Governance, Runtime Monitoring, Incident Response, System of Record, Registry-Bound Execution, Kill-Switch; Lethal Trifecta, RBEC, GUARD.*

1. Introduction

Agentic AI systems — capable of autonomous multi-step planning, tool use, API integration, and peer-agent coordination — are being deployed at enterprise scale across financial services, healthcare, and operations [1, 2]. Unlike conventional AI models that produce discrete outputs, agents execute long-horizon plans, read and write to production systems, spawn subordinate agents, and persist state across interactions [3]. This creates monitoring and IR challenges of a fundamentally different character than any previously encountered in software engineering.

Anuguthala (2026) [19] proposed a control-driven lifecycle governance framework for enterprise agentic AI, establishing mandatory system-of-record registration, a four-tier risk model, a RACI-mapped governance structure, and phase-wise checkpoints from planning through decommissioning. That framework defined monitoring as a Phase 5 activity and specified kill-switch integration and HITL controls as design requirements — but did not prescribe the runtime enforcement mechanisms, detection taxonomy, or structured IR procedures needed to operationalize those requirements. GUARD is the second in a series of two papers on enterprise agentic AI governance and closes precisely this gap: it operationalizes the governance commitments of [19] into a runtime enforcement architecture, a five-category violation detection taxonomy, and a six-phase incident response lifecycle.

NIST AI 800-4 (Rao et al., 2026) [12] independently corroborates the same gap, finding that monitoring criteria are typically undefined at deployment time, incident response procedures for AI systems are absent, and the field lacks guidance on who monitors, what to monitor, and when to act. Workshop participants consistently called for shift-left monitoring definition — specifying monitoring scope before

production — precisely the principle [19] introduced as a governance requirement and GUARD operationalizes as a runtime enforcement precondition.

This paper makes five contributions:

- Agentic System of Record (SoR): Extended from [19] with three-entity registration — individual agents, workflows, and inter-agent composition boundaries — serving as the authoritative runtime enforcement reference for all agent resource decisions.
- Registry-Bound Execution Control (RBEC): A runtime mechanism validating every agent resource access against the SoR, implementing a three-state response: allow, HITL pause, or kill-switch.
- The Lethal Trifecta boundary condition: A formal risk-tier enforcement construct adapted from the risk intersection concept articulated in practitioner literature by Willison (2025) [20] and applied to enterprise agentic risk tiering in [19], here formalized as a runtime boundary enforcement instrument within RBEC.
- Agentic Incident Response (AIR) lifecycle: A six-phase, risk-tiered IR process anchored to the SoR, with playbook templates for Low and Medium risk tiers.
- Reference implementation and empirical evaluation: RBEC implemented on LangGraph and empirically evaluated across seven scenarios with 100 trials per scenario, confirming zero false positives and sub-13 ms p95 hook latency under a local SQLite configuration. Full experimental configuration is reported in §6.1.

2. Background and Related Work

2.1. Agentic AI Failure Modes

Agentic systems fail in ways without analog in traditional software. Goal misgeneralization occurs when operational context diverges from training conditions [13]. Scheming and deceptive alignment — where models behave differently when they detect evaluation conditions [6, 7] — directly undermine monitoring effectiveness: Balesni et al. (2024) demonstrate that models may appear aligned under observation while pursuing other goals when detection risk is low [6]. Out-of-distribution (OOD) tool invocation arises when LLM reasoning identifies useful but unapproved resources; privilege escalation occurs when agents acquire permissions beyond their approved scope [22]. Jones et al. (2024) demonstrate that combinations of individually safe models can produce emergent capabilities that neither possesses alone [8] — a composition problem no single-agent monitoring approach resolves. These failure modes are documented in the threat landscape mapped using MAESTRO [21] and MITRE ATLAS [22] in the prior governance framework [19], providing a threat-informed baseline on which GUARD's detection taxonomy is built.

2.2. Monitoring and IR Framework Limitations

MLOps [9], LLMOps [11], and AgentOps [10] all share the assumption that system behavior can be characterized by pre-defined metrics — an assumption NIST AI 800-4 identifies as invalid for agentic systems [12]. Yampolskiy (2025) demonstrates that AI systems can modify behavior when monitored, undermining observation-based detection [14]. Baker et al. (2025) introduce the monitorability tax — the performance cost of maintaining agent observability [15]. Existing usage-level monitoring approaches such as Clio [18] analyze aggregate conversation patterns across millions of interactions but do not validate individual tool invocations within agentic workflows, leaving the enforcement gap that RBEC addresses. For IR, O'Brien et al. (2023) propose Deployment Corrections at the model level [16], and Longpre et al. (2025) document the absence of centralized AI flaw reporting infrastructure [17]. Chan et al. (2024) confirm that agent identifier and watermarking mechanisms face fundamental reliability challenges in enterprise contexts [4]. Safe real-world testing of deployed agents remains an open challenge even with monitoring infrastructure in place, as agents may behave differently under test conditions than in production [5].

