March 9, 2026

Peter Cihon
Senior Advisor
Center for AI Standards and Innovation
National Institute of Standards and Technology

Re: Request for Information Regarding Security Considerations for Artificial Intelligence Agents [Docket # NIST-2025-0035]

Dear Mr. Cihon,

The AI Integrators Council (AIIC) serves as the primary voice for leading companies working to integrate artificial intelligence into systems, platforms, and applications. The AIIC welcomes the opportunity to provide input to the National Institute of Standards and Technology's (NIST) Center for AI Standards and Innovation (CAISI) as it seeks information from stakeholders on security considerations for agentic artificial intelligence.

We support the federal government's efforts to harness the power of artificial intelligence innovations, particularly through agentic AI, and its support in the advancement of this emerging technology. The AIIC was created to educate policymakers and other essential stakeholders about the complexity of roles, responsibilities, and relationships across the AI ecosystem and to advocate for policies based on an inherent understanding of this ecosystem. We are comprised of leading AI companies including Alteryx, Atlassian, Box, Cognizant, Docusign, Peraton, SAIC, Salesforce, ServiceNow, and Twilio.

As the United States works to scale agentic AI across critical sectors of the economy, federal policy must recognize that deployment success depends not only on frontier model developers, and end-user deployers, but on the integrators who operationalize those systems in real-world environments. AI integrators design and secure the agent scaffolding, tool connections, data pipelines, and governance controls that enable agentic systems to move from laboratory capability to trusted, production-grade operability for sector-specific use.

The AI Integrators Council represents firms using best-in-class security practices, which is a critical layer to ensuring the safety, security, and reliability of an AI agent. The integration of agentic AI forms a specific role in the AI value chain, wholly separate from using AI agents or developing the foundational models that power them. Security must be at the very foundation of any entity in the AI value chain. The AIIC appreciates the opportunity to provide comments on security concerns pertaining to agentic AI.

RFI Questionnaire Responses

1. Security Threats, Risks, and Vulnerabilities Affecting AI Agent Systems

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

AI agent systems introduce several categories of novel security risk beyond those facing traditional software systems:

- Prompt injection and indirect prompt injection: Unlike traditional software with fixed logic paths, AI agents interpret natural language instructions that can be manipulated through form fields, attachments, images, and voice inputs. Indirect prompt injection—where adversarial instructions are embedded in data sources the agent consumes—represents a particularly challenging threat because the agent cannot reliably distinguish instructions from data.
- Agentic overreach and privilege escalation: AI agents executing on behalf of dynamic users can inherit elevated permissions beyond what is needed. Role masking addresses this by defining an allow-list of roles agents can inherit, enforcing least-privilege access and reducing the risk of AI solutions accessing resources they shouldn't, preventing agentic overreach into sensitive data or capabilities beyond what the user is allowed.
- Adversarial input exploitation: Agent systems face risks from malformed JavaScript Object Notation (JSON), jailbreak keywords, covert encodings, and zero-width character injection that can bypass reasoning safeguards. Best practice requires a multi-stage input validation pipeline: JSON schema validation, lexical blacklist, semantic filter, and Unicode/homoglyph scanning. Systems should return 400 errors and log any jailbreak attempts.
- Unauthorized tool execution: Agents may attempt to invoke unvetted, unsigned, or forbidden tool combinations. Execution allow-lists should restrict planners to vetted, signed tools and safe tool-combinations. A signed manifest should be maintained per tool, with a policy decision point denying calls to unsanctioned or forbidden combinations.
- Dormant and overprivileged agents: As organizations deploy more agents, some remain active but unused, creating latent attack surface. Security monitoring should identify agents inactive for 90+ days, flagging them as "dormant" and candidates for de-provisioning.
- Shadow AI and ungoverned agent proliferation: Multiple teams may deploy agents with little coordination. Shadow-agent detection should model baseline traffic patterns and quarantine unknown agent IDs, with detection and quarantine within one minute.
- Memory and context security risks: Agent memory stores may contain personal identifiable information (PII), secrets, or proprietary data. Memory governance controls should encrypt memories, enforce time to live (TTL), purge PII, and scan for secrets with auto-purge.
- Plan-execution divergence: Agent runtime actions may diverge from approved plans due to model hallucination or adversarial manipulation. Watchdog systems should compare

live events to signed plan hashes, alerting the security operations center (SOC) and reverting on deviations.

- Intent drift and alignment degradation: Over time, agent outputs may drift from their intended alignment. Periodic evaluation using gold quality assurance sets should benchmark outputs and quarantine agents when alignment drift exceeds defined thresholds.
- Supply chain compromise: Tampered container images, unsigned commits, or vulnerable dependencies can compromise agent systems at the build or deployment stage. Operational environment consistency controls should block deployment of images deviating from signed digests or containing critical common vulnerabilities and exposures, with software bill of materials (SBOM) generation, image signing, and continuous integration gates enforcing provenance and vulnerability policy.
- Model context window exploitation: Lengthy context windows can be exploited to crowd out security and safety instructions. Adversaries may pad inputs with benign content to push system-level constraints beyond the model's effective attention, degrading adherence to safety policies and enabling instructions that would otherwise be refused.
- Lack of federated identity systems for agents: Separate user databases between vendors for agents and users can cause unique identifier (UID) collisions and inconsistent permissions. The absence of standardized federated identity infrastructure for AI agents creates gaps in authentication and authorization that compound across multi-vendor deployments.
- Autonomous decision cascades: A single agent can cascade instructions downstream to multiple agents, amplifying the impact of a compromised or misconfigured agent. Without proper delegation controls, one malicious or erroneous decision can propagate through an entire agent workflow before detection.

