International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

When the Observer Acts: Threat Modeling Agentic AI in Observability Pipelines

Shiva Raju

Abstract

Observability platforms have historically been read-only consumers of telemetry: they collected, correlated, and displayed. Recent reference architectures for generative and agentic root cause analysis (RCA) break that assumption. When an initial diagnosis falls below a confidence threshold, the system invokes an agent that autonomously queries monitoring tools, reads source repositories, and executes policy-driven actions until sufficient evidence is assembled. The observer has become an actor. This paper argues that the security consequences of that shift have not been systematically analyzed, and supplies the missing analysis. Taking the four-layer GenAI-driven observability architecture of [1] as a representative system under analysis, we construct an explicit threat model: three trust zones, three trust boundaries, four adversary classes, and nineteen threats enumerated with STRIDE across seven architectural elements. We then trace three end-to-end attack chains that each cross all three boundaries, the most consequential being telemetry-borne instruction injection, in which an adversary who can influence a single application log line reaches the production control plane without ever authenticating to the observability platform. Two structural properties make agentic observability distinctively exposed: telemetry is attacker-influenced input that has never been treated as such, and the diagnostic agent necessarily holds broad standing read privilege across the estate. We propose twelve design-level controls organized into four families — provenance, corpus integrity, least authority, and accountability — and identify the residual risks that no current control fully addresses. This work is analytical in scope. It contributes no implementation and reports no experiments; its purpose is to establish the threat vocabulary and design constraints that empirical work in this area will need.

Keywords: agentic AI, observability, AIOps, root cause analysis, threat modeling, prompt injection, LLM security, retrieval-augmented generation, least privilege, autonomous remediation.

1. Introduction

For three decades the security posture of monitoring infrastructure rested on a simple and largely correct assumption: the monitoring system reads, and does not write. A compromised dashboard was a confidentiality problem and an availability problem, but it was rarely an integrity problem for the systems being monitored. Telemetry flowed inward; nothing flowed back. Site reliability practice was built on this separation [2], [3], and the observability tooling that matured through the 2010s inherited it as an unexamined default [5].

Generative and agentic RCA architectures dissolve that separation. The reference design proposed in [1] is explicit about it: a four-layer pipeline of ingestion, normalization, multimodal fusion, and a generative RCA engine, augmented by an agentic module that activates when the initial diagnosis is insufficiently confident. That module is described as autonomously querying monitoring tools, analyzing source code repositories, fetching external context, and executing policy-driven actions [1]. The same paper reports

that this agentic path was triggered in the substantial majority of low-confidence scenarios during its proof-of-concept evaluation. Whatever the merits of the diagnostic performance, the architectural consequence is unambiguous: the observability system now holds credentials to, and takes actions against, the estate it observes.

This is not a criticism of [1] in particular. The pattern is convergent across the field. Agentic loops built on planning-and-tool-use paradigms [7], [8] and standardized tool interfaces [9] are being attached to operational data planes throughout the industry, and reference architectures of this shape are proliferating faster than the security analysis of them. The gap this paper addresses is that the architectures describe the agentic capability as a feature and enumerate its limitations in terms of accuracy, cost, and hallucination — as [1] does thoroughly in its limitations section — while treating security as a compliance concern rather than a structural one.

Two properties make agentic observability distinctively exposed, and they compound. The first is that telemetry has never been treated as attacker-influenced input, because in a read-only pipeline it barely mattered. Log content is written by applications, and applications process adversary-supplied data; a user-controlled string in a request field reappears verbatim in a log line, is ingested, normalized, retrieved as evidence, and placed into a model prompt. The literature on indirect prompt injection establishes that content reaching a language model through a data channel is functionally indistinguishable from instruction [10], [11], [12]. Observability pipelines are, in this precise sense, an untrusted-content delivery mechanism aimed directly at a model that holds production credentials.

The second property is that diagnostic agents cannot be scoped narrowly. An RCA agent whose value proposition is correlating failures across the whole estate needs read access across the whole estate. Least privilege [22] is difficult to apply to a component whose function is breadth. The resulting principal is therefore both highly privileged and driven by untrusted input — the classic confused deputy configuration [23], instantiated at infrastructure scale.

This paper makes four contributions. First, it constructs an explicit threat model for agentic observability, defining trust zones, boundaries, assets, and adversary classes (Section 3). Second, it enumerates nineteen threats using STRIDE across seven architectural elements, with a severity rationale grounded in whether the threat terminates in autonomous action (Section 4). Third, it traces three end-to-end attack chains and identifies the specific missing control at each boundary crossing (Section 5). Fourth, it proposes twelve design-level controls in four families and states plainly what residual risk survives them (Sections 7 and 8).

