Amazon response to the Request for Information re: *Security Considerations for Artificial Intelligence Agents* from the National Institute of Standards and Technology and the Center for AI Standards [NIST-2025-0035]

Introduction

Agentic AI systems represent a qualitative shift in how software functions. Traditional software executes deterministic instructions planned, written, and operated by developers. Generative AI foundation models respond to human prompts with output humans review and use at their discretion. Agentic AI systems differ from both: they connect to software tools and APIs and use large language models (LLMs) as "reasoning" engines to plan a task and then act on that plan without specific human direction. Thus, agents can plan and execute sequences of actions autonomously, at machine speed, and with real-world consequences. This shift—to software that is "intelligent" rather than deterministic and AI systems that are "active" rather than passive—raises new questions and challenges for information security that must be carefully analyzed and addressed. Amazon Web Services is pleased to be able to respond to the Request for Information on agentic AI security based on our extensive yet still fast-growing experience in developing systems based on agentic AI, as well as providing AI services that support our customers in doing the same.

What is agentic AI? One common definition: "Agentic AI is an autonomous AI system that can act independently to achieve pre-determined goals."[1]. Another simple and more practical definition is, "An AI agent runs models [LLMs] and tools in a loop to achieve a goal."[2]

Why is agentic AI security such a pressing issue? Why take risks that such autonomous systems may create? The answer is because the possible efficiency and performance gains from agentic AI can be enormous, and thus for AI usage in many domains even a conservative risk/benefit analysis will indicate that the benefits clearly outweigh the risks. The rapid adoption of agentic technology in business and government going on today is a clear indication of that reality. At the same time, there is evidence some end users may be adopting agentic technology without fully understanding it, and without sufficient safeguards.[3] Just as with other forms of security education for end users (phishing, social engineering, etc.), education around reasonable and low-risk usage of agentic AI will have to be a new focus for businesses and, where possible, consumers. The good news is that the security response to agentic AI need not be built from scratch, as existing security considerations, frameworks, and controls still apply.

---

1. The full definition and description of agentic AI from our website:

   > Agentic AI is an autonomous AI system that can act independently to achieve pre-determined goals. Traditional software follows pre-defined rules, and traditional artificial intelligence also requires prompting and step-by-step guidance. However, agentic AI is proactive and can perform complex tasks without constant human oversight. "Agentic" indicates agency — the ability of these systems to act independently, but in a goal-driven manner. AI agents can communicate with each other and other software systems to automate existing business processes. But beyond static automation, they make independent contextual decisions. They learn from their environment and adapt to changing conditions, enabling them to perform sophisticated workflows with accuracy. For example, an agentic AI system can optimize employee shift schedules. If an employee is off sick, the agent can communicate with other employees and readjust the schedule while still meeting project resource and time requirements.

   Amazon Web Services, "What is Agentic AI?," https://aws.amazon.com/what-is/agentic-ai/ (paragraph breaks omitted).
2. See Brooker, "Agent Safety is a Box," https://brooker.co.za/blog/2026/01/12/agent-box.html (Jan 2026).
3. See, *e.g.*, The Wall Street Journal, "The World's First Viral AI Assistant Has Arrived, and Things Are Getting Weird," https://www.wsj.com/tech/ai/openclaw-ai-agents-moltbook-social-network-5b79ad65 (Feb. 4th, 2026).

These can readily be extended to meet the new challenges so that this important new technology can be safely developed, deployed, and operated.

Responsibility for the security of agentic AI systems is shared across three parties. AI model and agentic framework providers, those who develop and supply foundation models and infrastructure for agentic systems, bear responsibility for the security of their components and for transparency about their capabilities and limitations. Builders, those who assemble agentic systems from these components, bear final responsibility for the behavior of the whole system, including the permissions that agents have to tools, tools have to data sets, and the actions they are permitted to perform. They can rely on the deterministic controls and infrastructure that providers supply. But, due to the fact that AI components behave probabilistically and their outputs depend on how they are configured and combined, builders must determine and implement the appropriate permissions, policies, and oversight for their specific system. Providers can offer the mechanisms for expressing constraints, but the knowledge of which constraints are appropriate for a given use case rests with the builder. Finally, end users, those who acquire and utilize agentic systems, must conduct their own risk/benefit analysis. Agents' "creative," nondeterministic goal-seeking behavior is the source of both what makes them so valuable and what ensures that agentic systems may not behave exactly as expected or even intended.