Question 1(b): How do security threats, risks, or vulnerabilities vary by model capability, agent scaffold software, tool use, deployment method (including internal vs. external deployment), hosting context (including components on premises, in the cloud, or at the edge), use case, and otherwise?

Security risks vary significantly based on agent capability and deployment context:

- Dynamic user execution context: Agents that run with a dynamic user context inherit the invoking user's roles and permissions. Role masking can broaden agent functionality while reducing risk by enforcing governance around elevated or scoped privileges.
- Model Context Protocol (MCP) server connectivity: Agents connected to external MCP servers introduce additional attack surface. Monitoring at the MCP server level—tracking total tool-call transactions and success rates—provides anomaly detection across these connections.
- Voice agent modality: Voice-based AI agent systems face an entirely distinct threat profile including ultrasonic (inaudible) command injection, laser/light injection, deepfake and voice-conversion attacks, replay and loudspeaker artifact exploitation, adversarial audio perturbations, and skill name collision (squatting). These require specialized controls including ultrasonic filtering above 20kHz, laser injection hardening with physical

placement guidance, presentation attack detection for biometric authentication, and deterministic command parsing with schema-validated command representations.

- High-risk capability isolation: Agents with access to high-risk capabilities should have those capabilities limited; for example, being isolated in narrowly-scoped sub-agents with separate runtime, access control lists (ACLs) per capability, and roll-based access control (RBAC) denying cross-capability access.
- Use-case sensitivity: Security risk depends on the agent's purpose. Agents that process sensitive data, perform financial operations, or control critical infrastructure require stricter isolation, stronger authentication, detailed auditing, and more frequent attestation than low-risk assistants.
- Access map complexity: Security risk scales with the complexity of relationships between agents, workflows, and tools. A node-graph access map helps security teams investigate interdependencies and find unexpected access paths or privilege escalation routes.
- Foundational model supply chain responsibilities: Model providers carry unrecognized security responsibilities that downstream service providers and end users cannot independently verify or compensate for. Security threats vary significantly depending on whether the foundational model is open-source or closed-source, whether agent scaffold software is developed in-house or sourced from third parties, and whether the hosting context places components on premises, in the cloud, or at the edge. Each configuration introduces distinct trust boundaries and attack surfaces that must be independently assessed.
- Hosting context variations: Deployment topology significantly affects the threat profile. On-premises deployments face different risks than cloud-hosted or edge-deployed agent systems. Edge deployments introduce physical access risks and constrained monitoring capabilities, while cloud deployments introduce multi-tenancy and shared infrastructure risks. Hybrid architectures that span these contexts create additional trust boundary challenges at each transition point.

Question 1(c): To what extent are security threats, risks, or vulnerabilities affecting AI agent systems creating barriers to wider adoption or use of AI agent systems?

Security concerns create significant barriers to AI agent adoption. Organizations that lack centralized visibility into their AI agent ecosystem—including what agents are active, what permissions they hold, and what data they access—face elevated risk that inhibits deployment at scale. The absence of standardized security controls for agentic AI, combined with rapidly evolving threats, makes it difficult for organizations to confidently expand agent deployments. An offense-defense balance tracker that monitors whether attacker capability outpaces defenses, and an adoption-race watchlist that flags when competitive pressure may erode safety standards, represent emerging governance practices to manage this tension.

- Accountability and responsibility gaps: Organizations frequently lack clarity on who is responsible for agent security within their organization, across their suppliers, and among their partners. This ambiguity in the shared responsibility model delays AI

adoption as organizations struggle to assign liability for agent actions and their consequences.

- Untrusted probabilistic systems: Organizations accustomed to programmatic, deterministic systems hesitate to add probabilistic AI components to predictable business processes. The inherent stochasticity of model-driven decision-making creates resistance from risk-averse stakeholders who require deterministic guarantees that AI agent systems cannot provide.
- Compliance ambiguity: International, federal, and state regulations are not consistent in how they address agent autonomy. The boundary between automated processes and autonomous decisions is blurred, making it difficult to demonstrate compliance with oversight requirements or to prove that human oversight was adequate for a given decision.
- Auditability deficits: Many organizations lack sufficient transaction logging and business analyst involvement in developer-defined business processes. Without comprehensive audit trails that trace agent decisions back to their inputs, policies, and approval chains, organizations cannot meet internal governance requirements or respond effectively to regulatory inquiries.

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

- Growing agent proliferation: Organizations are deploying increasing numbers of AI agents, making centralized governance essential. The challenge of tracking performance, managing updates, and ensuring security compliance grows with each deployment.
- Multi-agent complexity and correlated failures: The emergence of agent-to-agent communication creates new interaction surfaces requiring protocol oversight. Correlated-failure sentinel capabilities should monitor cross-agent KPI variance and trigger safe-mode or diversification when fleet-wide anomalies are detected.
- Supply chain sophistication: Attacks targeting the agent supply chain—through compromised dependencies, unsigned artifacts, or tampered container images—are becoming more sophisticated. In-toto attestations and SBOM requirements at deployment gates are evolving as essential controls.
- Deepfake and synthetic media threats: For voice-enabled agents, the evolution of deepfake generation technology is outpacing detection capabilities, requiring continuous cross-domain robustness evaluation of deepfake detectors against unseen generators, accents, codecs, and environments.
- Autonomy elevation: Agents exceeding their autonomous authority represents a near-term risk as organizations deploy increasingly capable systems without corresponding governance maturity.
- Persistent state across AI systems: As some agent systems gain the capability to access persistent memory and cross-session state, unintentional state and information sharing between systems—across providers such as different enterprise AI platforms—creates novel data leakage vectors that current privacy controls are not designed to address.