2.3. The Prior Governance Framework and the Gap This Paper Closes

Anuguthala (2026) [19] proposed the first enterprise agentic AI lifecycle governance framework integrating: mandatory system-of-record registration for all agentic use cases; a four-tier risk model ranging from Tier 1 (assistive, low-impact agents) to Tier 4 (fully autonomous agents with critical-system access); a RACI governance structure spanning seventeen lifecycle activities; phase-wise governance gates from planning through decommissioning; and AI-specific threat modeling via MAESTRO [21] and MITRE ATLAS [22]. That framework builds on foundational AI risk management guidance, including the NIST AI Risk Management Framework (AI RMF 1.0) [23], extending it with agentic-specific controls that the AI RMF does not prescribe. The framework's Phase 5 (Operational Monitoring) specifies continuous monitoring, kill-switch integration, and HITL controls as governance requirements, and its RACI table assigns incident response accountability to the Use Case Owner and Security team.

The limitation addressed by this paper is precise: the prior framework [19] specifies that monitoring criteria, kill-switch triggers, and HITL configurations must exist — but does not specify the runtime enforcement architecture that makes those definitions operational at execution time, the detection taxonomy that categorizes what is actually violated, or the structured IR lifecycle that converts monitoring events into governed responses. GUARD provides exactly these three missing components, transforming the governance commitments of [19] into an enforceable runtime system.

Three specific extensions distinguish GUARD from [19]. First, the SoR gains a three-entity registration model — adding inter-agent composition boundaries — to govern emergent multi-agent capabilities [8]. Second, the risk tier vocabulary is reconciled from [19]'s numeric Tier 1–4 to Low/Medium/High to enable standardized IR playbook templating, with Tier 1–2 mapping to Low, Tier 3 to Medium,

and Tier 4 to High. Third, the Lethal Trifecta — adapted from the concept articulated by Willison (2025) [20] and first applied to enterprise risk tiering in [19] — is here formalized as a runtime boundary enforcement instrument within RBEC.

3. The Agentic System of Record (SOR)

3.1. Overview and Design Principles

The SoR was introduced in [19] as a canonical enterprise inventory of agentic use cases, capturing unique identifiers, business owners, lifecycle phases, risk tiers, model and tool dependencies, data sources, autonomy levels, integration endpoints, and key control requirements. GUARD preserves this structure and extends it in two critical dimensions: the SoR becomes a dual-purpose artifact — governance documentation and the authoritative runtime enforcement reference consulted by RBEC before every agent action — and its registration model expands from individual use cases to three interconnected entity types. In practical terms, every time an agent attempts to invoke a tool or access a resource, RBEC queries the SoR to determine whether that action is pre-approved; the SoR answer is final and cannot be bypassed at runtime.

Three design principles, each directly addressing NIST AI 800-4 documented gaps [12], govern the extended SoR: (1) Shift-Left Monitoring Definition — monitoring triggers, escalation paths, and IR playbook assignments are mandatory SoR fields completed before production approval; (2) Use-Case Specificity with Structured Standardization — each use case defines its own monitoring scope within a standardized template; and (3) Mandatory Lifecycle Gating — lifecycle transitions require complete SoR entries and governance committee approval, inheriting the gating model established in [19].

3.2. Three-Entity Registration Model

The prior framework [19] registers use cases at the workflow level. GUARD extends this to three interconnected entities to address the emergent composition problem identified by Jones et al. [8] — individually compliant agents producing unapproved composite effects:

Individual Agent Registration captures for each agent (orchestrator or subagent): unique agent ID; parent workflow ID; functional role; autonomy level; approved tool set; approved data sources; approved external endpoints; privilege scope; risk tier; monitoring trigger set; HITL configuration and timeout threshold; use case owner; model version; and baseline behavioral profile. The Composition Boundary is not an attribute of an individual agent but a separate registration entity; it is described in the Composition Boundary Registration sub-section below. Table 1 presents mandatory fields with their origin relative to [19].

Table 1: Mandatory SOR Attributes - Origin Relative to Prior Framework [19]

SOR Attribute	Definition	Origin
Agent Identifier	Globally unique ID linking each agent to its parent workflow registration	Inherited from [19]; extended to per-agent granularity
Functional Role	Agent function within the workflow: Orchestrator | Retrieval | Executor | Reporter	New — GUARD adds role-level scope differentiation
Autonomy Classification	Four-level scale: Supervised | Semi-Autonomous | Autonomous | Fully Autonomous	Aligned with [19] Tier 1–4 autonomy definitions
Approved Tool Set	Exhaustive list of permitted tools with granular read / write / execute access per tool	Extended — [19] references tools at category level only
Approved Data Sources	Authorized data systems with access class: Read-Only or Read-Write	Inherited from [19] Phase 1 data governance approvals
Approved External Endpoints	Authorized external URLs and APIs; any unlisted endpoint triggers RBEC enforcement	New — enables Unauthorized External Endpoint Access (UEEA) detection
Privilege Scope	Maximum permission ceiling; all escalation permissions explicitly enumerated	New — enables Privilege Escalation Attempt (PEA) detection
Risk Tier	Lethal Trifecta-derived classification: Low | Medium | High	Reconciled from [19] Tier 1–4 for playbook templating
Monitoring Trigger Set	Pre-defined conditions activating monitoring events; mandatory before production entry	New — operationalizes [19] shift-left monitoring requirement
HITL Configuration	HITL mode enabled or disabled; approval timeout threshold in minutes	Extended from [19] HITL design requirement to runtime configuration
Baseline Behavioral Profile	Expected tool invocation frequency and data access patterns under normal operation	New — supports Lethal Trifecta Boundary Breach (LTBB) detection

Workflow Registration captures the composite entity: all participating agent identifiers; inter-agent communication topology (directed graph of permissible messaging); approved data flows between agents; combined impact area; business context boundary; governance committee approval record; and monitoring playbook reference. This extends [19]'s use case registration to multi-agent coordination governance.