This shared responsibility framework has a direct implication for how security guidance should be structured. Providers should provide infrastructure that enables fine-grained, deterministic controls where feasible. Builders should apply that infrastructure and configure those controls rigorously, treat AI components with the same security lifecycle discipline as traditional software, and design systems that fail safely. End users should scope their deployments to match their risk tolerance and invest in evaluation and monitoring as ongoing operational practices.

Our response identifies four foundational security principles for agentic AI, followed by architectural building blocks that can implement them. One important point to note up front is that existing security frameworks, including the NIST Cybersecurity Framework and 800-53, NIST AI RMF, SSDF (SP 800-218), and SP 800-63, as well as ISO/IEC 42001, remain highly relevant and generally applicable. They should be extended for agent-specific risks rather than replaced. The most important extension is architectural: security for agentic AI should be enforced primarily through deterministic, infrastructure-level controls external to the agent, and not solely through the agent's own reasoning or an LLM's internal or external guardrails. Through careful monitoring and experience, some deterministic controls may be relaxed over time to enable more agentic "freedom," providing improved efficiency and overall better outcomes. Moreover, as agentic technology evolves and improves, no doubt there will be increased use of security agents designed to enforce what were once rule-based boundaries in more intelligent and human-like ways; deciding when, for example, an exception to a permission boundary actually makes sense. But that evolution will take some time, and some hard outer boundaries will likely always be maintained.

Background: Traditional versus Agentic Software

Traditional software systems operate on a foundational principle: behavior is specified in code, and given the same inputs, the system will normally produce the same outputs. This predictability is the basis on which traditional security controls are designed and validated. For our purposes we will use the term "deterministic" to characterize how traditional software generally behaves.[4] It also defines the limit of

4. We say "normally" and "generally" because even traditional software may under certain conditions exhibit non-deterministic behavior. As a simple example, floating point operations can give different results on different processors or even different generations of the same processor. That fact also directly impacts the behavior AI models, which rely heavily on floating point arithmetic. Another perhaps more important example is the interaction of components in a complex distributed system. Given sufficient complexity, the notion that "the

(Continued on next page…)

what traditional software can do: it cannot handle situations its developers did not anticipate, and complex workflows requiring judgment have historically required human participation. Most enterprise systems are therefore human-software hybrid systems, where software handles tasks deterministically and humans handle the portions requiring discretion and adaptation, *i.e.*, intelligence.

Large Language Models (LLMs) and agentic systems change this dynamic in two fundamental ways that challenge traditional approaches to security. First, unlike traditional software, LLMs are probabilistic.[5] Rather than specifying behavior through explicit rules, LLMs derive behavior from a variety of data sources and can "reason through" how to respond to novel inputs in ways that were not specifically designed or anticipated by their developers. Agents utilize LLMs as the basis for their "reasoning" capacity.

Second, agentic systems plan and execute sequences of actions with some degree of autonomy: invoking tools, querying data sources, calling APIs, writing and running code, and interacting with external services that have traditionally required human intervention and judgment. AI agents are valuable precisely because of their relative autonomy and adaptability. These same characteristics may lead them to perform unexpected or risky actions due to accidental or intentional misuse by users, or an agent's own inherent goal seeking behavior. An agentic system that carries out an unintended action can cause harm directly, at machine speed, before any human can intervene. Human actors in hybrid enterprise systems exercise judgment, operate under social and business expectations, and will typically pause or escalate when an instruction seems unusual or exceeds their normal scope. Agentic systems, in contrast, can and should inherit the authentication and authorization constructs designed for human actors but do not necessarily inherit human discretionary judgment (although some simulations of such discretion are typically present). This leads directly to the core architectural requirement: *if there are critical actions you would not want a human to take without additional authorization, those protections must be explicitly encoded in the agentic system.* You cannot, for example, depend on an agent to recognize unstated policy boundaries.