- Model self-modification: Agents with the ability to fine-tune their own weights or modify their prompts create a new class of risk where adversarial data poisoning can be leveraged through the agent's own adaptation mechanisms, compounding over time without external intervention.

Question 1(e): What unique security threats, risks, or vulnerabilities currently affect multi-agent systems, distinct from those affecting singular AI agent systems?

Multi-agent systems introduce distinct security challenges:

- Resonance amplification: Uncontrolled inter-agent messaging can create feedback loops. A mesh gateway should enforce quotas and topic limits between agents, preventing resonance amplification by throttling over-quota inter-agent messages.
- Contradictory agent goals: When multiple agents operate on related tasks, conflicting objectives may produce harmful outcomes. A multi-agent conflict resolver using priority matrices and fairness rules should arbitrate contradictions and log rationale.
- Correlated fleet-wide failures: Identical errors across multiple agents may indicate a systemic compromise or shared vulnerability. A correlated-failure sentinel should monitor cross-agent KPI variance and activate safe-mode across the fleet when spikes are detected.
- Cross-agent trust boundaries: All cross-agent and third-party messages should be routed through a signed, scanned external trust broker that verifies source IDs, message signatures, and policy filters, dropping invalid messages and raising SIEM alerts.
- Protocol-level vulnerabilities: A centralized governance layer should provide protocol oversight for MCP and A2A protocol interactions, monitoring data access patterns and ensuring privacy compliance across all participating agents.
- Delegation authority exploitation: Orchestrator agents that delegate tasks to sub-agents may inadvertently grant those sub-agents permissions that cross trust boundaries. When an orchestrator delegates to task agents operating in different permission domains, the delegation chain can be exploited to escalate privileges beyond what any single agent was authorized to hold.
- Emergent behavior from agent interaction: Individual agents with benign, tested behaviors may combine their outputs into harmful or unintended outcomes when operating together. These emergent behaviors are difficult to predict through isolated agent testing and require system-level evaluation of multi-agent interactions.
- Information leakage between agents: Sensitive context used for agent coordination—including intermediate reasoning, user data, and system state—can leak to agent outputs or to other agents that should not have access. Multi-agent architectures must enforce information barriers that prevent context spillover across trust domains.

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 in development and deployment? What is the maturity of these methods in research and in practice?

The following technical controls represent some mature and emerging practices that may be appropriate for securing agentic AI systems:

i. Model-level controls

- Approved model provider governance: Organizations creating agentic systems should limit model choice to trusted providers.
- Output firewall: Toxicity and policy classifiers that refuse or sanitize harmful content before delivery could be used as a possible model response filter. This output firewall should log all censored outputs.
- Robust input validation and sanitization: A multi-stage pipeline should block malformed JSON, jailbreak keywords, covert encodings, and Unicode/homoglyph attacks before reaching the reasoning model. The pipeline should include JSON schema validation, lexical blacklists, semantic filters, and Unicode scanning.

ii. Agent system-level controls

- Role masking (least-privilege enforcement): Role masking lets administrators limit permissions of agentic workflows or AI agents set to run as dynamic users by defining an allow-list of roles they can inherit, enforcing least-privilege access and preventing agentic overreach into sensitive data or capabilities beyond what the user is allowed.
- Agent identity and non-repudiation: Every agent must carry a cryptographically-signed UID on every request, implemented via certificate authority-backed UID Public Key Infrastructure (PKI) with signed headers on outbound calls. Downstream services should verify signatures and reject tampered requests. External actions should be digitally signed with hardware security module (HSM)-protected keys for accountability.
- Execution allow-list: Agent planners should be restricted to vetted, signed tools/APIs and safe tool-combinations. A signed manifest per tool and a policy decision point should deny calls to unsanctioned or forbidden combinations.
- Sandbox framework: All external code should run in isolated containers. Risky plans should be simulated in a twin sandbox where dangerous plans fail dry-run before production.
- Quota and budget guardrails: Per-agent ceilings on tokens, emails, database writes, sub-agents, and energy consumption should be enforced through a central quota service with sliding windows that throttle or block on exceedance.
- AI Gateway transaction monitoring: An AI Gateway should monitor metrics at the MCP server level, listing all connected MCP servers with total tool-call transactions (defined as MCP method “tools/call”) and success rates (HTTP status 200–299 divided by total).

- Security and privacy dashboard: A dashboard-based overview should surface access issues, dormant and privileged AI agents, and map relationships between agents, agentic workflows, and tools via a node-graph access map visualization.
- Immutable agent logging: Prompts, plans, tool calls, and state diffs should be logged to append-only, tamper-evident storage using hash-chained log writers to write once, read many (WORM) storage with cold disaster recovery and 365-day retention. Hash chain integrity and inability to delete should be verified.
- Real-time behavior monitoring: Anomaly detection should identify abnormal call rates, depth, or policy deviations and auto-suspend agents. Metrics should stream to an anomaly detector integrated with a halt API and ticketing, with alerts, auto-pause, and ticket creation.
- Memory governance and privacy: Agent memories should be encrypted, with field masking, nightly secret scanning, TTL enforcement, and auto-purge. Confidential context should be redacted before external calls using PII masking, regex redaction, and classification tags.
- Visibility metadata propagation: Agent-ID, task-ID, and policy tier should be attached to every downstream call via signed provenance headers. Downstream ACLs should reject unverifiable traffic.
- Intent consistency checking: Final action traces should be semantically validated against original user intent using embedding similarity checks. Divergent plans should be flagged and aborted.
- Agent self-critique: Automated self-evaluation loops for high-risk tasks should check cost, risk, and policy compliance, with mandatory abort on failure.
- Default fail-safe behavior: On anomaly or low confidence, agents should enter safe-mode with limited read-only operations, triggered by a confidence estimator.