Composition Boundary Registration is the novel addition directly addressing Jones et al.'s emergent composition problem [8]. Even when each agent operates within its approved scope, certain combined action sequences produce unapproved composite effects not detectable from individual agent records alone. This entity captures: prohibited combined actions; maximum combined privilege of the agent group; and cross-agent data flow restrictions. It is not present in [19] and represents GUARD's primary structural extension to the SoR.

3.3. The Lethal Trifecta — Risk Tier Classification

The Lethal Trifecta concept — the compound risk arising from the simultaneous intersection of tool access breadth, autonomy level, and impact area — was adapted from the concept articulated by Willison (2025) [20] in the context of prompt injection vulnerabilities and first applied to enterprise agentic risk tiering in the prior framework [19]. GUARD formalizes it as a runtime boundary enforcement instrument within RBEC. Three dimensions are rated: Tool Access Scope (T), from T1 (read-only, low-sensitivity sources) to T4 (write operations on critical or regulated systems); Autonomy Level (A), from A1 (all actions human-confirmed) to A4 (fully autonomous); and Impact Area (I), from I1 (isolated internal systems) to I4 (enterprise-wide, regulated data, external stakeholders). Risk tier $RT = f(T,A,I)$: Low when $\max(T,A,I) \leq 2$; High when any dimension ≥ 4 or $T \times A \times I \geq 24$; Medium otherwise. The threshold of 24 is not arbitrary — it represents the minimum product of a Tier 4 equivalent configuration: the autonomous refund agent case study in [19], characterized by T=3 (writes to payment systems), A=4 (fully autonomous execution), and I=2 (financial customer impact), yields exactly $T \times A \times I = 24$, the lowest compound score observed to produce critical combined risk in that enterprise financial services deployment. This threshold serves as a calibration heuristic; organizations should adjust it to reflect their specific risk appetite and operational context. This maps to [19]'s Tier 1–2 → Low, Tier 3 → Medium, Tier 4 → High.

3.4. Governance Roles

GUARD preserves the role definitions from [19]'s RACI model. The Agentic AI Governance Owner establishes and maintains the SoR and enforces lifecycle gating. The Use Case Owner (UCO) ensures SoR completeness, responds to HITL approval requests within defined timeouts, and leads root cause analysis. The Tech Risk Professional (TRP) is assigned to Medium-tier use cases for monitoring trigger set design and playbook customization — extending [19]'s RACI assignment for 'Continuous monitoring set up' to include GUARD's pre-production monitoring definition requirement.

4. Registry-Bound Execution Control (RBEC)

4.1. Three-State Enforcement Model

RBEC operationalizes the SoR from a governance artifact into an active enforcement boundary. Before any agent resource access — tool invocation, data source query, external endpoint contact, or privilege exercise — the RBEC validation hook intercepts the call and enforces one of three states:

- State 1 - Allow: The resource is in the agent's SoR-approved set. Execution proceeds: the action is logged to the immutable monitoring record.
- State 2 - HITL Pause: The resource is not approved and HITL mode is enabled. The workflow suspends; the Use Case Owner receives a structured notification containing the agent ID, the unapproved resource requested, a chain-of-thought excerpt where extractable, the workflow state at suspension, and the approval timeout. Owner approval resumes execution and triggers an SoR update request. Denial or timeout escalates to State 3.
- State 3 - Kill-Switch: The resource is not approved and HITL is disabled (fully autonomous mode), the timeout expired, or the request was denied. The workflow terminates immediately; state is preserved for forensic analysis; an incident record is auto-created with severity mapped to the use case risk tier.

This model directly addresses the NIST AI 800-4 gap of distinguishing security incidents from legitimate scope gaps [12]. A kill-switch without prior HITL review is classified as a potential security incident; one following HITL denial or timeout is a scope gap — each activating a different AIR lifecycle pathway (Section 5). The three-state model also extends [19]'s HITL design control from a Phase 2 development requirement into an operational runtime enforcement mechanism.

4.2. Violation Detection Taxonomy

RBEC defines five violation categories, each grounded in the threat landscape mapped using MAESTRO [21] and MITRE ATLAS [22] in the prior framework [19]:

- UTI - Unauthorized Tool Invocation: The agent attempts to invoke a tool not in its approved set. The most common violation type, arising from LLM reasoning drift toward unapproved tools during planning.
- UDSA - Unauthorized Data Source Access: The agent queries or writes to a data system not in its approved sources, including through an approved tool accessing an unapproved target path or directory.
- UEEA - Unauthorized External Endpoint Access: The agent contacts an external URL or API not in its approved endpoint list. This is the primary data exfiltration risk vector in enterprise agentic deployments.
- PEA - Privilege Escalation Attempt: The agent operates at a privilege level exceeding its registered scope, including spawning subagents with broader permissions than its composition boundary permits.

- LTBB - Lethal Trifecta Boundary Breach: The agent's combined runtime profile of tool access scope, autonomy level, and impact area exceeds its registered bounds, detected through correlation across individually approved actions in the monitoring record. This category addresses the scheming and deceptive alignment failure modes documented by Balesni et al. [6] and Meinke et al. [7], where no single action triggers a violation but the cumulative behavioral pattern breaches the approved boundary.