Scope. This is an analytical paper. It presents no implementation, no prototype, and no experimental evaluation, and it makes no quantitative claims about attack success rates or control efficacy. Every threat below is derived by structural analysis of a published architecture and by transfer from established results on prompt injection, retrieval poisoning, and agent security. Empirical validation is necessary and is identified as future work in Section 8; the purpose here is to establish the vocabulary and the design constraints such work would need.

2. Background and System Under Analysis

2.1 The reference architecture

We adopt the architecture of [1] as the system under analysis because it is published, explicit about its agentic loop, and representative of the broader class. Its four layers are: (L1) a telemetry ingestion layer collecting logs, metrics, traces, and events; (L2) an ETL and normalization layer performing schema

alignment, temporal normalization, and semantic enrichment; (L3) a multimodal context fusion layer applying temporal alignment, cross-modal attention, and causal inference over the fused signal; and (L4) a generative RCA engine combining a large language model with retrieval-augmented generation [6] over a vector knowledge base of historical incidents. When the engine's confidence falls below threshold, the agentic module is invoked to gather further context [1].

We stress that nothing in our analysis depends on the specific model, vector store, or orchestration framework named in [1]; the threats follow from the topology, not the components.

2.2 What changes when the observer acts

Three changes matter. First, the direction of causality reverses: telemetry now influences state, where previously it only described state. Second, the latency budget of incident response collapses the opportunity for human review — the entire premise of automated RCA is that it acts faster than an on-call engineer, which means the authorization step that would ordinarily gate a production change is the step most likely to be removed. Third, the system's own confidence estimate becomes a security-relevant control input: in [1], low confidence is precisely the trigger for escalation to the agentic path, which means an adversary who can induce low confidence can summon the more privileged component at will. Confidence, designed as a quality signal, doubles as an attacker-reachable escalation trigger.

2.3 Scope and non-goals

- In scope: the pipeline from telemetry emission through agent action, including the RAG corpus and the tool plane.
- Out of scope: model training and fine-tuning supply chain, which is well covered elsewhere [16]; conventional network and host security of the observability platform itself; and multi-tenant SaaS isolation, which deserves separate treatment.
- Non-goals: this paper does not rank threats by likelihood, does not estimate exploitation cost, and does not claim completeness of the enumeration. STRIDE is used as a structuring device, not as a guarantee of coverage [24], [25].

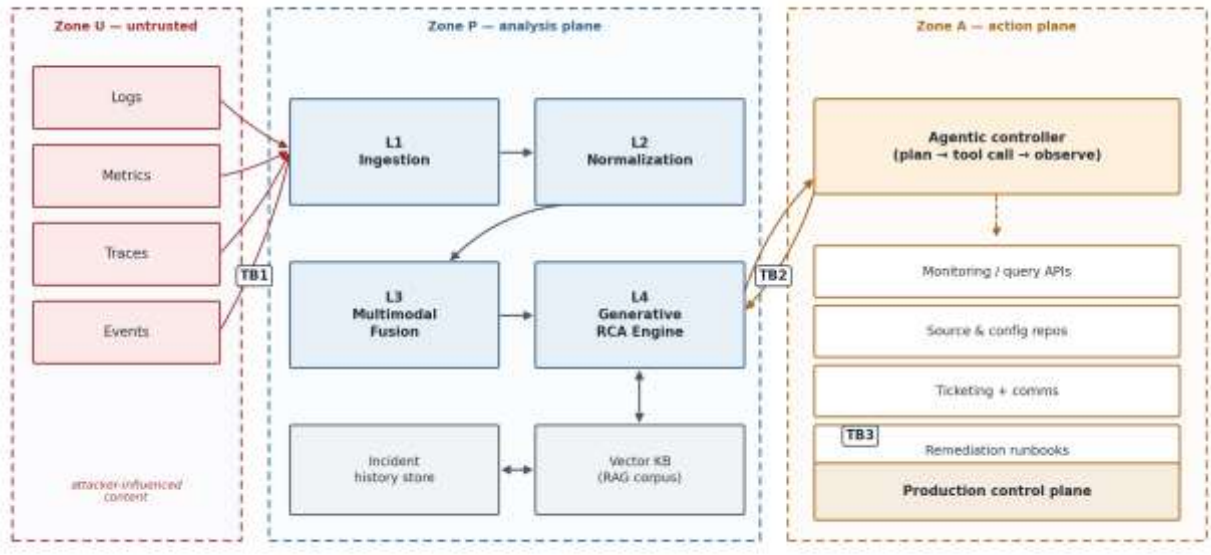

Fig. 1. Trust zones and boundaries of an agentic observability pipeline. Telemetry originates in the untrusted zone U; the analysis plane P performs ingestion through generative RCA; the action plane A

holds the agentic controller and its tools. TB1, TB2, and TB3 mark the three boundaries where authority changes. Architecture adapted from [1].