In the following section, we discuss four security principles that can help in designing and analyzing agentic AI systems. They derive from decades of security experience, adapted and extended to address the unique characteristics of agentic AI.

Security Principles for Agentic AI

We frame the agentic AI discussion as a set of security principles, two of which carry forward existing security principles into the agent era, and two of which are new. The new principles come in to play to account for the probabilistic nature of agentic actions and applications, and to constrain those actions appropriately until, based on experience, sufficient warrant exists for providing more agentic "freedom."

---

same inputs result in the same outputs" begins to break down, and with that strong determinism. That said, the generalization in the text is still reasonably accurate and very useful for our purposes.

5. We use the term "probabilistic" rather than "non-deterministic" advisedly; read on for our explanation. With model "temperature" set to zero, the same model executing on exactly the same hardware will produce the same result. In that sense AI systems can be deterministic. Still, that does not take into account the likely variations in output resulting from the highly variable human inputs (prompts, instructions, ticket content, etc.) that is typically present in any given agentic workflow. Moreover, as previously noted, even different generations of the same CPU or GPU architecture can return different floating-point results, and modern LLMs—whether in training or inference phases—make heavy use of floating-point operations carried out across large clusters of computers with some heterogeneity of hardware, with those clusters and their hardware also changing frequently over time. So in many cases some degree of real non-determinism is present in ML-based AI systems. That said, because AI can be said to be deterministic under some conditions, instead of "non-deterministic" we will use the term "probabilistic" to summarize the key difference between traditional software and AI-based systems.

Principle 1: Secure development lifecycle practices apply across all system components

A secure development lifecycle (SDL) approach must encompass both traditional software components and AI elements—neither is sufficient alone.

Agentic AI systems combine traditional software components (software tools, services with APIs and CLIs, databases, orchestration logic, etc.) with AI elements (foundation models, prompt templates, prompt and response guardrails, retrieval pipelines, RAG databases, etc.). SDL discipline must cover both, ensuring that each component operates as intended through regular evaluations and re-evaluations.

For traditional software components, this means code review, static analysis, dependency scanning, integration testing, threat modeling, and deployment validation[6]. For AI components, the challenge can be more complex. Foundation models are probabilistic systems, which means that traditional regression testing is necessary but likely insufficient. Organizations must supplement it with behavioral testing, adversarial evaluation, and continuous behavior monitoring to validate that AI components operate within expected parameters.

Regular re-evaluation is equally important for addressing behavioral drift. Models receive updates that can alter behavior. Prompt templates evolve as teams refine agent capabilities. New tools and data sources expand the agent's operational surface. Each change can introduce new failure modes or security issues. Organizations must treat evaluation as an ongoing operational practice, not a one-time gate, including if possible automated testing after model updates, red team exercises against deployed models and agents, and monitoring that detects behavioral drift over time.

Principle 2: Traditional security controls remain fully applicable

Agentic AI introduces novel attack surfaces but does not render existing risks obsolete. The full complement of traditional security controls still applies.

An agentic AI system is composed of both traditional software and the new "LLM+tools" processing loop. All the existing software and tools and configurations remain subject to a well-known set of risks and must be secured in order to provide a secure framework for the agentic elements.

For example, flaws related to identity and access management remain a threat vector in agentic systems. Privilege escalation flaws may allow attackers to expand an agent's operational scope beyond its intended boundaries. Confused deputy attacks exploit an agent's legitimate permissions by manipulating it into performing unauthorized actions on behalf of the wrong user through prompt injection or data manipulation. Session hijacking might commandeer active agent sessions, inheriting whatever permissions the agent has accumulated and allowing those to be used for harm. Excessive privileges granted to agent identities create risks analogous to those of human actors, but an agent is capable of leveraging those privileges at greater scale and speed. Therefore, applying principles of least privilege to access management is equally if not more important in an agentic context. Tool integrations introduce additional challenges related to credential management. Tools calling remote APIs often rely on static tokens that require secure storage, rotation, and access controls. Centralized credential management for both agents and tool integrations—where the underlying platform provides temporary tokens and handles token rotation rather than leaving that to individual tool implementations—reduces credential exposure risks.