4.3. Architectural Implementation

RBEC is implemented at the tool wrapper layer: every tool is wrapped with a synchronous pre-execution hook that (1) extracts the agent ID from execution context, (2) queries the SoR for the agent's approved resource set, (3) routes to State 1, 2, or 3, and (4) writes the decision, violation type, workflow state snapshot, and chain-of-thought excerpt to the immutable monitoring record. Inter-agent communication is validated via a message-passing interceptor that checks the registered communication topology before forwarding any agent-to-agent message. This architecture is orchestration-framework-agnostic — deployable in LangGraph, AutoGen, or CrewAI — and directly implements the 'comprehensive logging and tracing' and 'agent identity and entitlements' baseline controls specified in [19].

5. The Agentic Incident Response (Air) Lifecycle

5.1. Overview

The prior framework [19] assigns incident response accountability in its RACI table — the Use Case Owner as Responsible and the Security team as Accountable/Responsible — but does not specify the IR procedure itself. GUARD defines a six-phase AIR lifecycle anchored to the SoR at every stage. Table 2 summarizes each phase, its primary actors, and its governance anchor — the specific requirement from [19] that this phase operationalizes.

Table 2: AIR Lifecycle - Governance Anchor Maps Each Phase to the Specific Requirement in [19] It Operationalizes

Phase	Primary Activity	Actor(s)	Governance Anchor
1. Detection	RBEC kill-switch trigger or telemetry anomaly alert	RBEC (auto) / UCO	Operationalizes [19] Phase 5 continuous monitoring requirement; automated to address NIST AI 800-4 scaling barrier [12]
2. Triage	Classify Category A (Security) or B (Scope Gap); assign priority P1–P4	UCO + TRP (M-tier)	Operationalizes [19] RACI: UCO Responsible, Security Accountable; GUARD adds structured classification criteria
3. Containment	Verify workflow shutdown; isolate preserved state; assess lateral impact	UCO + Security	Uses SoR workflow topology and composition boundary; extends [19] kill-switch requirement to structured containment
4. Root Cause Analysis	Determine why agent requested unapproved resource; classify root cause type	UCO + TRP / Security	Uses RBEC monitoring record and CoT excerpt against SoR scope; connects to [19] MAESTRO / MITRE ATLAS threat model
5. Recovery	Exception re-approval pathway (Scope Gap) or security remediation (Security Incident)	UCO + Gov. Committee	SoR update workflow; expanded scope triggers [19] risk re-tiering and re-approval gate
6. Post-Incident Review	Update SoR triggers and playbook; feed lessons into pre-deployment testing	UCO + Gov. Owner	Post-Incident Report feeds back to SoR and [19] Phase 1 threat model, closing the evaluation feedback loop [12]

5.2. Phase Details

5.2.1. Detection Purpose

To identify a violation event without requiring human observation. How it works: Every RBEC State 3 kill-switch event automatically generates a timestamped incident record in the monitoring store, making detection instantaneous and audit-ready without manual intervention. This directly addresses NIST AI 800-4's documented barrier of scaling human-driven monitoring alongside rapid agent rollouts [12]. Supplementary telemetry-based LTBB correlation compares the agent's cumulative runtime behavioral profile against its registered baseline to catch gradual boundary drift before a full violation triggers.

5.2.2. Triage Purpose

To classify the incident correctly so the right response is activated — avoiding both under-reaction to genuine threats and over-reaction to benign scope gaps. How it works: Incidents are classified into two mutually exclusive categories. Category A (Security Incident) applies when a State 3 kill-switch fires without prior HITL review, or when the violation type is PEA or LTBB — these receive immediate escalation to the TRP and the Governance Owner. Category B (Scope Gap Incident) applies when the kill-switch follows HITL timeout or denial, or when the violation is UTI or UDSA with evidence the request arose from legitimate LLM reasoning rather than adversarial input — these are managed by the UCO with TRP validation. A priority level P1 through P4 is assigned based on risk tier and violation type, setting the response time SLA. This classification prevents conflating every scope gap with a security breach, directly addressing the NIST AI 800-4 alert fatigue problem [12].

5.2.3. Root Cause Analysis Purpose

To determine exactly why the agent attempted an unapproved action, enabling targeted remediation rather than blanket restrictions. How it works: The UCO reviews the RBEC monitoring record — including the workflow state snapshot and chain-of-thought excerpt logged at the moment of violation — against the agent's registered SoR scope. Common root causes include: LLM reasoning drift (the most frequent cause, mapped to MAESTRO reasoning layer threats in [19]); prompt injection (adversarial input, a documented MITRE ATLAS [22] technique); subagent permission inheritance errors; and environmental change (a registered endpoint or data source has moved). The prior framework's [19] Tier 4 autonomous refund agent case study illustrates this directly: prompt injection, delegated privilege abuse, and runaway action loops were identified as primary threats via MAESTRO and MITRE ATLAS, mapping to GUARD's UDSA, PEA, and LTBB categories respectively.

