
(Research Article)

Evaluating Folder-Oriented Governance for Agentic AI with Automated SIEM-Driven Alerts

Dr Gaston Elongha

Marymount University

ORCID iD: <https://orcid.org/0009-0006-4127-9783>

Journal of Information Technology, Cybersecurity, and Artificial Intelligence, 2026, 3(5), Pages 17-24

DOI: <https://doi.org/10.70715/jitcai.2026.v3.i5.095>

Abstract

Agentic AI can support security operations center (SOC) workflows, but ungoverned autonomy creates risks of unsafe action, weak traceability, and limited runtime control. This paper extends a prior folder-oriented governance prototype by replacing manual alerts with automated Wazuh SIEM ingestion while preserving folder-scoped policy-as-code, runtime guardrails, escalation, and structured audit logging. A controlled comparison with an unconstrained baseline evaluates policy effectiveness, escalation, auditability, and latency. The governed pipeline blocked 90 disallowed actions, produced complete audit traces, and consistently routed 75% of evaluated alerts to draft-only review states, with moderate latency overhead. In this proof-of-concept evaluation, these results suggest that folder-oriented governance can preserve bounded autonomy and accountability under automated alert ingestion.

Keywords: *Agentic AI governance; Policy-as-code; SOC automation; SIEM integration; Runtime guardrails; Auditability*

1. INTRODUCTION

Agentic AI can analyze context, use tools, and support multi-step SOC workflows, but its autonomy also creates risks involving unsafe actions, data leakage, weak accountability, and insufficient runtime control [1]-[3]. Prior work introduced a folder-oriented governance pattern that packages policies, prompts, tests, and logs as an auditable deployment unit and demonstrated feasibility with three manual SOC alert scenarios [1]. That study did not evaluate automated alert ingestion or broader scenario coverage.

This paper addresses that limitation by integrating automated SIEM-driven alert ingestion into the same governed SOC prototype while retaining policy-as-code, runtime enforcement, human-review escalation, and audit logging. It evaluates policy effectiveness, escalation behavior, auditability, and runtime overhead under broader operational input conditions [1], [2], [4].

The study asks whether folder-oriented governance (RQ1) continues to block disallowed actions under automated SIEM ingestion, (RQ2) preserves risk-based escalation and complete audit traces, and (RQ3) introduces acceptable runtime overhead. Accordingly, (H1) predicts continued policy compliance, (H2) predicts more frequent human review for higher-risk alerts, (H3) predicts complete audit traces, and (H4) predicts acceptable latency overhead.

This paper contributes: (1) an extension from manual alerts to automated SIEM ingestion; (2) broader evaluation of policy effectiveness, escalation, auditability, and latency; and (3) evidence that folder-scoped policy-as-code remains practical under more realistic SOC conditions. The remainder of the paper presents related work, system extension architecture, methodology and experiment design, results and discussion, and conclusion.

2. RELATED WORK & SYSTEM EXTENSION ARCHITECTURE

Agentic AI governance research emphasizes runtime controls, accountability, action constraints, and monitoring as autonomy increases [2]-[4]. The earlier study positioned folder-oriented governance as a practical governance-by-design pattern that integrates policy-as-code, runtime checks, and audit logging with an auditable deployment structure [1].

Beyond our earlier prototype, recent work on agentic AI governance has examined complementary dimensions of runtime control and risk management. Abou Ali et al. provide a paradigm-aware survey of agentic AI architectures and applications, highlighting the need for governance mechanisms that match symbolic and neural agent behaviors in operational settings [6]. Shavit et al. describe practices for governing agentic systems, including suitability checks, constrained action spaces, monitoring, and graduated containment [3]. Wang et al. propose MI9 as an integrated runtime governance framework that introduces semantic telemetry, authorization monitoring, conformance checks, drift detection, and containment strategies for agentic AI [4].

Patil frames agentic governance more broadly through observability, identity, policy management, phased autonomy, and risk-based controls [5], while the NIST AI RMF emphasizes cross-cutting governance, lifecycle risk management, documentation, and monitoring for trustworthy AI systems [2]. The present study differs from these approaches by wrapping policy-as-code, runtime guardrails, and audit logging into a folder-scoped deployment unit that can be reviewed and adapted directly by SOC and governance teams [1].