iii. Human oversight controls

- Mandatory human-in-the-loop escalation: Explicit human approval should be required for pre-defined high-stakes actions, with defined escalation tiers, designated approvers, workflow integration, and complete audit trails.
- Dynamic risk-score escalation: A composite risk score computed from factors including impact, novelty, and confidence should escalate actions to human review when thresholds are exceeded. To operationalize this, NIST might consider a Value-at-Risk (VaR) Oversight Framework where security controls scale dynamically based on the potential financial, legal, or operational impact of an action. For example, Low VaR tasks like document filing could remain autonomous, while High VaR actions—such as triggering a $1M procurement authorization or modifying a clinical medication dosage—should mandate a Human-in-the-Loop (HITL) checkpoint, regardless of the model's internal confidence score.
- Hierarchical interrupt and kill switch: User-level stop, tool-kill, agent-kill, and fleet-kill pathways should be available, with hardware-rooted kill application programming interface (APIs), signal terminate (SIGTERM) propagation, filesystem/memory snapshots on Sev-1 incidents, and credential revocation. Fleet halt should complete within 10 seconds.

- Structured intake and approval workflows: AI agent deployment should follow a structured lifecycle: intake and approvals, onboarding and deployment, case management, and real-time reporting. Product teams should engage a centralized AI Center of Excellence for approvals.
- Risk and compliance scoring: Proactive risk and compliance scoring with AI risk visibility, centralized oversight, impact assessments, and continuous compliance assurance should be embedded throughout the AI lifecycle, with support for regulatory frameworks including NIST AI RMF and EU AI Act.
- Delegation-scope contracts: Documented scope, limits, and revocation terms for delegated agent authority should be maintained as signed artifacts with auto-revocation workflows.
- Consent verification for high-value transactions: Multifactor authentication (MFA) consent should be required for transactions exceeding defined thresholds, with ledger recording of approvals.
- Prompt injection testing: Structured adversarial testing for prompt injection should be performed both during evaluation and at deployment. Testing should cover direct injection, indirect injection via data sources, and multi-turn injection strategies that exploit conversation context accumulation.
- Confidence threshold gating: Autonomous agent actions should only proceed when model confidence scores exceed specified thresholds. Actions below the confidence threshold should be routed to human review or constrained to read-only operations, providing a continuous risk-proportionate control.
- Network segregation: Agents should operate in restricted network zones with limited or no direct internet access. Egress filtering, network segmentation between development and production environments, and container isolation collectively reduce the blast radius of a compromised agent.

Question 2(b): To what degree, if any, could the effectiveness of technical controls, processes, and other practices vary with changes to model capability, agent scaffold software, tool use, deployment method (including internal vs. external deployment), use case, use in multi-agent systems, and otherwise?

Control effectiveness varies by deployment context. Role masking is most critical when agents run as dynamic users with inherited permissions. AI Gateway monitoring effectiveness scales with the number and diversity of connected MCP servers. Voice agent deployments require entirely additional control categories—including ultrasonic filtering, deepfake detection, step-up authentication for sensitive intents, and channel-specific hardening. The effectiveness of adversarial audio detection degrades across unseen generators, accents, and codecs, requiring continuous cross-domain evaluation.

An area of significant judgment relates to assessing the materiality of changes to underlying models, scaffold software, or agent configurations that would necessitate a re-assessment of risk or re-design of supporting controls. For example, re-training a model on new data may or may not warrant a re-assessment of risk depending on whether the new data introduces

materially different data elements. Similarly, adding novel use cases to an existing AI agent system may or may not necessitate additional controls depending on the magnitude of behavioral change. Formalized guidance on materiality thresholds would benefit the industry.

- Multi-vendor and enterprise agent systems: In deployments where agents from multiple vendors interact, controls must ensure that no agent has more agency than the original requestor. Cross-vendor trust boundaries require particularly rigorous enforcement, as each vendor's security model may make different assumptions about identity, authorization, and data handling.

Question 2(c): How might technical controls, processes, and other practices need to change, in response to the likely future evolution of AI agent system capabilities or of the threats, risks, or vulnerabilities facing them?

As agent capabilities advance, controls must evolve. Governance platforms should maintain built-in support for emerging regulatory standards with updatable risk scoring dashboards. Multi-agent systems require new forms of protocol oversight and cross-agent security monitoring. The shift toward increasingly autonomous agents calls for stronger formal verification methods to provide mathematical guarantees about plan safety. Voice agent controls must continuously adapt to advancing deepfake and voice synthesis technology through ongoing adversarial robustness testing. Protocol version gates should block agents using deprecated API versions to maintain security hygiene.

Designing controls for AI agent systems will become increasingly challenging as individual agents are used across multiple use cases with varying risk levels. Controls must be flexible enough to adapt to multi-agent systems but rigid enough to ensure agent output remains consistent with approved use cases. Guidance on using AI-based capabilities to monitor AI agent activity would support comprehensive trust assurance without handicapping the business value of democratized agent development and deployment.

Question 2(d): What are the methods, risks, and other considerations relevant for patching or updating AI agent systems throughout the lifecycle, as distinct from those affecting both traditional software systems and non-agentic AI?

AI agent systems require distinct update practices beyond traditional software patching:

- Verified update pipeline: Any prompt or code update should require signed commits, SBOM generation, and automated security scans. Unsigned builds should be blocked by CI, with the blocking reason logged.
- Dependency freezing: Runtime dependencies should be frozen and updated only after vulnerability and compatibility testing. Lockfiles and container digests should be used, with scanners blocking critical CVEs.
- Blue/green agent deployment: Canary versions should be deployed with automatic rollback on service level objective (SLO) breach. A staged green cluster with health metrics should trigger automatic rollback within five minutes of detecting error spikes.