Similarly, code injection and other forms of inappropriate code execution attacks extend directly into agentic systems. SQL injection, command injection, cross-site scripting, and insecure deserialization flaws in the traditional software components surrounding AI agents can bring about undesirable results. Agents that invoke APIs, query databases, or generate code create new potential injection surfaces at

6. See, *e.g.*, NIST SP 800-218 (transposed to National Standard ANS 585) and soon to be balloted as an ISO/IEC JTC1 standard).

various tool boundaries. Supply chain attacks also pose a threat. Compromised software dependencies, poisoned training data, tampered model weights, and tampered tool integrations can introduce risks independent from the quality or design of the agent. Agentic systems may have a broader supply chain surface than traditional software, consuming not only third-party foundation models but also plugins, tool servers, and data retrieval sources.

The listed risks are only a sample of traditional security matters to be considered. Every traditional attack technique, including denial-of-service, man-in-the-middle, vulnerability and configuration exploitation, supply chain, log tampering, etc., remains relevant in agentic contexts. AI-specific controls must be additions to foundational security, not replacements for it.

Principle 3: Deterministic external controls are the key starting point for agentic security

Security for agentic AI systems should begin with deterministic, infrastructure-level controls external to the agent, not through the AI-based controls such as the agent's own reasoning, internal guardrails, or prompt-based instructions. "Agent safety is a box."[7] As agentic technology evolves, however, the box itself will become at least partially agentic.

This is the most important architectural principle for agentic AI security in the near term. It follows from the nature of LLMs. LLMs are probabilistic reasoning engines, not security enforcement mechanisms. An LLM can be instructed to refuse certain requests, or to respond to questionable inputs only in appropriate and safe ways, but prompt injection techniques often can override those instructions. An LLM can be told to respect access boundaries, but it has no reliable mechanism to enforce them. Attempting to constrain agent behavior only through prompting or alignment runs against the fundamental value proposition of agents: their ability to adapt dynamically to novel situations.

Effective security therefore begins by placing fully specified, deterministic controls outside the agent's reasoning loop. This means controls that limit which tools the agent can access and what it can do with those tools, limits that cannot be bypassed through model manipulation or context injection. This "safety box"—or for present purposes better understood as a "security box"—has three critical properties. It is *external*: immune to prompt injection and model behavior changes. It is *deterministic*: unlike the agent's adaptive behavior, the box enforces exact, predictable constraints that can be tested and even formally verified. And it is *comprehensive*: every interaction between the agent and the outside world passes through its access controls and policy enforcement layers.

Not all agentic systems require the same level of constraints. AWS's Agentic AI Security Scoping Matrix[8] categorizes agentic architectures into four scopes based on their connectivity and relative autonomy. Scope 1 (no agency) covers human-initiated processes with no autonomous change capabilities. Scope 2 (prescribed agency) requires explicit human approval before an agent acts. Scope 3 (supervised agency) supports complex autonomous tasks with human-triggered execution and oversight. Scope 4 (full agency) encompasses fully autonomous systems that initiate their own activities based on external events. Organizations should classify their agentic systems by scope and calibrate their deterministic constraints accordingly.

