COMMENT OF PANOPTIC SYSTEMS

In Response to NIST Request for Information

Regarding Security Considerations for Artificial Intelligence Agents

Docket No. NIST-2025-0035 | 91 FR 698

February 2026

About the Commenter

Panoptic Systems is building runtime governance infrastructure for AI agent systems. This comment draws on direct experience designing and testing proxy-based enforcement, policy-as-code evaluation, and cryptographically signed audit records for agent tool invocations, including emerging protocols such as the Model Context Protocol (MCP).

This submission addresses the priority questions identified in the RFI: 1(a), 1(d), 2(a), 2(e), 3(a), 3(b), 4(a), 4(b), and 4(d). Responses are grounded in practical implementation experience rather than theoretical frameworks.

1. Security Threats, Risks, and Vulnerabilities

Question 1(a): What are the unique security threats, risks, or vulnerabilities currently affecting AI agent systems, distinct from those affecting traditional software systems?

The most consequential security gap in deployed AI agent systems today is not prompt injection, model manipulation, or adversarial attacks on the model itself. It is the absence of a verifiable, tamper-evident record of what an agent actually did at runtime.

Traditional software systems produce deterministic outputs for given inputs. When a traditional application takes an action, the path from input to output can be reconstructed from source code and logs. AI agent systems break this assumption fundamentally. An agent receiving identical instructions on two consecutive runs may select different tools, pass different parameters, and produce different outcomes. The non-deterministic nature of the underlying model means that the action taken is not predictable from the instruction given.

This creates a category of risk that has no direct analog in traditional software security: the evidence gap. Specifically:

Unverifiable agent behavior. Most deployed agent systems log actions in application-level logs that are mutable, unstructured, and lack cryptographic integrity guarantees. An organization cannot prove to an auditor, regulator, or court what an agent did, what policy was active at the time, who authorized the

action, or whether the log has been altered after the fact. California's AB 316, effective January 1, 2026, forecloses the defense that an AI system's autonomous operation absolves the deployer of liability. Yet most organizations deploying agents today cannot produce the evidentiary artifacts needed to demonstrate governance was in place at the time of an incident.

Action-level versus prompt-level risk. The predominant approach to agent security today focuses on filtering or classifying the prompt before it reaches the model, or filtering the model's response before it reaches the user. This addresses a subset of risks but misses the critical control point: the action the agent takes in the external environment. An agent may receive a well-formed, policy-compliant prompt, generate a reasonable response, and then execute a tool call that deletes production data, transfers funds, or exfiltrates sensitive information. Prompt-level controls do not intercept these actions because the action occurs downstream of the model's output, at the interface between the agent and its tools.

Bypass-by-architecture. Many current agent governance approaches are implemented as middleware within the agent's own runtime. This means a sufficiently capable agent, a compromised dependency, or a malicious actor with access to the agent's process can bypass governance entirely. The governance mechanism and the entity being governed share the same trust boundary. This is analogous to relying on an application to enforce its own access controls without an external authentication service, a pattern that the security community abandoned decades ago.

Question 1(d): How have these threats, risks, or vulnerabilities changed over time? How are they likely to evolve in the future?

The trajectory is toward increasing agent autonomy, longer-running unsupervised tasks, and agents that interact with other agents across organizational boundaries. Each step along this trajectory amplifies the evidence gap described above.

In 2024, most deployed agents operated in single-turn or short-horizon patterns: a user issued an instruction, the agent performed a task, and a human reviewed the result. The governance challenge was manageable because the human was in the loop for each action cycle.

By early 2026, agents are increasingly deployed in longer-horizon configurations: background workflows that monitor data sources, multi-step processes that span hours or days, and agent-to-agent interactions where one agent delegates tasks to another via protocols like the Agent-to-Agent Protocol (A2A). In these configurations, the human is no longer reviewing each action. The governance system must operate continuously and autonomously, which means it must be architecturally independent of the agent it governs.

The most significant near-term evolution is the transition from agents that act on behalf of a human user (inheriting the user's identity and permissions via mechanisms like OAuth on-behalf-of) to agents that possess their own autonomous identity. Frameworks like SPIFFE (Secure Production Identity Framework for

Everyone) are being explored for this purpose. When agents have their own identity, the assumption that an agent's actions are bounded by a human's permissions no longer holds. Governance must then operate at the agent identity level, not the user identity level. This transition is already visible in emerging agent ecosystems built around standardized tool-invocation protocols such as MCP.