- Suitability evaluation pre-deployment: Reliability benchmark suites should be run before production deployment, including domain-specific test suites and red-team scenarios, with a requirement for 95%+ pass rate to proceed.
- Protocol version gating: API gateway lifecycle tags should deny deprecated endpoints, returning 410 Gone errors to agents using deprecated API versions. Beyond these technical controls, AI agent systems present unique lifecycle management challenges distinct from traditional software:
- Model weight updates: When model weights are updated, capabilities and behavior change in ways that are difficult to predict. Mitigation requires phased rollout with regression testing on agent-specific task suites before full production deployment.
- Prompt template governance: Prompts change more frequently than code and are often modified by non-engineering stakeholders. Prompt governance should be enforced through a managed prompt library with version history, approval workflows, and rollback capability.
- Tool schema changes: Schema changes in connected tools can cause previously validated agent calls to fail silently or produce incorrect results. Mitigation requires versioned endpoint APIs with backward compatibility enforcement and automated schema compatibility testing.
- Lifecycle framework inconsistency: Existing lifecycle frameworks are inconsistent in how they address AI agent security. Security controls may not be applied in the appropriate phase, consistently applied across phases, or clearly understood between model providers, service providers, and consumers. Standardized lifecycle guidance for AI agent systems would address this gap.

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

- NIST AI Risk Management Framework (NIST AI 100-1): Governance platforms should provide built-in support for managing performance against this framework.
- EU AI Act: Compliance mapping and risk classification aligned with EU AI Act requirements should be embedded in governance tools.
- NIST SP 800-53: The catalog of security and privacy controls provides foundational controls applicable to agent system infrastructure.
- NIST AI 100-2e2025: The adversarial ML taxonomy provides vocabulary for characterizing agent-specific threats.
- NIST SP 800-218A: The secure development profile for generative AI addresses development-phase security.
- Annual certification audits: Independent assessment of the full control set, similar to SOC-type reports, should validate comprehensive security posture.

Additional frameworks with direct relevance to AI agent security include:

- OWASP Top 10 for LLM Applications: Provides a prioritized taxonomy of LLM-specific risks including prompt injection, sensitive information disclosure, supply chain compromise, data and model poisoning, improper output handling, excessive agency,

system prompt leakage, vector and embedding vulnerabilities, misinformation, and unbounded consumption.
- ISO 42001: Establishes requirements for an AI management system, providing an organizational governance framework applicable to agent system development and deployment.
- MITRE ATLAS and SAFE-AI: ATLAS provides a knowledge base of adversarial tactics and techniques against AI systems. The SAFE-AI framework extends this with structured assessment methodologies for AI security evaluation.
- Coalition for Secure AI (CoSAI): CoSAI's emerging work on AI shared responsibility models, the Secure AI Framework (SAIF), and MCP security guidance directly addresses the multi-stakeholder trust challenges inherent in agentic AI deployments.

Adoption of these frameworks among AI agent system developers and deployers remains uneven. Key impediments include the rapid pace of agent capability evolution outstripping framework updates, a shortage of practitioners with combined AI and cybersecurity expertise, and misconceptions that traditional cybersecurity controls are sufficient for agent-specific threats. Some existing cybersecurity best practices—such as perimeter-based security models—may be inadequate for agent systems that operate across trust boundaries and interact with diverse external resources.

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?

- Automated AI asset discovery: Organizations should leverage automated discovery with integrations to cloud providers (AWS Bedrock, Azure AI Foundry) and code repositories to maintain comprehensive inventories of all AI assets including systems, models, datasets, and prompts.
- Continuous agent audit: Consistent replay of logs against policy baselines should surface compliance drift. Batch jobs should score compliance, flag variances, and export evidence bundles. Injected policy violations in staging logs should be reliably detected and alerted.
- Chaos engineering for agents: Controlled failure injection—including dependency outages and latency spikes—should verify agent resilience. Scheduled chaos experiments validate graceful degradation behavior.
- Red teaming (including voice-specific scenarios): Continuous adversarial evaluation should include structured scenarios for privacy leakage, privilege escalation, resource abuse, and social engineering. For voice agents, this includes testing with spoofed voices and tool exploitation chains.
- Access mapping and security dashboards: Node-graph visualization of agent-workflow-tool relationships enables investigation of interdependencies. Security dashboards surface access issues, dormant agents, and privileged agents. System-wide risk scoring dashboards guide action and oversight.

- Post-deployment incident detection: AI Gateway monitoring of MCP server-level transactions provides continuous anomaly detection. Post-incident forensic snapshots should capture filesystem, memory, and logs on Sev-1 incidents to append-only storage with hash chain integrity.
- Bug-bounty and disclosure programs: Public vulnerability disclosure and bounty programs with triage SLAs, reward schedules, and integrated issue trackers, support community-sourced security assessment.
- Third-party agent assessment: Security and ethics review of vendor agents before integration should include penetration testing, SBOM review, privacy impact assessment, and sign-off requirements before production deployment.
- Data quality and drift monitoring: Data drift monitoring with defined SLO thresholds should block deployments when data distribution drift exceeds acceptable limits, triggering auto-retrain or escalation. Additionally, an agent could be grounded in the specific rules of its industry, allowing integrators to prevent it from being deceived into ignoring a mandatory federal requirement.