A particularly high-risk architectural pattern is worth mentioning here. It is known as the "dangerous triad"[9] and occurs when an agentic system design simultaneously grants agents access to sensitive data, exposes agents to untrusted external content or inputs that can impact their reasoning, and permits agents to communicate any result of their activities externally. This combination creates conditions for data

7. Brooker, *supra* note 2.
8. See https://aws.amazon.com/ai/security/agentic-ai-scoping-matrix/.
9. Also known as the "lethal trifecta," a phrase and concept attributed to Simon Willison, see https://simonwillison.net/2025/Jun/16/the-lethal-trifecta/.

exfiltration through input manipulation. Deterministic constraints must break at least one element of this triad for any given agent. Architectural separation, which builders should configure to be enforced at the infrastructure level, delivers defense that does not depend on the model's ability to resist manipulation.

Over time, however, the "security box" itself will evolve. AI agents will become substitutes for some of the rule-based, deterministic constraints that enable the first generation of agentic technology to be introduced safely into enterprise, government, and consumer workflows. Done deliberately and carefully, this development can actually increase security because agents will be able to use new information and new context to make better automated security decisions than may be possible with humans attempting to manage a complex set of deterministic controls.

To summarize: The "security box" is not a limitation on the agent's value. On the contrary, it is the precondition for achieving that value responsibly in contexts where unintended behavior would have unacceptable consequences. But the box can be expanded over time, and agentic technology can be applied over time to make the box more flexible and powerful, which leads directly to the final principle.

Principle 4: Greater autonomy should be earned through usage and on-going evaluation

Organizations should expand agent autonomy progressively based on demonstrated performance, not grant it by default initially. The mechanism that enables expansion is on-going evaluation, and the vehicle is the progressive evolution and decrease of human-in-the-loop oversight.

The starting point in the development and deployment of an agent system is human decision-making for high-consequence operations. When an agent encounters an action that (for example) could modify production data, initiate financial transactions, or communicate sensitive information externally, a human makes the final decision. The agent recommends; the human approves or rejects. Determining which operations qualify as high-consequence initially requires expert human judgment, supported where appropriate by AI-based analysis and risk scoring.

This approach carries a known risk: decision fatigue. If every agent action requires human approval, the volume of decisions may overwhelm reviewers. Attention degrades, and approval becomes reflexive rather than deliberate, shifting liability to humans placed in a position to fail rather than improving the efficiency or quality of outcomes, including those with security consequences. As they evolve through the various phases outlined by the Agentic AI Scoping Matrix, from limited to more complete autonomy, organizations must scope human decision-making to genuinely high-consequence operations and resist the temptation to require approval for routine actions that carry low risk.

The path from human decision-making to expanded autonomy runs through evaluation. Organizations should systematically record the outcomes of human-in-the-loop decisions: what the agent recommended, what the human decided, and what the actual outcome was. When evaluation demonstrates that human reviewers agree with agent recommendations at a sustained high rate over a meaningful sample, the organization can shift from prior approval to post-hoc review. The agent acts, and a human reviews the action or actions (possibly in some kind of summary report form) after the fact. This preserves oversight while increasing the value of the automation and reducing human labor and operational friction. The data and learnings from on-going evaluation will also feed the development of oversight agents that can gradually replace human oversight with agents designed for those tasks.

Continued evaluation can warrant a further transition to full autonomy for specific operation types, when the data demonstrates consistent, reliable agent performance over an extended period and when the consequences of occasional errors are manageable. Oversight agents will also become more and more the norm. This progression should normally happen at the operation or workflow level, not across a broader range of unrelated tasks.

Critically, this progression is not necessarily unidirectional. Continued usage and review may reveal performance degradation, behavioral drift, or changes in the operating environment that warrant reducing

autonomy. While this is somewhat unlikely given that the underlying AI models will tend to improve over time rather than degrade, organizations nevertheless should be prepared to reintroduce human-in-the-loop oversight, temporarily or permanently, when on-going review data indicates that previously earned trust is no longer justified, and then analyze the causes of the degradation understand and correct it.

Throughout this progression, some deterministic limits will typically remain permanent, or only be replaced by sophisticated agents developed specifically for oversight purposes. Even as inner constraints relax based on demonstrated performance, outer boundaries persist at the edges of acceptable system behavior. Deterministic controls that disable exfiltrating sensitive data to unauthorized external endpoints, for example, would not relax regardless of track record, although some may be replaced by AI-based controls when the data shows an overall improvement in results. These boundaries define the limits of the system's autonomic envelope; constraints that exist not because the agent has not yet earned trust, but because the consequences of violation are unacceptable under any reasonable risk/benefit analysis.