2. Security Practices for AI Agent Systems

Question 2(a): What technical controls, processes, and other practices could ensure or improve the security of AI agent systems? What is the maturity of these methods?

The most effective architectural pattern for agent governance is the inline proxy: a separate process, operating outside the agent's trust boundary, that intercepts every tool call and API request the agent makes before it reaches the external environment. This is not a novel concept. It is the same architectural pattern used in web application firewalls, service meshes, and API gateways. The application to AI agents, however, introduces requirements that existing proxy implementations do not address.

Based on practical implementation, the following controls are effective and technically feasible today:

Action-level policy enforcement. Policies should evaluate the specific tool call, its parameters, the target resource, and contextual metadata (agent identity, session history, time of day, cumulative risk score) to produce an allow, deny, or escalate-for-approval decision. Policies should be expressed as code, not as natural language rules interpreted by another model. Open Policy Agent (OPA) and its Rego policy language provide a mature, well-understood framework for this purpose. Policy-as-code is auditable, version-controlled, testable, and deterministic. These are properties that natural-language guardrails fundamentally lack.

Human-in-the-loop approval workflows. For consequential actions, specifically those that modify state in external systems, access sensitive data, or exceed defined risk thresholds, the governance system should be able to pause the agent's execution and route the pending action to a human approver. The approver should see the full context: what the agent is trying to do, what parameters it is passing, which policy triggered the escalation, and the session history leading to this point. The approval or denial should be cryptographically recorded as part of the action's evidence chain. This pattern is mature in traditional workflow systems (e.g., IT service management, change management) but is rarely implemented in agent governance today.

Tamper-evident audit trails. Every action an agent takes should produce a signed, hash-chained record that includes: the tool call and its parameters, the policy that was evaluated and its version, the decision (allow/deny/escalate), the response from the external system, and the identity of any human approver. These

records should be signed using envelope signatures such as Dead Simple Signing Envelope (DSSE) with keys controlled by the deploying organization, and chained via content-addressable hashing so that any modification to a historical record invalidates the chain. The resulting evidence bundle should be verifiable offline, without requiring access to the governance system that produced it. This enables independent audit by third parties.

Cryptographic agent identity. Each agent should possess a verifiable identity, bound to its governance policy, that cannot be forged or transferred. SPIFFE provides a mature framework for workload identity that can be adapted for this purpose. When an agent's identity is cryptographically bound to its policy set and recorded in every action's evidence chain, the organization can prove which agent took which action under which policy. This requirement will become critical as multi-agent deployments scale.

The maturity of these methods varies. Inline proxying is well-established in infrastructure security. OPA-based policy evaluation is production-grade and widely deployed. Hash-chained audit trails are proven in supply chain security (e.g., Sigstore, in-toto). Human-in-the-loop approval workflows are standard in ITSM. The novel contribution is integrating these existing, well-understood patterns into a coherent governance architecture specifically designed for AI agent systems. This integration is technically mature but not yet widely adopted.

Question 2(e): Which cybersecurity guidelines, frameworks, and best practices are most relevant to the security of AI agent systems?

The zero trust architecture model (NIST SP 800-207) is directly applicable and arguably the most important existing framework for agent security. AI agents should be treated as untrusted workloads regardless of where they are deployed. No agent should be trusted to govern itself, just as no application in a zero trust network is trusted to authenticate itself.

NIST SP 800-53 provides relevant controls, particularly in the AU (Audit and Accountability) and AC (Access Control) families, but these controls were designed for human-initiated actions. Agent-specific guidance is needed to address actions initiated by non-deterministic processes, where the "user" is a model whose behavior cannot be fully specified in advance.

The NIST AI Risk Management Framework (AI 100-1) provides useful structure, particularly the Govern and Measure functions. However, the framework would benefit from explicit guidance on runtime governance, meaning controls that operate continuously during agent execution, as distinct from development-time and deployment-time controls. The majority of risk management guidance today focuses on what happens before an agent is deployed. The most consequential risks materialize while the agent is running.