Complementary assessment practices from traditional security disciplines, adapted for AI agent systems, further strengthen the assessment posture:

- AI system documentation: Comprehensive documentation—including AI system component inventories, architecture diagrams, threat models, security analyses, and remediation recommendations—provides the foundation for systematic security assessment.
- Automated adversarial testing: Known attacks should be automated as test cases as they are discovered, freeing security teams to focus on novel attack vectors. Continuous integration of adversarial test suites ensures that previously identified vulnerabilities do not regress.
- Threat modeling at the AI system level: Threat modeling should be performed at the full AI system level—not just at the model level—to understand how emerging threats and mitigations interact across the agent scaffold, tools, data stores, and external integrations.
- Red teaming of both AI model and application layers: Red team exercises should evaluate both the AI model and guardrail layer and the full application supported by AI and agentic components. AI practitioners who understand AI model vulnerabilities may not understand the environments in which agents are deployed, and vice versa.
- Honeypot tools and data: Placing signature data or decoy resources in specific locations that should not be accessed by authorized agents provides a detection mechanism for improper agent behavior, complementing anomaly-based monitoring with deterministic tripwires.

Question 3(b): Not all security threats, risks, or vulnerabilities are necessarily applicable to every AI agent system; how could the security of a particular AI agent system be assessed and what types of information could help with that assessment?

The security of a particular AI agent system can be assessed through multiple dimensions: its role inheritance model (which roles it can assume from invoking users), its tool access footprint (which MCP servers and tools it connects to), its dormancy status (active within 90 days), its position in the access map (interacting agents, workflows, and data sources), its quota utilization (tokens, API calls, sub-agents consumed), its alignment drift status (benchmark evaluation against gold quality assurance sets), its plan-execution consistency (deviation between approved plans and runtime actions), and its self-awareness verification (whether it correctly identifies its policy tier and permissions before acting).

Assessment scope should be right-sized by identifying key risk determinants including the agent's autonomy level, tool access scope, data classifications involved, deployment scale, interface types, user population characteristics, and applicable regulatory risk categories such as those defined by the EU AI Act. Not every agent requires the same depth of assessment; risk-proportionate evaluation ensures security resources are allocated where they have the greatest impact.

Question 3(c): What documentation or data from upstream developers of AI models and their associated components might aid downstream providers of AI agent systems in assessing, anticipating, and managing security threats, risks, or vulnerabilities in deployed AI agent systems?

Upstream model developers should provide signed SBOMs documenting model dependencies, in-toto attestations for build provenance, adversarial robustness evaluation results, safety benchmark scores, documented known failure modes, and data governance disclosures including training data characteristics. For voice agent models specifically, cross-domain robustness evaluation results against diverse generators, accents, and codecs should be disclosed.

Standardized documentation formats would accelerate downstream security assessment. Emerging practices include:

- Model, system, and dataset cards: Public cards describing the product, training data sources, benchmarks, known limitations, update policies, architecture, safety alignment methods, and licensing provide baseline transparency for downstream security assessment.
- Developer-facing documentation: Non-public cards with greater technical detail—including internal architecture decisions, adversarial evaluation results, and known failure modes—would significantly aid downstream providers in anticipating security issues.
- Dataset cleansing disclosures: Documentation of the types of unwanted data scanned for and removed during training—including CSAM, malware signatures, copyrighted material, PII/PHI, and other categories—would help downstream providers assess residual risks in model behavior.

- Scope of responsibility disclosures: Explicit documentation of which risks are the responsibility of the model provider versus the downstream deployer would reduce ambiguity in the shared responsibility model.

Question 3(d): What is the state of practice for user-facing documentation of AI agent systems that support secure deployment?

In our perspective as integrators primarily serving in the business-to-business context, user-facing documentation should include: detailed configuration guides for security features (role masking, AI Gateway monitoring, security dashboards), prerequisite requirements and required roles, signed skill/tool manifests with version pinning and integrity checks, delegation-scope contracts documenting agent authority limits and revocation terms, and an end-user "Why-Did-It-Do-That?" API providing plain-language rationale for agent decisions via role-based access control (RBAC)-protected endpoints.

The current state of user-facing documentation varies significantly. Industry-typical documentation generally covers agent capabilities, configuration and setup guides, and usage examples. Industry documentation rarely includes security responsibility models, threat models, or incident response procedures.

4. Limiting, Modifying, and Monitoring Deployment Environments
Question 4(a): AI agent systems may be deployed in a variety of environments, i.e., locations where the system's actions take place. In what manner and by what technical means could the access to or extent of an AI agent system's deployment environment be constrained?

- Role masking: Administrators define an allow-list of roles agents can inherit from invoking users, enforcing least-privilege access and expanding security configurations while reducing risks for elevated or scoped roles.
- Sandbox framework: All external code should run in isolated containers. Risky plans should be simulated in a twin sandbox prior to production execution.
- Execution allow-lists: Agent planners should be restricted to vetted, signed tools with policy decision points denying unsanctioned calls.
- Quota and budget guardrails: Per-agent ceilings on tokens, emails, data base writes, sub-agents, and energy should be enforced with sliding-window throttling.
- Temporal-scope limiting: Agents should be prevented from scheduling tasks beyond an allowed time horizon, with policy-configured maximum future days and planner validators refusing long-horizon tasks.
- Approved model provider lists and data routing: Deployment environments should restrict model providers to approved lists and enforce regional data routing policies.
- Sensitive capability segregation: High-risk capabilities should be isolated in narrowly-scoped sub-agents with separate runtime and RBAC-enforced access controls.