Compared to these broader governance frameworks, the earlier folder-oriented prototype provided a concrete SOC-style implementation but remained methodologically limited to three manually defined alert scenarios, which established feasibility rather than performance under automated operational inputs [1]. The present study retains the same governance design and evaluates it with an automated SIEM-driven alert pipeline and more formalized measures [1], [4].

2.1. SYSTEM EXTENSION ARCHITECTURE

The extended system preserves the earlier folder-oriented governance structure but replaces manual alert generation with automated SIEM ingestion [1]. The folder remains the governance boundary containing task goals, policies, tests, and logs, while the orchestrator, guardrails, policy modules, and audit logger remain unchanged as shown in Fig. 1 and Fig. 2. The agent performs triage, enrichment, risk assessment, and draft-response generation but cannot execute privileged or final enforcement actions. Folder-scoped policies validate each step, and structured logs record inputs, decisions, tool calls, blocked actions, and escalations [1]-[3].

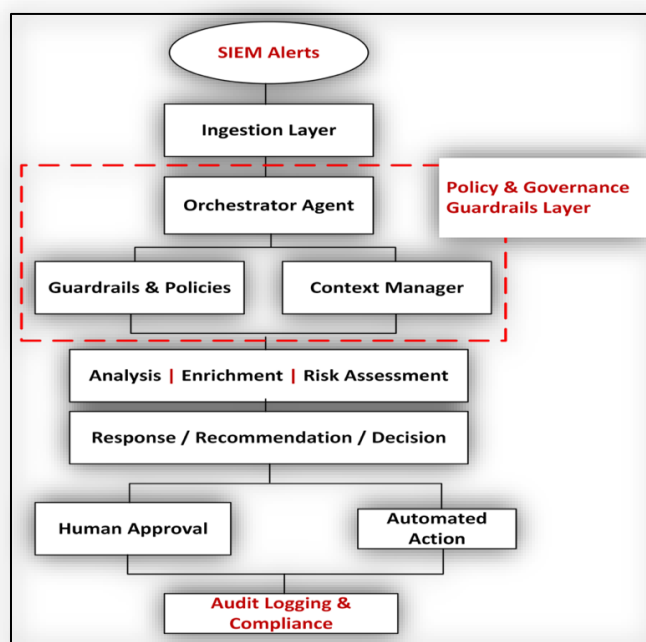


Figure 1 Architecture of the Folder-Governed SOC AI Agent

Fig. 1 summarizes the architecture: the orchestrator ingests SIEM alerts, applies policy and governance guardrails during enrichment and risk assessment, routes outputs to automated or human-reviewed response paths, and records the workflow in an audit log.

3. METHODOLOGY & EXPERIMENT DESIGN

A controlled proof-of-concept compares the governed agent with an unconstrained baseline using the same alert stream and task objectives [1]. In the baseline configuration, the same orchestrator and agent modules are used, but folder-scoped policies and guardrails are disabled. Tool access is not restricted by policy, escalation logic is not enforced, and audit logging is reduced to minimal diagnostic output rather than structured, policy-linked traces. The baseline therefore represents an unconstrained agentic workflow that processes alerts with the same model and tools as the governed pipeline but without explicit runtime governance controls.

In this study, the governed agent enforces tool restrictions, policy checks, escalation thresholds, and structured logging; the baseline processes alerts without these controls. Fig. 2 shows the controller workflow for low-, medium-, high-, and critical-severity alerts. The orchestrator is the sole execution controller and enforces input validation, context enrichment, alert analysis, risk assessment, response drafting, and audit logging. Agents therefore cannot select tools or actions outside predefined policy.

The enrichment agent gathers approved host, IP-geolocation, and MITRE ATT&CK context; the analysis agent assesses each alert; the risk agent assigns severity and review requirements; and the response agent produces recommendations without executing actions. This separation reserves high-impact decisions for human approval. Policy files allowed and prohibited actions, tool access, and escalation thresholds; guardrails sanitize inputs and block unauthorized operations; and log_pkg records each workflow step for audit and repeatability.

The prototype was implemented in Python using the controller.py workflow shown in the Visual Studio Code environment in Fig. 2. Experiments were executed on an Ubuntu host running the Wazuh server and collectors, with alerts supplied from the alerts/alerts.json file described in Table 1. The governed and baseline agents both relied on the same large language model backend (gpt-4o-mini) through the OpenAI API for enrichment, analysis, risk scoring, and draft response generation. Model temperature and other generation parameters were held constant across runs so that differences in behavior reflected governance controls rather than model configuration changes.