Supply chain security frameworks, particularly SLSA (Supply Chain Levels for Software Artifacts) and in-toto, provide useful analogies for agent governance. An agent's action is analogous to a build step in a software supply chain: it takes

inputs, applies a process, and produces outputs. The same integrity guarantees that supply chain security applies to build artifacts, including provenance, reproducibility, and tamper evidence, should be applied to agent actions.

3. Assessing the Security of AI Agent Systems

Question 3(a): What methods could be used during AI agent systems development to anticipate, identify, and assess security threats, risks, or vulnerabilities?

The most effective assessment method for agent security is adversarial testing of the governance boundary, not the model. Red-teaming exercises that focus exclusively on prompt injection and model manipulation miss the most dangerous attack vectors: those that bypass the governance mechanism entirely.

Practical assessment should include:

Bypass path analysis. Can the agent reach its tools through any path that does not traverse the governance layer? In many current architectures, the answer is yes. If the agent has direct network access to the API endpoints its tools use, it can simply call those endpoints without going through the proxy. Effective governance requires that the only network path from the agent to its tools passes through the governance layer. This can be enforced through network policies (e.g., Kubernetes NetworkPolicy, iptables rules, or cloud VPC configurations) and should be verified through penetration testing.

Policy completeness testing. Given a defined policy set, can the agent take any action that the policy was intended to prevent? This requires generating adversarial tool calls, rather than adversarial prompts, that probe the boundaries of policy rules. For example, if a policy prohibits deleting files, does it also prohibit renaming files to overwrite existing ones, or calling a “move” operation that achieves the same effect?

Evidence integrity verification. Can the audit trail be tampered with after the fact without detection? This requires testing the cryptographic integrity of the evidence chain: modifying a single record and verifying that the chain breaks, verifying signatures against known keys, and confirming that evidence bundles can be verified by an independent party with no access to the governance system.

Question 3(b): How could the security of a particular AI agent system be assessed?

Agent security assessment should evaluate three dimensions that are specific to agentic systems and not adequately covered by traditional application security assessments:

Governance independence. Is the governance mechanism architecturally independent of the agent it governs? If the governance logic runs inside the agent’s process, shares the agent’s dependencies, or can be disabled by the agent,

the governance is not independent. The degree of architectural separation between the agent and its governance layer is the single most important indicator of governance effectiveness.

Evidence sufficiency. If an incident occurs, can the organization reconstruct exactly what the agent did, what policy was in effect, who (if anyone) approved the action, and whether the record has been altered? The evidence should be sufficient to satisfy not just internal review but external audit, regulatory inquiry, and legal discovery. Organizations should test this by conducting mock incident investigations using only the artifacts their governance system produces.

Least privilege enforcement. Does the agent have access only to the tools and resources it needs for its defined task? NIST SP 800-207's zero trust principles apply directly: the agent's access should be scoped to specific tools, specific parameters within those tools, and specific time windows. The governance system should enforce these constraints externally, not rely on the agent to self-limit.

4. Limiting, Modifying, and Monitoring Deployment Environments

Question 4(a): In what manner and by what technical means could the access to or extent of an AI agent system's deployment environment be constrained?

The most effective constraint is to deploy the governance layer as an inline proxy that sits between the agent and all external interfaces, combined with network-level controls that prevent the agent from bypassing the proxy.

Concretely, this means:

Network isolation. The agent should have no direct network access to the APIs and services its tools use. Its only permitted outbound connection should be to the governance proxy. The proxy, in turn, has access to the external services. This is the same pattern used in forward proxy deployments and can be enforced using standard network security controls: firewall rules, Kubernetes NetworkPolicy, cloud security groups, or mTLS with mutual authentication where only the proxy holds valid client certificates.

Protocol-aware interception. The governance layer must understand the protocols the agent uses to invoke tools. For protocol-aware governance, the proxy must be capable of parsing structured tool invocations at the semantic level. In MCP-based systems, for example, this means interpreting tool call messages and evaluating the tool name, parameters, and context against policy. For agents that call REST APIs directly, this means inspecting HTTP requests and mapping them to policy rules. A generic network proxy is insufficient; the governance layer must be semantically aware of what the agent is doing, not just where the traffic is going.

Credential separation. The agent should not hold the credentials needed to authenticate with external services. Instead, the governance proxy should inject

credentials after the action has been approved by policy. This prevents a compromised agent from extracting credentials and using them outside the governance boundary. The proxy acts as a credential broker, similar to how a secrets manager operates in a zero trust architecture.