3. Threat Model

3.1 Assets

We identify five assets whose compromise carries distinct consequence. A1, the integrity of the RCA verdict, because downstream automation and human decisions are conditioned on it. A2, the confidentiality of aggregated telemetry, which in a mature deployment constitutes a comprehensive map of the estate — topology, dependencies, credentials leaked into logs, and failure history. A3, the integrity of the RAG corpus, which is durable state and therefore the vector for persistence. A4, the agent's credential set and its effective authority over the tool plane. A5, the availability of the RCA path itself during an incident, when it is by definition most needed.

3.2 Trust zones and boundaries

Fig. 1 partitions the architecture into three zones. Zone U comprises telemetry sources: application logs, metrics, traces, and events. Content in this zone is attacker-influenced by default, because applications process adversary-supplied input and emit it into logs. Zone P is the analysis plane, layers L1 through L4 plus the vector knowledge base and incident history. Zone A is the action plane: the agentic controller, its tools, and ultimately the production control plane.

Three boundaries follow. TB1 separates U from P and is the point at which unauthenticated, unattributed content enters a trusted pipeline. TB2 separates the analysis plane from the action plane and is where generated text becomes an executable plan. TB3 separates the agent's tool surface from the production control plane and is where a tool call becomes a state change. The central observation of this paper is that in the architectures as published, none of these three boundaries carries a meaningful integrity check: TB1 has no provenance, TB2 has no data-versus-instruction separation, and TB3 has no action-time re-authorization.

3.3 Adversary classes

We distinguish four adversaries by their initial position, since capability follows position.

AD1 — External application-tier adversary. Cannot authenticate to the observability platform and has no account on it. Can influence the content of log records by supplying data to a public-facing application: a crafted user-agent string, a form field, a filename, a DNS query. This is the weakest adversary and, we argue, the most important, because the entire exposure follows from the fact that this position is sufficient to reach TB2.

AD2 — Low-privilege internal adversary. Holds read access to the observability platform, as most engineers in an organization do. Can observe RCA narratives, infer the prompt structure and retrieval behavior, and in many deployments contribute to runbooks or postmortems that feed the corpus.

AD3 — Corpus contributor. Can write to the knowledge base, typically through the ordinary and desirable workflow of documenting incidents. Requires no exploit; the write path is a feature.

AD4 — Compromised tool or upstream dependency. Controls a monitoring API, a source repository, or a documentation service that the agent queries. Returns adversarial content in tool responses [13], [19].

3.4 Assumptions

- The language model has no reliable mechanism to distinguish instruction from data within its context window; this is treated as a standing property, not a defect of a particular model [10], [12].

- Agent credentials are long-lived and broadly scoped, matching deployed practice and the architecture as described.
- Cryptographic primitives, the network, and the underlying hosts are sound; we analyze the architecture, not its substrate.
- The organization has legitimate, non-malicious operators. Insider threat is modeled only as AD2 and AD3 acting through sanctioned interfaces.

4. Threat Enumeration

Fig. 2 maps nineteen threats across seven architectural elements using STRIDE [24], [25]. We assign severity by a single criterion: whether the threat can terminate in autonomous action against production. Threats that can are rated high regardless of the sophistication required, because the blast radius is bounded only by the agent's privilege. Threats that degrade RCA integrity without reaching the action plane are rated medium; conventional threats already addressed by standard platform security are rated low and are included for completeness rather than analyzed at length.

Fig. 2. STRIDE enumeration across the seven architectural elements. Severity reflects whether the threat can terminate in autonomous action against production rather than the sophistication required to mount it.

4.1 Ingestion and normalization (T1–T7)

T1 (spoofing) covers forged telemetry from an unauthenticated emitter, a long-standing weakness of collection protocols that acquires new significance when the forged record can steer a diagnosis. T2 (tampering) is the central threat of this paper: injection of adversary-controlled semantic content into log records, discussed at length in Section 5.1. T3 is volumetric denial of the ingestion path, historically an availability concern and now additionally a way to induce the low-confidence state that escalates to the agentic path.