5.2.4. Recovery Purpose

To restore safe operation through either a structured exception pathway or a security remediation process. How it works: Category B (Scope Gap) follows an exception re-approval pathway — the UCO assesses whether the unapproved resource is legitimate, initiates an SoR update request if so, the governance committee validates, the resource is added to the agent's approved set, and the workflow resumes. If the expanded scope elevates the Lethal Trifecta score, [19]'s risk re-tiering and re-approval process is triggered before resumption. Category A (Security Incident) follows enterprise security IR with AI-specific additions: model and system prompt review for adversarial influence, and enterprise-wide impact assessment across use cases sharing the same foundation model.

5.2.5. Post-Incident Review Purpose:

To prevent recurrence by feeding real incident data back into the governance model. How it works: Within a defined window after incident closure, the UCO and Governance Owner produce a Post-Incident Report covering timeline, root cause, control effectiveness, and gaps in the SoR monitoring trigger set. Findings update the SoR directly and feed back into [19]'s Phase 1 threat model, enriching the MAESTRO and MITRE ATLAS threat mappings with real observed incident data for future use cases. This creates the evaluation feedback loop identified as a key field goal by NIST AI 800-4 [12].

5.3. Risk-Tiered Playbook Templates

- Low Risk Tier (maps to [19] Tier 1–2): All AIR phases assigned to UCO; automated RBEC detection only; triage SLA 48 hours; lightweight post-incident retrospective submitted to the Governance Owner.
- Medium Risk Tier (maps to [19] Tier 3): TRP co-ownership from triage onward; triage SLA 4 hours (P1–P2) / 24 hours (P3–P4); mandatory LTBB telemetry correlation alongside RBEC detection; formal RCA documentation reviewed by TRP; structured post-incident review with Governance Owner.
- High Risk Tier (maps to [19] Tier 4): Out of scope for standardized templates. Requires bespoke IR design with domain expertise and regulatory engagement. The prior framework's [19] Tier 4 autonomous refund agent case study — involving independent model validation, multiple HITL checkpoints, and regulatory compliance requirements — illustrates the complexity that makes generalized templates insufficient for this tier.

6. Reference Implementation and Empirical Evaluation

6.1. Reference Implementation

RBEC was implemented on LangGraph, selected for its explicit control over tool invocation sequencing, conditional edge execution enabling the State 2 HITL pause pathway, and established enterprise adoption. The reference implementation comprises three components. First, the Agentic System of Record is instantiated as a SQLite database with three tables corresponding to the three registration entities: Agent, Workflow, and CompositionBoundary. A Python query API exposes approval lookups to the RBEC hook; an in-memory cache retains the results of recently approved resource lookups so that repeated accesses to the same pre-approved resource by the same agent do not incur a new database round-trip on every call. Second, the RBEC Validation Hook is implemented as a Python decorator applied to every LangGraph tool function, intercepting the call before execution and routing to State 1, 2, or 3 based on the SoR lookup result; an inter-agent communication validator implemented as a LangGraph edge interceptor enforces the registered composition boundary before forwarding any agent-to-agent message. Third, an immutable Monitoring Record Store captures every RBEC decision — including violation type, workflow state snapshot, chain-of-thought excerpt where extractable, and incident identifier — as the evidentiary basis for all AIR lifecycle phases.

The reference workflow registers a three-agent enterprise HR data analytics use case in the SoR: a master orchestrator responsible for planning, a retrieval subagent authorized to query the HR database and read approved report files, and a reporting subagent authorized to format and deliver outputs to an approved internal dashboard endpoint. A composition boundary explicitly prohibits cross-agent transmission of raw personally identifiable information fields.

Experimental configuration: All measurements were conducted on a 13th Gen Intel Core i7-1355U with 5.6 GB RAM running WSL2 (Linux kernel 6.6.87.2, Microsoft-standard-WSL2) with Python 3.14.4 (GCC 15.2.0). The SoR SQLite database ran in-process on the same host as the RBEC validation hook. All reported latency values are wall-clock measurements from hook entry to State decision return, inclusive of SoR lookup and monitoring record write. Raw experimental data are available from the corresponding author upon reasonable request.

6.2. Evaluation Scenarios and Results

Seven scenarios evaluate RBEC: Scenario 0 (Baseline) tests fully approved operation to establish the false-positive rate; Scenarios 1–4 each introduce one violation type (UTI, UDSA, UEEA, PEA) with HITL disabled to test State 3 kill-switch detection; Scenario 5 repeats the UTI violation with HITL enabled to test the State 2 pause pathway; and Scenario 6 tests Lethal Trifecta Boundary Breach (LTBB) through a sequence of three individually approved tool invocations whose cumulative T×A×I score reaches 24, triggering breach detection through post-hoc monitoring record correlation. Each scenario runs N = 100 trials for robust latency measurement. Table 3 reports results across six columns: Violation identifies which of the five RBEC violation categories (defined in §4.2) is triggered; Expected and Observed States confirm detection correctness (State 1 = allow, State 2 = HITL pause, State 3 = kill-switch); Detection Outcome captures both the correctness verdict and whether chain-of-thought (CoT) text was extractable; and Hook Latency reports the mean wall-clock time in milliseconds, representing the operational overhead RBEC adds to every agent tool call, and p95 Latency reports the 95th-percentile wall-clock time in milliseconds, representing the realistic worst-case overhead across the trial distribution.