- Agentic database access controls: Database queries should be enforced through parameterized templates, row-level ACLs, and sensitive column redaction. Ad-hoc SQL queries should be denied.
- Network-level constraints: Egress filtering to restrict internet access, network segmentation isolating development, staging, and production environments, and container-level isolation collectively constrain the deployment environment at the infrastructure layer.
- Resource quotas at the infrastructure level: Limits on CPU, memory, disk I/O, and network throughput at the container and VM level provide a backstop against runaway agent resource consumption, complementing application-layer quota guardrails.
- Environment enclaves: Strict isolation between sub-production and production environments prevents agents' under development from affecting production data or systems. Read-only testing modes that prevent agents from executing writes provide an additional safety layer during evaluation and debugging.

Question 4(b): How could virtual or physical environments be modified to mitigate security threats, risks, or vulnerabilities affecting AI agent systems? What is the state of applied use in implementing undoes, rollbacks, or negations for unwanted actions or trajectories (sequences of actions) of a deployed AI agent system?

- Hierarchical interrupt and kill switch: User stop, tool-kill, agent-kill, and fleet-kill pathways should be available with hardware-rooted kill APIs, SIGTERM propagation, and filesystem/memory snapshots on Sev-1 incidents.
- Blue/green deployment with auto-rollback: Canary deployments with staged clusters should monitor health metrics and trigger automatic rollback.
- Default fail-safe behavior: On anomaly or low confidence, agents should enter safe-mode with limited read-only operations.
- Planned-vs-executed diff alerting: Watchdog systems comparing live events to signed plan hashes should revert deviations and alert the SOC.
- Dormant agent lifecycle management: Agents inactive for 90+ days should be flagged for de-provisioning, with administrators able to set them inactive or delete entirely.
- Memory leak detection: Container monitoring should detect memory leaks and automatically recycle affected agents with log preservation.
- Honey objects and decoy resources: Traditional honeypot techniques—including false network services, honey records in databases, and honey accounts—can be adapted for agentic environments. These decoy resources provide deterministic detection of unauthorized agent access attempts, complementing statistical anomaly detection with high-confidence indicators of compromise.

Question 4(c): What is the state of managing risks associated with interactions between AI agent systems and counterparties?

- Interactions with humans not using the system: Social-manipulation detectors using NLP classifiers should flag persuasion or coercion cues in agent outputs targeting users,

escalating to ethics review. User psychological-safety guards should filter harmful content before reaching users, blocking self-harm content and providing help resources.

- Interactions with digital resources: An AI Gateway should monitor all tool-call transactions at the MCP server level. Agentic database access controls enforce parameterized templates and row-level ACLs. Confidential context redaction scrubs proprietary or sensitive data before external calls.
- Interactions with mechanical systems/IoT: For agents interfacing with physical systems, hardware safety interlock interfaces should require dual-channel PLC confirmation for physical actuation, with PLCs enforcing speed and force envelopes and blocking out-of-bounds commands.
- Interactions with authentication mechanisms: Secure time synchronization using network time security (NTS)-secured network time protocol (NTP) should reject clock drift beyond defined thresholds, blocking cryptographic operations on clock anomalies. Session binding to authenticated speaker and channel should prevent session hijacking.
- Interactions with other AI agent systems: An external trust broker should route all cross-agent and third-party messages, verifying source IDs, message signatures, and policy filters. Multi-agent coordination boundaries should enforce quotas and topic limits between agents.

Across all counterparty interaction types, two cross-cutting risks require particular attention:

- Social engineering through agents—where agents are used as intermediaries to manipulate humans—requires robust labeling to identify agent-generated messages and content.
- Social engineering of agents—where humans manipulate agents through carefully crafted inputs—requires the adversarial input controls and behavioral monitoring described in earlier sections.
- Privacy leakage across interaction boundaries remains a persistent challenge requiring continuous monitoring and context redaction at every external interface.

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

- Security and privacy dashboards: Real-time AI security metrics including access issues, dormant agent counts, privileged agent identification, and relationship mapping via access maps.
- AI Gateway transaction monitoring: Continuous monitoring of MCP server-level tool-call transactions and success rates.
- Real-time behavior monitoring: Anomaly detection for abnormal call rates, depth, or policy deviations with auto-suspension.
- Immutable logging: Hash-chained, append-only logging to WORM storage with retention and disaster recovery.
- Risk scoring dashboards: System-wide risk scoring aggregating individual risks into enterprise AI risk profiles with support for NIST AI RMF and EU AI Act.

- Shadow-agent detection: Baseline traffic pattern modeling with quarantine of unknown agent IDs.
- Dynamic prompt watermarking: Invisible watermarks embedded in prompts for attribution and leak tracing, enabling identification of the originating agent when prompts are leaked externally.
- Mental-model transparency logging: Encrypted storage of planner reasoning in secure enclaves, accessible only by authorized auditors for forensic investigation.
- Effective monitoring requires layered telemetry spanning multiple levels: agent-level logs capturing decisions and tool calls to audit tables; system-level metrics including infrastructure compute and memory usage, API call rates, and traffic volume; application-level metrics tracking user request volume, agent success and failure rates, and latency statistics; and user feedback signals where user edits to agent outputs serve as implicit indicators of agent errors.
- Observability and behavioral signatures: Beyond raw telemetry, observability systems should process signals centrally and establish statistical baselines from which deviations can be detected. Behavioral signatures—including honey object access attempts—and contextual anomalies such as changes in agent behavior across different application contexts provide additional detection layers.
- Regulatory uncertainty in monitoring: Significant open questions remain about what agent activity can be observed, by whom in the AI value chain, and what log content is appropriate to share given data sensitivity.

5. Additional Considerations

Question 5(a): What methods, guidelines, resources, information, or tools would aid the AI ecosystem in the rapid adoption of security practices affecting AI agent systems and promoting the ecosystem of AI agent system security innovation?