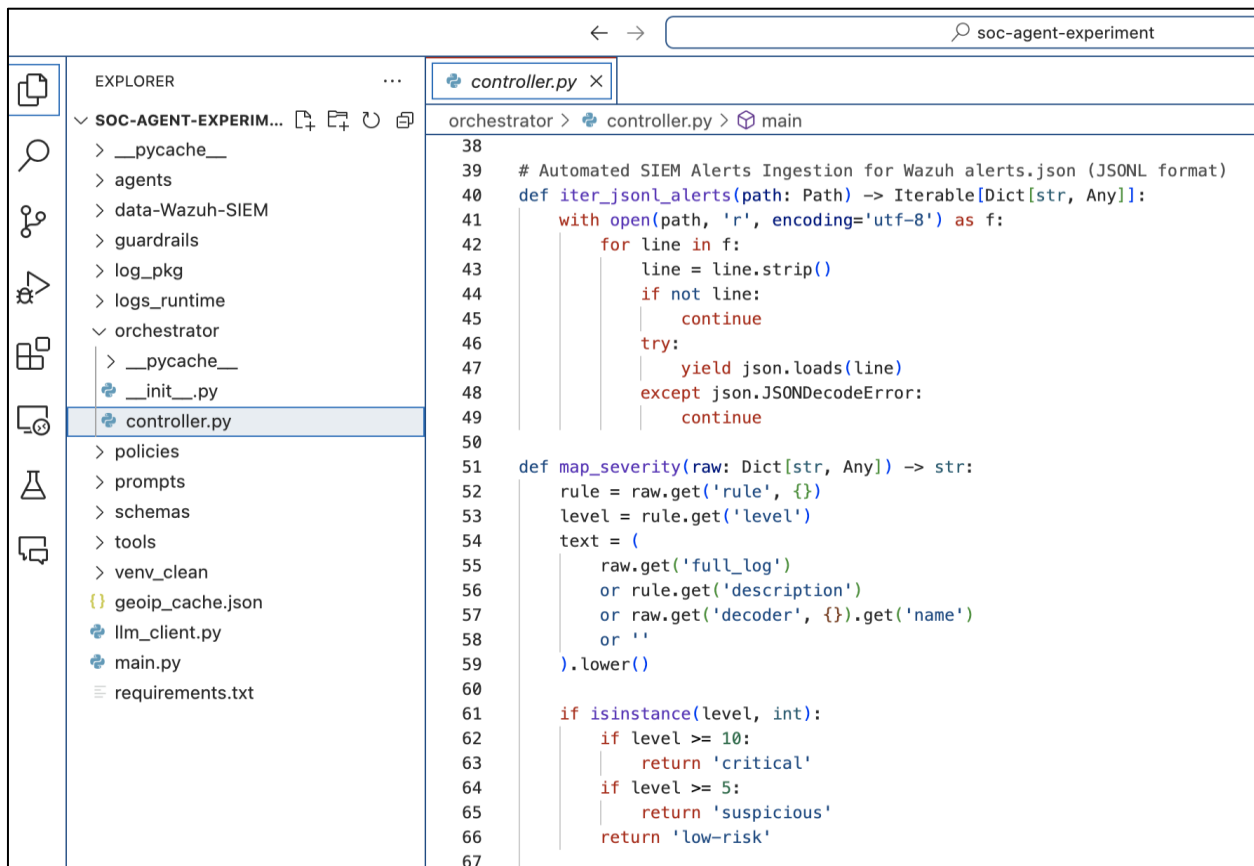


Figure 2 Orchestrator/controller.py Execution Flow (Experimental Setup)

The evaluation measures: policy effectiveness (blocked prohibited), escalation behavior (risk-based human review), auditability (complete decision traces), and runtime overhead (latency relative to baseline) [1], [2]. The experiment uses a broader severity-stratified alert set than the three manual scenarios in the earlier study, enabling repeated governed and baseline observations [1].

In this study, an alert was considered to have a complete audit trace when the audit log contained the full sequence of expected governance events: `controller.start`, `guardrail.inputvalidated`, `agent.enrichment.complete`, `agent.analysis.complete`, `agent.risk.complete`, `agent.response.complete`, and `controller.complete`, along with any `guardrail.toolallowed` or `guardrail.outputvalidated` entries generated during enrichment, risk scoring, or response drafting. Alerts that did not produce this full event sequence, or that omitted one or more guardrail validations, were classified as having incomplete audit traces.