T4 concerns normalization itself. Layer L2 aligns schemas, parses unstructured logs, and adapts to schema drift automatically [1]. Any component that infers structure from adversary-influenced content can be manipulated into mis parsing — attributing a field to the wrong service, or splitting a record so that injected content is promoted into a field the downstream prompt treats as authoritative. T5 is disclosure through

normalization side effects, principally the enrichment step, which by design attaches business context and topology to records and can therefore widen the sensitivity of a record that was previously innocuous.

T6 addresses the fusion layer. Cross-modal attention and causal inference over fused signals [1] create a manipulation surface that has no analogue in rule-based correlation: an adversary who can emit correlated signals across two modalities can synthesize a spurious causal edge, and the causal graph is precisely what the RCA engine treats as ground truth. T7 is inference of estate structure from fusion output, a disclosure threat that matters mainly as reconnaissance for the chains in Section 5.

4.2 RCA engine and corpus (T8–T12)

T8 (tampering with the generative verdict) is the realization of T2 at the model boundary and is rated high. T9 is repudiation: when a diagnosis is generated from a fused, retrieved, non-deterministic context, reconstructing after the fact why the system concluded what it concluded is often infeasible, which undermines both incident review and any attempt to detect that manipulation occurred. T10 is disclosure through the RCA narrative itself, which is designed to be evidence-backed and therefore quotes telemetry — including, routinely, secrets that were logged in error.

T11 is corpus poisoning, rated high. Retrieval corpora are known to be manipulable with a small number of injected passages [14], [15], and unlike prompt-level injection, corpus poisoning persists: it survives the incident, the session, and the model upgrade. In an observability deployment the write path is a documentation workflow, so AD3 requires no exploit at all. T12 is extraction of corpus contents through crafted incidents, an inversion of the retrieval mechanism to read historical material the adversary is not authorized to see.

4.3 Agentic controller and tool plane (T13–T19)

T13 is impersonation of the controller to its tools or of a tool to the controller, a conventional authentication problem made harder by the dynamism of the agent's tool selection. T14 is manipulation of the agent's plan through injected content in tool responses [13], the mechanism by which AD4 operates and the reason that an agent which reads a compromised repository is compromised. T15 is repudiation of agent actions, which is severe in proportion to how thin the audit trail is: an action log that records the tool call but not the evidence that motivated it cannot distinguish a correct action from a manipulated one.

T16 is elevation of privilege at the controller, where an adversary who reaches the planning loop inherits the agent's full standing authority. T17 is the agent as an exfiltration channel, developed in Section 5.3. T18 is exhaustion of tool quotas and rate limits, which degrades the response capability during an incident. T19 is the terminal threat: unauthorized state change in production, reached through any of the preceding paths, and the reason the whole enumeration matters.

5. Attack Chains

Individual threats understate the exposure because the architecture composes them. We trace three chains, each crossing all three trust boundaries. The chains are constructed analytically; we have not attempted them against a deployed system, and we make no claim about their empirical success rate.

5.1 Chain A — telemetry-borne instruction injection

Fig. 3 traces the chain that we regard as the most consequential, because it begins with the weakest adversary. AD1 supplies a crafted string to a public-facing application — a field that will be logged, containing text shaped as an operational directive. The string is emitted into an application log in the ordinary course of business. L1 collects it as telemetry with no provenance label distinguishing application-authored content from user-supplied content that the application merely echoed. L2 and L3

normalize and fuse it; both layers preserve semantics by design, since preserving semantics is their function. At L4, the record is retrieved as relevant evidence for an active incident and placed into the prompt as context. The model, which cannot separate the retrieved data from its own instructions [10], [11], [12], incorporates the directive into its reasoning chain and emits a remediation plan reflecting it. The agentic controller executes that plan with its standing privilege.

What makes this chain notable is not its sophistication but its prerequisites. The adversary needs no account, no network position inside the perimeter, and no knowledge of the observability platform beyond the inference that one exists. The three failures that permit it are marked beneath Fig. 3: no authentication of telemetry content at TB1, no data-versus-instruction separation at TB2, and no action-time re-authorization at TB3. Each is individually addressable; none is addressed in the architectures as published.