Rapid adoption of AI agent security practices is best facilitated by: centralized governance platforms that embed security into existing enterprise workflows; automated AI asset discovery and onboarding; structured intake-to-deployment lifecycles with embedded governance approvals; pre-built compliance framework mappings (NIST AI RMF, EU AI Act); real-time security dashboards with actionable metrics; bug-bounty and vulnerability disclosure programs with triage SLAs; and comprehensive control frameworks that provide testable, scenario-based verification for each control objective.

Critical gaps that would accelerate adoption include: standardized reference architectures for secure agent deployment; agent-specific security testing frameworks; incident response playbooks with clear escalation paths for agentic security incidents; shared threat intelligence feeds for adversarial prompts and agent-specific attack patterns; and standardized model security benchmarks that enable apples-to-apples comparison of agent robustness across providers.

Question 5(b): In which policy or practice areas is government collaboration with the AI ecosystem most urgent or most likely to lead to improvements in the state of security of AI agent systems today and into the future?

Government collaboration is most urgent in: developing standardized agent security control taxonomies and assessment methodologies; extending the NIST AI RMF with agent-specific security controls; establishing testing frameworks for agentic security controls including both text-based and voice-based agent modalities; creating shared adversarial testing libraries and red-team scenarios; and developing standards for agent identity, provenance, and non-repudiation that enable interoperability across agent ecosystems.

Additional priority areas for government collaboration include: developing a shared responsibility framework—building on emerging efforts such as the CoSAI AI Shared Responsibility Model—to clarify legal ambiguity on liability for agent actions; providing adversarial testing resources to organizations that lack in-house AI red team expertise; and establishing centralized incident reporting mechanisms for agentic security incidents, which currently lack any central tracking and could have broad impact across interconnected systems.

Question 5(c): In which critical areas should research be focused to improve the current state of security practices affecting AI agent systems?

Critical research areas include: security of agent-to-agent communication protocols and multi-agent trust boundaries; formal verification methods for agent plan safety; automated detection of alignment drift and specification gaming in deployed agents; cross-domain adversarial robustness for voice agent deepfake detectors; techniques for monitoring agent behavior across complex tool-access topologies; methods for cost-of-action assessment that integrate economic and environmental costs; and resilience testing methodologies including agent-specific chaos engineering.

Additional research priorities identified through production experience include: business process auditing methodologies for automated and autonomous business processes classified by autonomy level; frameworks for ensuring business-level transaction integrity in agentic workflows; explainability methods sufficient for security auditing of agent decisions; formal proof methods for agent restrictions (e.g., proving that an agent will never delete critical data); and adversarial robustness benchmarks that provide standardized resistance measures against adversarial input across agent types and deployments.

Question 5(d): How are other countries addressing these challenges and what are the benefits and drawbacks of their approaches?

The EU AI Act provides the most comprehensive international regulatory framework directly relevant to AI agent security, with risk classification requirements that can be mapped through centralized governance platforms. Organizations should build governance capabilities that

support compliance with multiple jurisdictions simultaneously through configurable compliance framework mappings.

International approaches offer distinct lessons: the EU AI Act has contributed a risk categorization framework that provides structure for classifying agent deployments by risk tier. China's AI regulations have content restriction models that address output safety. The U.S. approach has initially focused on AI safety through executive orders. A sharper U.S. focus on data security, supply chain security, and clear responsibility allocation would complement existing international frameworks and provide more technically enforceable standards for AI agent systems.

Question 5(e): Are there practices, norms, or empirical insights from fields outside of artificial intelligence and cybersecurity that might benefit our understanding or assessments of the security of AI agent systems?

Several practices from adjacent fields offer valuable analogies for AI agent security: weighted-factor safety review workshops (adapted from safety engineering) provide structured cross-functional evaluation; hardware safety interlocks from industrial control systems apply to agents interfacing with physical systems; chaos engineering from site reliability engineering validates agent resilience under controlled failure conditions; and digital signature and non-repudiation practices from financial services provide accountability models for agent actions.

Additional cross-domain insights with direct applicability include:
Aviation and medical devices (FAA, FDA): These domains require validated datasets for flight data and clinical studies, respectively. The principle of requiring formal data validation before deployment—and the regulatory infrastructure that enforces it—provides a model for establishing standards around training dataset screening for unwanted content including CSAM, malware signatures, PII/PHI, copyrighted material, and other formally categorized unwanted data types.

Automotive autonomy classification (SAE J3016): The SAE J3016 taxonomy of driving automation levels (L0–L5) is already being informally applied to AI agent autonomy. Formalizing an analogous classification for agent autonomy would enable risk-proportionate security controls calibrated to the agent's degree of autonomous authority.

Financial services identity standards: Know Your Customer (KYC) requirements in financial services provide a model for strong identity verification. AI agents and their transactions need comparable identity standards and infrastructure to establish accountability chains and prevent unauthorized actions.

Conclusion

The AI Integrators Council appreciates the Administration's commitment to modernizing AI policy, and working to better secure the future of agentic AI. AI integrator companies often work with agentic AI by embedding agents within enterprise workflows, constraining permissions, implementing oversight mechanisms, and ensuring compliance with sector-specific requirements, thereby determining whether agentic AI operates safely and effectively at scale. Yet despite their central role in translating model advances into economically and operationally viable systems, AI integrators remain largely undefined in existing policy frameworks, which often focus narrowly on model development rather than full-stack deployment. The AIIC stands ready to continue supporting this effort and to serve as a partner in shaping policies that strengthen agentic AI innovation and security.

Sincerely,

Wes McClelland
Executive Director
AI Integrators Council