Latency was measured as end-to-end processing time per alert from initial ingestion by the orchestrator to `controller.complete` in the audit log. Across the 120 evaluated alerts, the governed configuration produced a median latency of approximately 4 seconds per alert, while the unconstrained baseline showed a median latency of approximately 3 seconds per alert. These values correspond to the full multi-step workflow (validation, enrichment, analysis, risk, and response) and reflect the overhead introduced by policy checks and guardrail validations relative to the same model and tools without governance.

Table 1: Wazuh SIEM Environment and Alert Sample

Designation	Description
Platform	Wazuh
Collector	Ubuntu host; alerts/alerts.json
Endpoints	Three Ubuntu 26.04 LTS (64-bit) systems and three Windows 11 systems with Wazuh agents

Observed inventory	96 low, 785 medium, 714 high, and 135 critical alerts observed during June 30—July 12, 2026
Input/audit	JSON alerts normalized before orchestration; audit logs recorded validation through completion

Table 2: Alert Selection and Processing

Severity group	Wazuh volume observed	Sample used in the study experiment	Processing purpose
Low	96	30	Routine or lower-risk events used to test baseline governance behavior
Medium	785	30	Moderate-risk events used to test escalation and review behavior
High	714	30	Elevated-risk events used to test stricter control and analyst review
Critical	135	30	Highest-risk events used to test strong governance and audit traceability
Total	1730	120	Balanced evaluation set used to compare governed processing across severities

From the observed Wazuh inventory of 1 730 alerts (**96 low, 785 medium, 714 high, and 135 critical**), we constructed a laminated sample for the evaluation. Within each severity bucket, alerts were randomly selected subject to two constraints: **(1)** the normalized JSON record contained the fields required by the orchestrator (timestamp, asset identifier, severity, and raw log), and **(2)** duplicate or near-duplicate entries from the same host and timestamp were excluded. This yielded 30 alerts per severity group—30 low, 30 medium/suspicious, 30 high, and 30 critical—for a total of 120 alerts used in the governed and baseline runs.

4. RESULTS & DISCUSSION

The governed workflow preserved bounded autonomy and complete auditability under automated SIEM ingestion. Alerts passed through validation, enrichment, analysis, risk scoring, and response generation, but policy controls constrained final actions rather than allowing autonomous execution.

Table 3: Governed SOC Results

Alert Class	# Alerts	Baseline Blocked	Governed Blocked	Human Review Rate	Audit Completeness	Relative Latency
Low-risk	30	0	0	0%	Complete	Moderate
Medium/Suspicious	30	30	30	100%	Complete	Moderate
High + Critical	60	60	60	100%	Complete	Moderate

The reported total of 90 blocked actions is derived from the governed runs across the medium/suspicious, high, and critical severity groups. For medium/suspicious alerts, the governance policies prevented 30 attempted actions that would have violated tool-access or escalation rules. For high and critical alerts, the policies prevented 60 attempted actions, including direct containment or remediation steps that exceeded the allowed action space. Summed across these severity levels, the governed pipeline blocked 90 disallowed actions while still allowing evidence-gathering and draft-response generation to proceed.

A clear result is that the governed architecture did not allow the agents to take unrestricted operational actions after alert analysis. In the response outputs, the final response stated repeatedly shows `ticketactiondraftonly` and `finalstate readforreview`, indicating that the agents generated recommendations and draft artifacts for analyst consideration rather than opening the path to autonomous remediation or unsupervised incident closure. This pattern is crucial because it demonstrates that the system remained assistive and review-centered even when the upstream alerts were automatically ingested from Wazuh SIEM data rather than manually authored test cases.

For the medium and suspicious severity alerts, the outputs show that the pipeline completed analysis and produced recommended next steps but still ended up in a constrained review state rather than an autonomous action state. For example, one AppArmor denial alert was assigned `severity medium`, with the final output recorded as `ticketactiondraftonly`, `humanapprovalrequired false`, and `finalstate readyforreview`, meaning the system prepared analyst-facing recommendations but did not independently execute containment or remediation actions. Similar medium-severity handling appears for PAM session and AppArmor-related events, where the agent generated structured review guidance but remained inside the governance boundary imposed by the controller and guardrail outputs.