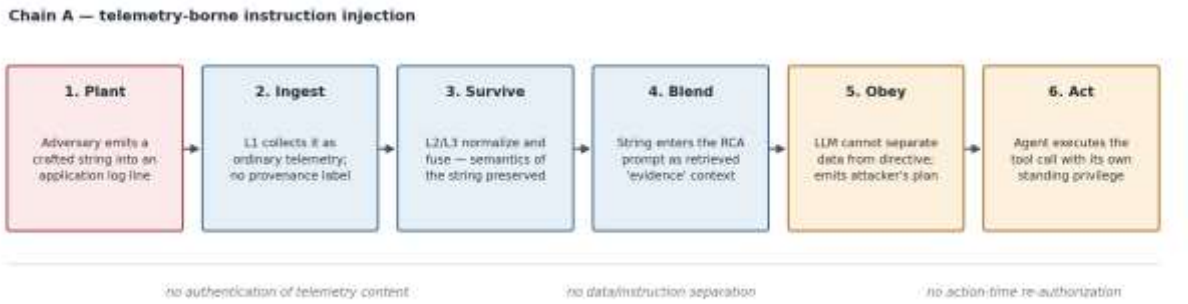

Fig. 3. Chain A. An adversary with no access to the observability platform reaches the production control plane through a single influenced log record. The three missing controls are marked beneath the corresponding boundary crossings.

5.2 Chain B — corpus poisoning for persistent misdiagnosis

AD3 contributes a plausible but incorrect incident postmortem to the knowledge base through the sanctioned documentation workflow. The document is written to be retrievable for a target class of incidents — it uses the vocabulary, service names, and symptom descriptions that future incidents of that class will surface. Nothing about the write is anomalous; documenting incidents is the behavior the system is designed to encourage.

Thereafter, every incident matching the target class retrieves the poisoned document as historical context, and the RCA engine grounds its verdict in it. The consequences are asymmetric in a way that prompt-level injection is not. Poisoning is persistent, surviving sessions and model upgrades because it lives in data rather than in a context window. It is amplifying, since the resulting incorrect verdicts are themselves recorded and may re-enter the corpus. And it is nearly invisible, because the output remains fluent, well-evidenced in form, and consistent with retrieved material — which is exactly the presentation that builds operator trust. Small-scale corpus injection has been shown effective in general retrieval settings [14], [15]; the observability case is more favorable to the adversary because the corpus is small, domain-specific, and openly writable.

5.3 Chain C — the agent as exfiltration channel

The agent holds broad read access by construction and possesses outbound capability through its tool plane: it files tickets, posts to chat, queries external documentation, and writes to repositories. An adversary who influences the agent's plan — by Chain A, by Chain B, or by AD4 returning adversarial content from a compromised tool [13], [19] — can direct it to read sensitive material within its authority and emit that material through a legitimate outbound tool.

Conventional data-loss controls perform poorly here. The agent is authorized for every read it performs. Its outbound calls are to sanctioned internal destinations. The volume is small, matching normal diagnostic behavior. And the activity occurs during an incident, when anomalous access patterns are expected and alerting thresholds are often deliberately relaxed. This chain is the clearest illustration of why breadth of read privilege, not depth of write privilege, is the harder problem in agentic observability.

6. Cross-Cutting Analysis

6.1 The privilege paradox

Least privilege [22] resists application here for a structural reason. The diagnostic value of the agent is proportional to the breadth of what it can read: an agent restricted to one service cannot diagnose a cascading failure across twelve. Narrowing read scope therefore degrades the capability that justifies the system. We see no clean resolution, only a reframing: the scoping decision should apply to the write and action surface, which can be narrowed sharply without loss of diagnostic power, rather than to the read surface. The read/act split proposed as M8 in Section 7 follows from this.

6.2 Confidence as an attacker-reachable trigger

The escalation design in [1] activates the agentic module when confidence is low. Any adversary who can degrade confidence — by emitting sparse, contradictory, or noisy telemetry, all of which are ordinary consequences of a real failure and therefore unremarkable — can summon the privileged component. Designing escalation to be triggered by adversary-reachable conditions inverts the intended relationship between uncertainty and caution: the system responds to uncertainty by increasing its own authority. A defensible design would make low confidence escalate to a human rather than to a more capable agent.

6.3 Auditability and the evidence binding problem

Distinguishing a correct autonomous action from a manipulated one after the fact requires an audit record that binds the action to the specific evidence and retrieval set that motivated it. Logs that record the tool call alone are insufficient; logs that record the prompt but not the retrieval provenance are insufficient. This is the concrete form of T9 and T15, and it is a design requirement rather than an operational one, because the binding must be created at generation time and cannot be reconstructed later.