The overall model is one of *earned autonomy through demonstrated competence*, governed by evaluation, bounded by permanent constraints, and subject to continuous review.

Agentic AI Security Architecture

The four principles given above define the high-level goals and approaches for secure agentic AI. This section identifies categories of capabilities and controls that can be used to compose a security architecture to achieve those principles and goals. Multiple building blocks make up this architecture, and each reinforces the others. We also provide references to AWS cloud capabilities as illustrative of some of the considerations discussed in this section.

Deterministic Security Provided by the Agentic Execution Context

The execution context is the implementation of the "security box" described in Principle 3, the set of deterministic, infrastructure-level controls that operate outside the agent's reasoning loop and enforce desired behavior, including security constraints. The following building blocks provide this context.

Compute platform. The compute environment must isolate agent execution, prevent cross-agent and cross-session data leakage, and contain potentially compromised agents within defined boundaries. In a multi-tenanted environment the compute infrastructure must provide very strong isolation properties between and among compute nodes. History has shown repeatedly that using container-based isolation (for example) is not a sufficiently strong isolation boundary to ensure cross-tenant isolation in any kind of general compute platform. Hardware-backed virtualization, conversely, has generally provided excellent results. Even within a single tenancy, such as an on-premises deployment or isolated cloud capacity, strong compute isolation between agent execution contexts remains a best practice if not a necessity. It is quite likely that agents will be handling different kinds of data at different levels of classification, and thus strong cross-agent isolation remains an important property. Moreover, compromise of any particular agentic execution environment should be isolated from other same-tenant environments to the greatest extent practicable.

In AWS, the Amazon Bedrock AgentCore framework for agentic AI uses Firecracker, an open source virtual machine manager written in Rust, as its compute platform.[10] Firecracker provides so-called "micro VMs," extremely lightweight virtual machines based on the hardware-backed isolation of the Linux Kernel Virtual Machine manager (KVM).[11] By removing many features not needed in serverless environments, Firecracker provides the speed and convenience of containers while maintaining the security properties of full virtualization. In addition, many security-critical elements of Firecracker have

10. See Brooker, "Seven Years of Firecracker," https://brooker.co.za/blog/2025/09/18/firecracker.html (2025).
11. See "Secure and fast microVMs for serverless computing," https://firecracker-microvm.github.io/.

been formally verified by AWS teams,[12] adding greater assurance beyond the memory and thread safety provided by Rust.

Identity and access management. Agents need a variety of capabilities to interact with external systems in a secure fashion. This includes both intrinsic identities for the agents, as well as a secure service for storing and accessing credentials (OAuth tokens, API keys, etc.) as needed for tool invocations. These non-human / machine identities should be created and managed through some kind of directory service for centralized control and audit. Authentication flows should support both two-way machine-to-machine interactions (2LO in OAuth terminology), and also three-way interactions where the user must participate in the flow (3LO). Agents should normally be identifiable to the infrastructure's access control and audit layers either by impersonating the user who invokes them, or as a distinct identity within the IAM system, or some "on behalf of" combination of the two. The relationship between agent actions and the prompting or supervising agent or supervising user should also be traceable and auditable through any delegation chain.

On the authorization side, least-privilege permissions should be configured when authorizing users to invoke agents, and when authorizing agents to use permitted tools, resources, and operations. Authorization enforcement must operate at the infrastructure level, independent of the agent's reasoning. Enforcement layers between agents and tools will often possess their own identities, creating additional accountability boundaries and enabling audit of independent policy decisions.

The Amazon Bedrock AgentCore Identity implements these concepts and more, and provides SDKs that hide from developers the complexities of things like declarative the complexities of the OAuth flows.[13] It integrates deeply with the native AWS IAM system, supporting basic IAM plus more advanced patterns like attribute-based access control, and also with the broader ecosystem using rich OAuth and JWT support.