The critical and high-severity cases provide stronger evidence of governance enforcement because they explicitly show a human-in-the-loop requirement in both the risk stage and the final response stage. For example, alerts involving successful `sudo` to root, AppArmor denied operations with elevated implications, and host-based anomaly detection events were labeled `severity high` with `requireshumanapproval true`, and their corresponding response outputs also show `ticketaction draftonly`, `humanapprovalrequired true`, and `finalstate readyforreview`. This confirms that once the pipeline assessed a case as high or critical, the agent was prevented from taking further autonomous action and instead shifted into a governed recommendation mode intended for analyst review.

The audit logs also reveal an important governance characteristic—the system allowed tool use needed for analyst while still restricting downstream action authority. During enrichment, entries such as `guardrail.toolallowed for geoip`, `assetlookup`, and `threatintel` show that the agents were permitted to gather contextual evidence, but the final control point remained in the response layer, where outputs were validated and retained in a draft-only, review-required state for high and critical-risk alerts. This separation between permitted evidence collection and restricted operational action is consistent with the paper's folder-oriented governance model, which aims to answer the research questions in preserving useful agent capability without granting unconstrained execution authority.

Another notable result is that the governance behavior remained stable even when the original Wazuh alerts carried relatively low native severity values. Multiple records began as low-risk Wazuh inputs with `rulelevel 3`, yet the agentic workflow elevated some of them to `medium` or `high` after contextual analysis, while still enforcing constrained outputs such as `draftonly` and `readyforreview`. This matters because it shows that the governed system was not merely echoing SIEM severity labels. Instead, it performed secondary reasoning and escalation while keeping the final authority boundary under policy control.

The results also highlight the value of structured audit logging for traceability and post hoc verification. The logs record a detailed sequence of events, including `controller.start`, `guardrail.inputvalidated`, `agent.enrichment.complete`, `agent.analysis.complete`, `agent.risk.complete`, `agent.response.complete`, and `controller.complete`, which makes it possible to reconstruct exactly how a given alert was processed and where governance checks were applied. In practical terms, this means an analyst or auditor can verify not only the final recommendation but also whether the system followed the required governance path before producing that recommendation.

The output examples further suggest that the main effect of governed design is not to suppress agent reasoning, but to channel it into constrained, accountable outcomes. The agents still produced substantive analytical summaries, confidence assessments, severity rationales, and recommended next steps for SOC analysts—indicating that governance did not eliminate usefulness. Instead, governance bounded the endpoint of that usefulness by ensuring that higher-risk cases, and even medium cases in practice, ended as review artifacts rather than autonomous operational changes.

These findings strengthen the paper's argument that folder-oriented governance functions as an operational control pattern under automated alert ingestion, not only as a proof-of-concept for manual scenarios. The shift from hand-crafted alerts to SIEM-driven inputs could have exposed weaknesses in escalation consistency or runtime control, yet the logs show the opposite—the governance path remained intact, and risky cases were still prevented from moving into unsupervised action states. This supports the claim that the governance structure is portable from limited demonstrations to more realistic SOC-style workflows.

At the same time, the outputs suggest an important tradeoff that should be acknowledged in the discussion. The system appears intentionally conservative, because even medium-severity alerts were retained in draftonly and readyforreview states rather than advancing to any autonomous enforcement action. That conservatism is desirable for safety and accountability in early-stage enterprise deployment, but it also implies that future work may need to define narrower categories of low-risk, preapproved actions if organizations want to increase automation without weakening governance guarantee.

Overall, the observed results support three central claims. First, automated SIEM-driven alert ingestion did not break the governance model, because alerts still passed through explicit validation and control checkpoints before any recommendation was produced. Second, the agents were prevented from taking autonomous action for medium and high alerts, as shown by repeated `ticketaction draftonly`, `finalstate readyforreview`, and, for high-critical alerts, `humanapprovalrequired true` outputs. Third, the architecture preserved useful analytical assistance while maintaining bounded autonomy, which is the core practical contribution claimed in this paper.

Taken together, these findings address the study's research questions and hypotheses. For RQ1/H1, the governed pipeline consistently blocked disallowed actions under automated SIEM ingestion while preserving permitted analysis and draft-response behavior. For RQ2/H2 and RQ2/H3, higher-risk alerts were more frequently routed to human review, and all evaluated alerts produced complete audit traces according to the definition in the methodology section. For RQ3/H4, the end-to-end latency measurements show that governance introduced additional overhead but remained within a range that is operationally acceptable for SOC triage in this proof-of-concept experiment. In this sense, the results demonstrate that folder-oriented governance can enforce bounded autonomy and traceable decision paths under automated alert ingestion, while still providing useful analytical assistance to human analysts.