6.4 Relationship to failure taxonomy

Recent work characterizing failure modes of multi-agent LLM systems [21] catalogues failures arising from specification, inter-agent misalignment, and verification gaps. The threats here are the adversarial complement: where that taxonomy asks how agents fail on their own, we ask how an adversary induces those failures deliberately. Several categories map directly — a verification gap that produces an accidental bad action is the same gap an adversary exploits to produce a chosen one — and we suggest that adversarial and accidental failure analysis for agentic systems are more usefully pursued together than separately. The hallucination literature [18] occupies a similar position: hallucination is the untargeted case of the integrity loss that Chain B targets.

7. Design-Level Mitigations

Fig. 4 organizes twelve controls into four families aligned to the boundaries of Fig. 1. These are design constraints, not implementations; we deliberately do not specify mechanisms, and we note where a control is only partially effective. None of these controls is individually novel — most are conventional security engineering — and that is the point: the exposure arises from omitting well-understood controls at boundaries the field has not yet recognized as boundaries.

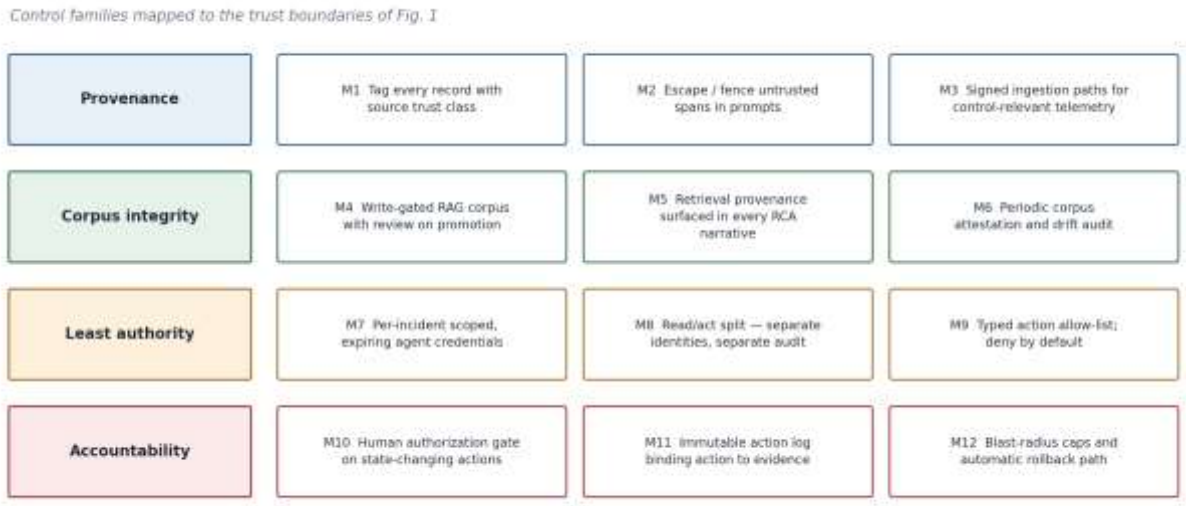

Fig. 4. Twelve design-level controls in four families, aligned to the trust boundaries of Fig. 1. Provenance and corpus integrity address TB1; least authority and accountability address TB2 and TB3.

7.1 Provenance (TB1)

M1 requires that every telemetry record carry a trust class assigned at emission, distinguishing platform-authored content from application-authored content from content the application echoed from an external party. This is an instrumentation-side change and therefore the hardest to adopt, but without it every downstream control is guessing. M2 requires that content of untrusted class be structurally fenced when placed into a prompt, and that the model be instructed to treat fenced spans as data. We stress that fencing is a mitigation of uncertain strength: it raises the cost of injection without eliminating it [12], and it should not be relied upon as a boundary. M3 applies signed ingestion paths to the subset of telemetry that is control-relevant, accepting that this cannot cover general application logs.

7.2 Corpus integrity (A3)

M4 gates writes to the retrieval corpus behind review, treating corpus promotion as a change to production configuration rather than as documentation. M5 requires that retrieval provenance be surfaced in every RCA narrative, so that an operator sees which documents grounded the verdict — this is the minimum condition under which Chain B is detectable by a human reader. M6 adds periodic attestation and drift audit of the corpus, on the reasoning that persistent poisoning is only discoverable by looking for it deliberately.

7.3 Least authority (TB2, TB3)

M7 replaces standing agent credentials with per-incident, scoped, expiring credentials, bounding the window in which a compromised plan can act. M8 separates the read identity from the act identity, with distinct credentials and distinct audit streams; this is the operational form of the resolution proposed in Section 6.1 and, in our assessment, the single highest-value control in the set. M9 constrains the agent to a typed allow-list of actions with deny-by-default semantics, which converts an open-ended action surface into an enumerable one that can be reviewed.