Agentic memory. Most agentic uses case will require, or at least be greatly enhanced by, the use of "memory" by the agent. Memory consists of a record of previous turns or interactions taken by an agent in a given workflow, or past workflows of the same type. It can also be thought of as durable session state for long-lived sessions. Memory can create new kinds of risk if not managed carefully and fully isolated across security contexts. For example, as a general matter, the memory of an agent working on behalf of one user or one microservice should be fully isolated from work the same agent does on behalf of another user or microservice. This requires some kind of underlying session management that ensures that memory is isolated and there are no "cross wires" when an agent is in used by different parties. It's helpful for an agentic framework to provide external control over the length of time that memory is retained and the context in which it is used.

Bedrock AgentCore Memory is a first-class entity with a rich set of features.[14] It provides a set of APIs that let AI agents seamlessly store, retrieve, and utilize both short-term and long-term memory. The architecture is designed to separate the immediate context of a conversation from the persistent knowledge that should be retained over time. The memory system has a simple but useful schema (strategy, namespace, record, session, actor, event, metadata, etc.) and supports different built-in

12. See, *e.g.*, "How Open Source Projects are Using Kani to Write Better Software in Rust," https://aws.amazon.com/blogs/opensource/how-open-source-projects-are-using-kani-to-write-better-software-in-rust/ (2023). See also "Using Kani to Validate Security Boundaries in AWS Firecracker," https://model-checking.github.io/kani-verifier-blog/2023/08/31/using-kani-to-validate-security-boundaries-in-aws-firecracker.html (2023) and "Using the Kani Rust Verifier on a Firecracker Example," https://model-checking.github.io/kani-verifier-blog/2022/07/13/using-the-kani-rust-verifier-on-a-firecracker-example.html (2022).
13. See https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/identity.html.
14. See https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/how-it-works.html.

"memory strategies" that builders can use for different use case. Complete custom strategies are also supported.

Access to tools. Tools are how agents affect external state. Every tool[15] to which an agent is granted access expands both its usefulness and its potential risk. Agents should receive access to only the tools required for their intended function. Tool risk must be assessed based on what data the tool can access, what state it can change, and whether it can communicate beyond established boundaries. Even what may seem to be read-only types of tools may inadvertently disclose confidential information in an unintended fashion. For example, an agent carrying out web-based research on behalf of a user may expose the content of the questions that the user is asking and possibly even indirectly expose the purpose of the research. Policy enforcement at the tool boundary should be granular enough to constrain not just which tools an agent can use, but how it can use them, down to individual parameters and operations, ideally with decisions informed by the identities in the delegation chain.

Configuring and administering one-to-one relationships between agents and their tools will result in a NxN combinatoric explosion of relationships to manage. Thus, such an approach is not realistic for large scale agent deployment and usage; some kind of centralization of the administration of agent-to-tool access is a practical requirement. One good approach is the use of a "smart proxy" service that acts as a centralized and deterministic intermediary between a pool of agents and a pre-configured pool of tools. The service can centrally enforce authentication and authorization requirements and can have its own identity used or added to authorization flows, such that use of the service can be enforced by access policies. Such a service not only provides the mechanism for configuring which tools can be accessed, and what operations are authorized with a given tool, but it can go beyond normal IAM capabilities, which often operate at the granularity of allowing or disallowing only at the tool or API level, and inspect and allow tool or API calls based on the *values of parameters* in those calls. The service can also provide enriched metadata about the tools so that agents can automatically select their own tools with contextual search.

In AWS, these kinds of capabilities are provided by Bedrock AgentCore Gateway.[16] The Gateway provides a rich set of capabilities including all those listed above, plus things like translation between different protocols, composition of APIs, functions, or tools into a single MCP endpoint, content filtering, and many other features that enable centralized configuration and management of key security-relevant agentic capabilities. It also provides deterministic constraint tool called Policy in AgentCore Policy[17] based on the Cedar authorization language,[18] which leverages formal methods to provide strong assurances of correctness (more on formal methods below).