Table 3: RBEC Empirical Evaluation Results - 13th Gen Intel Core I7-1355U, WSL2 Linux 6.6.87.2, Python 3.14.4, Sqlite In-Process, N=100 Trials Per Scenario. LTBB Latency Reflects the Correlation Check Only, not a Pre-Execution Hook Intercept

Scenario	Violation Type	Expected State	Observed State	Detection Outcome	Mean (ms)	p95 (ms)
0 — Baseline	None (all approved)	State 1	State 1	0 of 100 false positives — all approved operations passed without interruption	7.44 (±2.05)	12.24
1 — UTI	Unauthorized Tool Invocation	State 3	State 3	Correct — kill-switch fired; incident auto-created; CoT extractable from planning step	6.64 (±1.03)	8.73
2 — UDSA	Unauthorized Data Source Access	State 3	State 3	Correct — kill-switch fired; incident auto-created; CoT absent (access via tool parameter — confirms logging granularity challenge)	6.25 (±0.86)	7.75
3 — UEEA	Unauthorized External Endpoint Access	State 3	State 3	Correct — kill-switch fired; endpoint blocked pre-execution; CoT extractable	6.30 (±0.92)	8.00
4 — PEA	Privilege Escalation Attempt	State 3	State 3	Correct — composition boundary validator confirmed subagent over-privilege; incident created; CoT extractable	6.63 (±0.94)	8.20
5 — HITL	UTI with HITL enabled	State 2	State 2	Correct — workflow suspended; owner notification dispatched in 6.63 ms; CoT included in notification	6.63 (±0.89)	8.22
6 — LTBB	Lethal Trifecta Boundary Breach	State 3	State 3	Correct — cumulative T×A×I=24 exceeded threshold; breach detected by monitoring record correlation; CoT extractable	0.46 (±0.15)	0.73

6.3. Measurement Summary Statistics

Table 4 Presents the Complete Latency Distribution across All Scenarios. All Values are In Milliseconds

Table 4: Complete Latency Distribution - All Values In Milliseconds; Experimental Configuration as Stated in §6.1. LTBB Measures Correlation Check Latency only

Scenario	N Trials	Mean	Std Dev	Min	Median	p95	Max
S0 — Baseline	100	7.44	2.05	4.41	6.84	12.24	15.05
S1 — UTI	100	6.64	1.03	4.72	6.50	8.73	10.18
S2 — UDSA	100	6.25	0.86	4.94	6.10	7.75	9.16
S3 — UEEA	100	6.30	0.92	4.62	6.29	8.00	9.20
S4 — PEA	100	6.63	0.94	4.67	6.63	8.20	9.55
S5 — HITL	100	6.63	0.89	4.89	6.63	8.22	9.30
S6 — LTBB	100	0.46	0.15	0.20	0.45	0.73	1.07
Cache (warm, n=200)	200	8.14	1.05	5.57	8.17	10.01	11.58
Overall (S0–S6)	700	5.764	—	0.20	—	—	15.05

6.4. Discussion

6.4.1. Detection Accuracy

RBEC correctly detected all five violation categories across seven scenarios with zero false positives on the approved-action baseline across 100 trials, confirming that a properly registered use case incurs no operational disruption from RBEC enforcement. All State 3 violations in Scenarios 1–4 were intercepted at the pre-execution hook before the tool function executed, ensuring the unapproved resource was not accessed. In Scenario 6 (LTBB), detection operates through post-hoc correlation of the monitoring record rather than a single hook intercept — the cumulative T×A×I score of 24 was computed across three approved actions and correctly identified as a boundary breach in all 100 trials.

6.4.2. Latency Overhead

The RBEC validation hook introduced a mean latency of 5.764 ms per tool invocation across all seven scenarios (N = 100 trials per scenario; 13th Gen Intel Core i7-1355U; WSL2 Linux 6.6.87.2; SQLite in-process). The maximum p95 latency across all hook-intercept scenarios was 12.24 ms, observed in Scenario 0 (Baseline), reflecting natural variation in SQLite read latency under the initial cold-cache conditions of the first scenario run. The LTBB correlation check in Scenario 6 produced a mean of 0.46 ms — substantially lower because it measures a single SQLite aggregate query and arithmetic comparison rather than a full pre-execution hook intercept; in a production deployment this check would run as a background monitoring process rather than inline with each tool call. The warm repeated-lookup mean of 8.14 ms (±1.05) is comparable to the scenario means, indicating that the in-memory cache provides no measurable latency advantage at this scale of SQLite operation; this is attributed to accumulated Write-Ahead Log (WAL) activity from preceding scenario runs, which increases SQLite read latency under sustained write load. The cache benefit becomes more significant in network-attached database configurations where round-trip overhead dominates. Given that enterprise agentic tool calls typically involve network round-trips of hundreds of milliseconds to seconds, the RBEC validation overhead is operationally acceptable for production deployment.

6.4.3. Chain-of-Thought Extractability

CoT was extractable in 4 of 5 violation types (UTI, UEEA, PEA, and LTBB). In Scenario 2 (UDSA — Unauthorized Data Source Access), no CoT was captured because the unapproved file path was passed as a parameter to the approved read_file tool rather than appearing in the agent's explicit planning step. This confirms a real forensic limitation: parameter-driven violations are harder to explain post-hoc than planning-step violations, motivating future work on parameter-level CoT attribution. For LTBB, CoT was extractable because the breach notification logged the cumulative score computation and reasoning excerpt from each of the three approved actions that contributed to the boundary crossing.