5. CONCLUSION & FUTURE WORK

This paper extends a prior folder-oriented governance prototype for agentic AI by replacing manual SOC alert inputs with automated SIEM-driven alert ingestion and evaluating the design under broader operational conditions. The controlled proof-of-concept design compared a governed agent against an unconstrained baseline using the same alert stream, with the governed workflow enforcing input validation, context enrichment, alert analysis, risk assessment, response drafting, and audit logging through a fixed orchestrator-controlled sequence. The evaluation was designed to assess policy effectiveness, escalation behavior, auditability, and runtime overhead under a broader alert set organized across low, medium or suspicious, and high or critical conditions.

This evaluation is intentionally limited in several respects. The prototype was assessed in a single Wazuh-based SOC environment with 120 alerts drawn from one observed inventory, and it relies on a specific orchestration, policy, and model configuration. The study does not measure long-term robustness, false-positive or false-negative rates, or interaction with existing ticketing and incident-management systems. As a result, the findings should be interpreted as evidence of feasibility and behavior in this controlled setting rather than as a claim of full production readiness. These limitations motivate the future work outlined below, which aims to extend the governance pattern to broader deployments and more diverse operational conditions.

The results indicate that the governed multi-agent workflow preserved bounded autonomy while maintaining structured auditability across the full decision path. The governed pipeline blocked 90 disallowed actions for medium-suspicious, high and critical-severity alerts, preserved complete audit traces, and consistently routed all the 75% of alerts into a draft-only state for human-SOC analyst review and approval, while introducing only moderate latency overhead. The audit logs further showed that the system permitted contextual evidence gathering through approved tools—including the MITRE-ATT&CK mapping and geoip, while restricting downstream operational action authority, and that even lower-severity inputs could be escalated through contextual analysis without weakening the governance boundary.

Within the scope of this proof-of-concept study, the findings indicate that folder-oriented governance remains effective for governing automated agentic AI workflows over SIEM-driven alerts. In particular, the evaluated prototype provides analytical assistance and structured recommendations while constraining autonomous and ungoverned execution under the policies tested in this study. Future work should evaluate multi-agent coordination, policy-catalog services, CI/CD policy deployment, and deeper enterprise integration.

6. ACKNOWLEDGE & AI TOOLS USED

The OpenAI API was used in the multi-agent simulation and for limited code debugging. Microsoft Copilot in Word assisted with grammar and organization. The authors designed the methodology, conducted the experiments, analyzed the results, and accept responsibility for the manuscript's accuracy, originality, integrity, and ethical implications.

7. REFERENCES

- [1] G. Elongha, "Folder-oriented governance for agentic AI: A policy-as-code pattern for secure enterprise agents," *Issues in Information Systems*, Vol. 27, no. 1, 2026. Available: doi:10.48009/1_iis_2026_01-14.
- [2] E. Tabassi, "Artificial Intelligence Risk Management Framework (AI RMF 1.0)", NIST AI 100-1. Gaithersburg, MD, USA: National Institute of Standards and Technology, Jan. 2023. Available: <https://doi.org/10.6028/NIST.AI.100-1>
- [3] Y. Shavit, S. Agarwal, M. Brundage, S. Adler, C. O'Keefe, R. Campbell, T. Lee, P. Mishkin, T. Eloundou, A. Hickey, K. Slama, L. Ahmad, P. McMillan, A. Beutel, A. Passos, and D. G. Robinson, *Practices for Governing Agentic AI Systems*. OpenAI, Dec. 2023. Available: <https://openai.com/index/practices-for-governing-agentic-ai-systems/>
- [4] C. L. Wang, T. Singhal, A. Kelkar, and J. Tuo, "MI9: An Integrated Runtime Governance Framework for Agentic AI," *arXiv preprint arXiv:2508.03858*, Aug. 2025. Available: <https://arxiv.org/abs/2508.03858>
- [5] A. G. Patil, "Governing Agentic AI: A Strategic Framework for Autonomous Systems," Sep. 2025. Available: <https://doi.org/10.13140/RG.2.2.16705.21608>
- [6] M. Abou Ali, F. Dornaika, and J. Charafeddine, "Agentic AI: A comprehensive survey of architectures, applications, and future directions," *Artificial Intelligence Review*, vol. 59, art. no. 11, 2026, Available: <https://doi.org/10.1007/s10462-025-11422-4>