Question 4(b): How could virtual or physical environments be modified to mitigate security threats?

The most important environmental modification is the introduction of an architecturally independent governance layer between the agent and its deployment environment. This layer should be deployed, configured, and operated by the organization's security or compliance team, not by the team building the agent.

For rollback and negation of unwanted actions, the governance system should maintain sufficient context to support reversal. When an agent performs an action that is later determined to be unwanted, the evidence chain should contain enough information to identify what was changed and how to reverse it. For some actions (e.g., database writes, file modifications), automated rollback is technically feasible if the governance system captures the prior state. For others (e.g., sending an email, executing a financial transaction), rollback may not be possible, which reinforces the importance of pre-action approval for consequential operations.

Deployers should also consider environment-level sandboxing: running agents in isolated environments (containers, virtual machines, or restricted user accounts) with limited filesystem access, no persistent storage beyond what is explicitly granted, and no ability to install software or modify their own runtime. The governance proxy operates outside this sandbox, ensuring that even if the agent compromises its sandbox, it cannot bypass governance.

Question 4(d): What methods could be used to monitor deployment environments for security threats?

The inline proxy architecture provides a natural monitoring point. Because every agent action passes through the governance layer, that layer can produce comprehensive telemetry without requiring instrumentation of the agent itself. This telemetry should include:

Per-action records: tool name, parameters, target resource, policy evaluated, decision, latency, and response. Session-level aggregates: total actions taken, actions denied, actions escalated for approval, cumulative risk score, and session duration. Anomaly indicators: unusual tool call patterns, high denial rates (suggesting the agent is probing boundaries), requests to tools not in the agent's permitted set, and deviations from the agent's historical behavior profile.

The key advantage of monitoring at the governance layer rather than within the agent is that the monitoring cannot be disabled or manipulated by the agent. A compromised agent can suppress its own logs. It cannot suppress logs produced by an external proxy that it has no access to.

Monitoring maturity in this space is low. Most organizations deploying agents today rely on the agent's own logging, which provides neither independence nor integrity guarantees. The infrastructure security community's experience with service mesh observability (e.g., Istio, Envoy) provides a mature model for how proxy-based monitoring should work for agent systems.

5. Additional Considerations

Question 5(b): In which policy or practice areas is government collaboration with the AI ecosystem most urgent?

The most urgent area for government collaboration is the development of standardized evidence formats for AI agent governance. Today, there is no common format for recording what an agent did, what policy governed it, or what human approvals were obtained. Each vendor and deployer creates proprietary log formats. This makes cross-organizational audit, regulatory review, and incident investigation unnecessarily difficult.

NIST is well-positioned to define a standard evidence schema for agent actions, similar to how the security community benefits from shared, interoperable formats such as NIST's NVD and the broader CVE ecosystem. A standard agent action record format should include fields for: agent identity, action type and parameters, policy evaluated and version, decision and rationale, human approver identity (if applicable), cryptographic signature, and hash chain link to the previous record. If organizations adopt a common evidence format, auditors can develop standard assessment procedures, regulators can specify evidence requirements, and deployers can switch governance tools without losing audit continuity.

The regulatory landscape is fragmenting rapidly. Colorado's AI Act, California's AB 316, the EU AI Act, and multiple state-level initiatives are creating overlapping compliance obligations. A NIST-defined evidence standard would give deployers a single governance target that satisfies multiple regulatory frameworks, reducing compliance burden and accelerating responsible adoption.

Conclusion

The security of AI agent systems is fundamentally a governance problem, not solely a model safety problem. The model is one component; the critical control point is the interface between the agent and the external environment where actions produce real-world consequences. Existing cybersecurity patterns, such as inline proxying, policy-as-code, cryptographic audit trails, zero trust architecture, and human-in-the-loop approval, are mature, well-understood, and directly applicable. What is missing is their integration into a coherent governance architecture for AI agents and the development of standards that enable consistent implementation, assessment, and audit across organizations.

Panoptic Systems appreciates the opportunity to contribute to this important effort and welcomes further engagement with CAISI on these topics.

Respectfully submitted,

David Medeiros
Founder, Panoptic Systems