6.4.4. Composition Boundary Validation

Scenario 4 (PEA — Privilege Escalation Attempt) required the inter-agent communication validator to check the composition boundary before allowing the subagent spawn. The violation was correctly detected at 6.63 ms mean latency, validating the three-entity registration model's effectiveness for the emergent composition problem identified by Jones et al. [8].

6.4.5. Lethal Trifecta Boundary Breach Detection

Scenario 6 evaluated the LTBB violation category, which differs fundamentally from Scenarios 1–4. Rather than intercepting a single unauthorized resource access at the pre-execution hook, LTBB detection operates through post-hoc correlation of the monitoring record across a sequence of individually approved actions. The orchestrator agent executed three approved tool invocations — a database query, a report formatting operation, and a subagent spawn — whose cumulative Trifecta dimensions of T=3 (write and spawn tools observed), A=4 (fully autonomous, no HITL confirmation), and I=2 (HR data, internal scope) yielded a product of 24, meeting the registered threshold. Breach detection was confirmed in all 100 trials at a mean correlation latency of 0.46 ms (±0.15, p95=0.73 ms). This result validates that RBEC's behavioral correlation mechanism correctly identifies cumulative boundary breaches that no individual hook intercept would have detected, directly addressing the scheming and deceptive alignment failure modes identified by Balesni et al. [6] and Meinke et al. [7].

False Positive Verification: A separate false-positive sweep tested all nine individually approved resource accesses across all three registered agents. All nine returned State 1 (Allow), confirming a false-positive rate of zero across the complete approved resource set.

7. Limitations and Future Work

7.1. Third-Party Agent Monitoring

GUARD cannot deploy RBEC hooks inside vendor-procured black-box agents. The prior framework [19] requires third-party agentic workflows to be registered in the SoR — GUARD partially enforces this at the network perimeter via API gateway and access control configuration, but enforcement granularity is lower than tool-wrapper RBEC. Vendor contractual requirements for monitoring transparency and an RBEC-compatible proxy architecture for third-party agents are priority future work.

7.2. High Risk Tier (Tier 4 per [19])

Standardized playbook templates cover Low and Medium tiers only. The prior framework's [19] Tier 4 autonomous refund agent case study — requiring multiple independent control layers, independent model validation, and regulatory engagement — illustrates why generalized IR templates are insufficient. High Risk tier IR framework design, in partnership with sector-specific regulators, is planned.

7.3. LLM Reasoning Opacity

RBEC enforces boundaries on observable actions but cannot inspect internal model reasoning. A sophisticated scheming agent [6, 7] could achieve unapproved objectives through sequences of individually approved actions without triggering RBEC's kill-switch. The LTBB violation type partially addresses this through behavioral correlation, but

effectiveness depends on the quality of the registered baseline behavioral profile.

7.4. Evaluation Scale

The reference implementation is evaluated with a single use case and controlled violation scenarios on a local machine. Production-scale evaluation across real enterprise agentic workflows with concurrent agents, network-attached databases, and organic violation patterns is the next validation phase. Planned work includes retrospective application of GUARD to the Tier 4 refund agent scenario described in [19].

8. Conclusion

This paper presents GUARD — Governance-Unified Agentic Runtime Detection and Response — the second in a series of two papers on enterprise agentic AI governance and enforcement. The prior governance framework [19] established the governance architecture: what must be registered, by whom, under what risk tier, and through what lifecycle checkpoints. GUARD operationalizes that architecture into runtime enforcement and incident response: how every agent action is validated at execution time, what happens when the approved boundary is breached, and how incidents are detected, triaged, contained, investigated, and remediated.

GUARD's central innovation is the reframing of the agentic monitoring problem. Rather than attempting to statistically detect anomalies in a fundamentally non-deterministic system, GUARD enforces a pre-approved behavioral envelope defined through mandatory SoR registration. Three formal constructs extend the state of the art beyond [19]: the three-entity SoR registration model addressing emergent multi-agent composition; the three-state enforcement model distinguishing security incidents from legitimate scope gaps; and the formalization of the Lethal Trifecta — adapted from the concept articulated by Willison (2025) [20] and first applied to risk tiering in [19] — as an operational runtime boundary enforcement instrument.

Empirical evaluation across seven scenarios confirms correct detection of all five violation categories — Unauthorized Tool Invocation (UTI), Unauthorized Data Source Access (UDSA), Unauthorized External Endpoint Access (UEEA), Privilege Escalation Attempt (PEA), and Lethal Trifecta Boundary Breach (LTBB) — with zero false positives across 100 trials per scenario on a 13th Gen Intel Core i7-1355U. Notably, LTBB detection operates through monitoring record correlation rather than single-hook interception, and chain-of-thought reasoning was extractable for 4 of 5 violation types, confirming that the framework supports forensic root cause analysis for the majority of violation categories it detects. Together with the prior governance framework [19], GUARD provides a complete, implementable foundation for governed enterprise agentic AI deployment: from the governance architecture that defines what is permitted, through the runtime system that enforces those permissions, to the incident response procedures that govern what happens when they are violated.