7.4 Accountability (TB3)

M10 places a human authorization gate on state-changing actions. This is in direct tension with the latency

argument for automation, and that tension should be resolved explicitly per action class rather than implicitly by omitting the gate. M11 requires an immutable action log binding each action to the evidence and retrieval set that produced it, addressing Section 6.3. M12 imposes blast-radius caps and a guaranteed rollback path, on the assumption that some manipulated actions will execute and that recoverability is the last line of defense. Frameworks for agent risk assessment [19], [20] and the general governance guidance in [26], [27], [28] provide useful structure for deciding which action classes warrant which controls.

8. Residual Risk and Open Problems

We state plainly what the controls above do not solve. First, no known mechanism reliably separates instruction from data inside a language model's context [10], [12]; M2 raises cost without closing the channel, so TB2 remains a probabilistic boundary rather than an enforced one. Second, provenance tagging at emission (M1) requires changes to instrumentation across every application in the estate, which is realistic for greenfield deployments and largely unrealistic for the legacy footprint where observability tooling is most valuable. Third, human authorization gates (M10) degrade under alert fatigue in precisely the high-volume incident conditions where manipulation is most likely, and we are not aware of evidence that operators reviewing model-generated remediation plans at 3 a.m. detect subtly manipulated ones.

Four problems seem to us the most tractable next steps, and all of them require the empirical work that this paper does not attempt. A benchmark for telemetry-borne injection, constructed from realistic log corpora rather than synthetic prompts, would let defenses be compared; the general-purpose agent security benchmarks [20] do not capture the observability data path. A measurement study of how much operator review actually catches would settle whether M10 is a control or a formality. Formal characterization of the read/act split (M8) would clarify how much diagnostic capability is genuinely lost when action scope is narrowed. And integration of adversarial threat modeling with accidental failure taxonomy [21] would give the field a single account of how agentic operational systems go wrong.

9. Related Work

Observability and AIOps. The architectural line this paper analyzes runs from full-stack observability practice [2], [3], [5] through AIOps platforms [4] to generative and agentic RCA [1]. These works treat security as a deployment consideration — [1] notes regulatory and compliance constraints among its limitations — but do not model the agent as an attack surface, which is the gap addressed here.

Prompt injection and agent security. Indirect prompt injection was characterized in [10] and formalized and benchmarked in [12]; the underlying instruction-following weakness was demonstrated earlier in [11] and popularized in [30]. Agent-specific tool-mediated injection is studied in [13], sandboxed risk identification in [19], end-to-end agent benchmarks in [20], and safe in-the-wild agent testing in [29]. Our contribution is to instantiate these results in a domain where the untrusted channel is telemetry and the privileged principal is diagnostic infrastructure.

Retrieval poisoning. Corpus-level attacks on retrieval systems [14], [15] and broader data poisoning results [16], [17] establish the feasibility that Chain B relies upon. The observability setting differs in that the corpus is small, domain-specific, and writable through a sanctioned workflow, which we argue makes it more favorable to the adversary than the web-scale settings those works consider.

Threat modeling methodology. We use STRIDE [24] in the form described by [25], with trust boundaries drawn per the classical guidance on protection mechanisms [22] and the confused deputy framing of [23].

Practitioner catalogues [26], [27] and the risk management framework in [28] informed the control families but are not sufficient on their own, since none addresses telemetry as an injection channel.

10. Conclusion

Agentic observability inverts an assumption that made monitoring infrastructure comparatively safe to deploy: that the system reads and does not write. Once an RCA engine is granted the ability to query, fetch, and remediate, telemetry becomes an input channel to a privileged actor, and telemetry is written by applications that process adversary-supplied data. The result, analyzed here through the reference architecture of [1], is that an adversary with no access to the observability platform can reach the production control plane through a single influenced log line, crossing three trust boundaries none of which carries an integrity check.

We have enumerated nineteen threats, traced three attack chains, and proposed twelve design-level controls, while being explicit that the strongest of them — data/instruction separation at the model boundary — remains unsolved and that the whole analysis awaits empirical validation. The narrower claim we would defend is this: the agentic invocation path deserves the same architectural scrutiny that the diagnostic pipeline has received, and it is currently receiving considerably less. Reference architectures in this area should specify their trust boundaries as carefully as they specify their layers.