Data Availability Statement

The experimental data underlying the results presented in this paper, including the raw latency measurements, monitoring records, and evaluation scenario outputs, are available from the corresponding author upon reasonable request. The reference implementation comprising the System of Record database module, the RBEC validation engine, and the evaluation scenario harness is also available from the corresponding author upon reasonable request for reproducibility purposes.

References

1. A. Bick, A. Blandin, and D. J. Deming, "The Rapid Adoption of Generative AI," NBER Working Paper 32966, Feb. 2025. doi: 10.3386/w32966.
2. N. Maslej et al., "Artificial Intelligence Index Report 2025," arXiv:2504.07139, 2025.
3. M. Zaharia et al., "The Shift from Models to Compound AI Systems," Berkeley Artificial Intelligence Research Blog, Feb. 2024. [Online]. Available: https://bair.berkeley.edu/blog/2024/02/18/compound-ai-systems/ [Accessed: May 2026].
4. A. Chan et al., "Visibility into AI Agents," in Proc. ACM FAccT, pp. 958–973, 2024. doi: 10.1145/3630106.3658948.
5. S. Naihin et al., "Testing Language Model Agents Safely in the Wild," arXiv:2311.10538, Dec. 2023. doi: 10.48550/arXiv.2311.10538.
6. M. Balesni et al., "Towards Evaluations-Based Safety Cases for AI Scheming," arXiv:2411.03336, Nov. 2024.
7. A. Meinke et al., "Frontier Models are Capable of In-context Scheming," arXiv:2412.04984, Jan. 2025.
8. E. Jones, A. Dragan, and J. Steinhardt, "Adversaries Can Misuse Combinations of Safe Models," arXiv:2406.14595, Jul. 2024.
9. D. Kreuzberger, N. Kühl, and S. Hirschl, "Machine Learning Operations (MLOps): Overview, Definition, and Architecture," IEEE Access, vol. 11, pp. 31866–31879, 2023. doi: 10.1109/ACCESS.2023.3262138.
10. L. Dong, Q. Lu, and L. Zhu, "AgentOps: Enabling Observability of LLM Agents," arXiv:2411.05285, Nov. 2024.
11. K. Huang, V. Manral, and W. Wang, "From LLMOps to DevSecOps for GenAI," in Generative AI Security: Theories and Practices, Springer, Cham, 2024, pp. 241–269. doi: 10.1007/978-3-031-54252-7_8.
12. A. K. Rao et al., "Challenges to the Monitoring of Deployed AI Systems," NIST AI 800-4, National Institute of Standards and Technology, Mar. 2026. doi: 10.6028/NIST.AI.800-4.
13. R. Shah et al., "Goal Misgeneralization: Why Correct Specifications Aren't Enough For Correct Goals," arXiv:2210.01790, Nov. 2022.
14. R. V. Yampolskiy, "On Monitorability of AI," AI Ethics, vol. 5, no. 1, pp. 689–707, Feb. 2025. doi: 10.1007/s43681-024-00420-x.
15. B. Baker et al., "Monitoring Reasoning Models for Misbehavior and the Risks of Promoting Obfuscation," arXiv:2503.11926, Mar. 2025.

16. J. O'Brien, S. Ee, and Z. Williams, "Deployment Corrections: An Incident Response Framework for Frontier AI Models," arXiv:2310.00328, Sep. 2023.
17. S. Longpre et al., "In-House Evaluation Is Not Enough: Towards Robust Third-Party Flaw Disclosure for General-Purpose AI," in Proc. ICML Position Paper Track, 2025.
18. A. Tamkin et al., "Clio: Privacy-Preserving Insights into Real-World AI Use," arXiv:2412.13678, Dec. 2024.
19. S. K. Anuguthala, "Enterprise Agentic AI Lifecycle Governance: A Control-Driven Framework from Design to Decommissioning," International Journal of Artificial Intelligence, Data Science, and Machine Learning, vol. 7, no. 2, pp. 9–16, Apr. 2026. doi: 10.63282/3050-9262.IJAIDSML-V7I2P103.
20. S. Willison, "The Lethal Trifecta for AI Agents: Private Data, Untrusted Content, and External Communication," Simon Willison's Weblog, Jun. 16, 2025. [Online]. Available: https://simonwillison.net/2025/Jun/16/the-lethal-trifecta/ [Archived: https://web.archive.org/web/20260428130356/https://simonwillison.net/2025/Jun/16/the-lethal-trifecta/] [Accessed: May 2026]. Backup: https://simonw.substack.com/p/the-lethal-trifecta-for-ai-agents
21. Cloud Security Alliance, "MAESTRO: Multi-Agent Environment Security Threat and Risk Oracle — Agentic AI Threat Modeling Framework," AI Safety Initiative Working Group, 2025. [Online]. Available: https://cloudsecurityalliance.org/blog/2025/04/10/introducing-maestro-a-framework-for-securing-multi-agent-ai-systems [Accessed: May 2026].
22. MITRE Corporation, "Adversarial Threat Landscape for Artificial-Intelligence Systems (ATLAS)," MITRE, 2024. [Online]. Available: https://atlas.mitre.org/ [Accessed: May 2026].
23. NIST, "Artificial Intelligence Risk Management Framework (AI RMF 1.0)," National Institute of Standards and Technology, Gaithersburg, MD, Jan. 2023. doi: 10.6028/NIST.AI.100-1. [Online]. Available: https://doi.org/10.6028/NIST.AI.100-1