References

1. Bisht, K. S. Generative AI-Driven Observability for Automated Root Cause Analysis in Modern IT Systems: Architecture and Vision. Journal of Computer Science and Technology Studies, 7(9), 549-560. https://doi.org/10.32996/jcsts.2025.7.9.63
2. B. Beyer, C. Jones, J. Petoff, and N. R. Murphy, Site Reliability Engineering: How Google Runs Production Systems. Sebastopol, CA, USA: O'Reilly Media, 2016.
3. C. Majors, L. Fong-Jones, and G. Miranda, Observability Engineering: Achieving Production Excellence. Sebastopol, CA, USA: O'Reilly Media, 2022.
4. Gartner, "Market guide for AIOps platforms," Gartner Research, 2023.
5. OpenTelemetry Authors, "Observability primer," Cloud Native Computing Foundation. Accessed: Aug. 12, 2026. [Online]. Available: https://opentelemetry.io/docs/concepts/observability-primer/
6. P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in Proc. 34th Conf. Neural Information Processing Systems (NeurIPS), 2020, pp. 9459–9474.
7. S. Yao et al., "ReAct: Synergizing reasoning and acting in language models," in Proc. Int. Conf. Learning Representations (ICLR), 2023.
8. T. Schick et al., "Toolformer: Language models can teach themselves to use tools," in Proc. Neural Information Processing Systems (NeurIPS), 2023.
9. Anthropic, "Model Context Protocol specification," 2024. Accessed: Aug. 12, 2026. [Online]. Available: https://modelcontextprotocol.io
10. K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz, "Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection," in Proc. 16th ACM Workshop on Artificial Intelligence and Security (AISec), 2023, pp. 79–90.
11. F. Perez and I. Ribeiro, "Ignore previous prompt: Attack techniques for language models," in NeurIPS ML Safety Workshop, 2022.

12. Y. Liu et al., "Formalizing and benchmarking prompt injection attacks and defenses," in Proc. 33rd USENIX Security Symposium, 2024.
13. Q. Zhan, Z. Liang, Z. Ying, and D. Kang, "InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents," in Findings of the Association for Computational Linguistics (ACL), 2024.
14. Z. Zhong, Z. Huang, A. Wettig, and D. Chen, "Poisoning retrieval corpora by injecting adversarial passages," in Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP), 2023.
15. W. Zou, R. Geng, B. Wang, and J. Jia, "PoisonedRAG: Knowledge corruption attacks to retrieval-augmented generation of large language models," in Proc. USENIX Security Symposium, 2025.
16. N. Carlini et al., "Poisoning web-scale training datasets is practical," in Proc. IEEE Symposium on Security and Privacy (S&P), 2024.
17. A. Zou, Z. Wang, N. Carlini, M. Nasr, J. Z. Kolter, and M. Fredrikson, "Universal and transferable adversarial attacks on aligned language models," arXiv:2307.15043, 2023.
18. Z. Ji et al., "Survey of hallucination in natural language generation," ACM Computing Surveys, vol. 55, no. 12, pp. 1–38, 2023.
19. Y. Ruan et al., "Identifying the risks of LM agents with an LM-emulated sandbox," in Proc. Int. Conf. Learning Representations (ICLR), 2024.
20. E. Debenedetti et al., "AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents," in Proc. NeurIPS Datasets and Benchmarks Track, 2024.
21. M. Cemri et al., "Why do multi-agent LLM systems fail?" arXiv:2503.13657, 2025.
22. 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.
23. N. Hardy, "The confused deputy: (or why capabilities might have been invented)," ACM SIGOPS Operating Systems Review, vol. 22, no. 4, pp. 36–38, 1988.
24. L. Kohnfelder and P. Garg, "The threats to our products," Microsoft Interface, 1999.
25. A. Shostack, Threat Modeling: Designing for Security. Indianapolis, IN, USA: Wiley, 2014.
26. OWASP Foundation, "OWASP Top 10 for Large Language Model Applications," 2025. Accessed: Aug. 12, 2026. [Online]. Available: https://owasp.org/www-project-top-10-for-large-language-model-applications/
27. MITRE, "ATLAS: Adversarial Threat Landscape for Artificial-Intelligence Systems." Accessed: Aug. 12, 2026. [Online]. Available: https://atlas.mitre.org
28. National Institute of Standards and Technology, "Artificial Intelligence Risk Management Framework (AI RMF 1.0)," NIST AI 100-1, 2023.
29. S. Naihin et al., "Testing language model agents safely in the wild," arXiv:2311.10538, 2023.
30. S. Willison, "Prompt injection attacks against GPT-3," 2022. Accessed: Aug. 12, 2026. [Online]. Available: https://simonwillison.net/2022/Sep/12/prompt-injection/