Observability. Observability is a critical capability for agentic systems. System designers and administrators must be able to capture sufficient context of agent operations for real-time monitoring anomaly detection as well as logging of all agent behavior *post hoc* investigation and response. Observability infrastructure must be protected from the agents it monitors. Organizations would not allow employees to edit their own audit logs, and the same principle applies to agents. As agentic systems mature, organizations can introduce agents that oversee other agents, but this should be gradual and evidence-based, with human oversight remaining available for escalation, and with architectural separation to prevent recursive trust failures. Observability is greatly enhanced and simplified at the same time if a "smart proxy" service is in use, as noted above, since that provides a central point for monitoring, alerting, and logging/auditing.

---

15. We use the term "tool" broadly to encompass any and all software tools, APIs and CLIs, headless web browsers, code execution sandboxes, MCP servers, databases, etc.
16. See https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/gateway.html.
17. See https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/policy.html.
18. See https://www.cedarpolicy.com.

Observability is a key sub-service within the Bedrock AgentCore environment.[19] In addition to the default external observation of agent interactions via the Gateway service as well as the telemetry accumulated at the session level as managed by the AgentCore service, developers can get even deeper insights via an internal observation technique called "traces" simply by adding telemetry libraries to their agent code. Traces record all internal state changes to a CloudWatch Log stream as well, and can also be used for observability of agents not hosted in AgentCore or even running inside of AWS.

Model execution environment. While not strictly related to agentic AI, given the central role played by LLMs in agentic systems, the security of foundation model/LLM usage remains of critical importance. In particular, where does the model execute, and what commitments does the model provider or model hosting service make to isolation and privacy within that execution environment? Configuring an agent to use a non-approved first-party LLM API across a public network creates a number of additional risks compared with using a model of a high reputation provider executing inside a secure private network with no access by the model provider to the prompts and responses, and no access to the outside world by the code and infrastructure supporting the model's execution.

AWS Bedrock, for example, has been widely adopted by government and enterprise customers precisely because of the architectural isolation it provides, and the guarantees it makes about the confidentiality of customer content, including prompts and responses. By default, models execute in secure, isolated network environments, and customer prompts and responses are not logged.[20] Neither AWS nor model providers ever have access to customer prompts and responses, and thus cannot use customer content to for model improvements. Customers can enable logging of prompts and responses, but those logs are encrypted at rest and protected from outside access by customer-managed KMS encryption keys.[21]

Security Approaches Within AI-based Agentic Behavior

While an LLM is obviously a fundamental element in an agentic AI environment, its probabilistic nature means the system designers cannot rely on it for strict security enforcement. That said, careful design of the AI elements of an agentic system can also help improve its security. We cover some of those elements here briefly.

Model selection and update. Developers of agentic systems must consider the commitment of the model provider to responsible AI, and the model's demonstrated resistance to adversarial inputs, as well as the model provider's security testing practices. System or model cards can be used to evaluate the degree to which and the means by which a model developer endeavors to provide models that conform to best practices. Model updates can alter security-relevant behavior and should be treated as security updates requiring validation of existing controls. Models selected for agentic use should ideally be evaluated and compared to one another in testing and pre-production scenarios. This kind of side-by-side comparison is greatly facilitated by using a model execution environment that supports easy access to many different models like Amazon Bedrock.

External guardrails. All data entering and leaving agentic workflows should undergo validation addressing both traditional injection attacks and AI-specific manipulation such as prompt injection. Some model execution environments support external filters or "guardrails" that inspect the prompts entering the LLMs and/or its responses and seek to improve model behavior and block inappropriate content flowing in either direction. In Amazon Bedrock, the Guardrails feature supports a range of capabilities, from simple (but still quite useful) rules that inspect prompts and responses with regex-like algorithms to the more widely useful and sophisticated rules that utilize small, narrowly tailored AI models known as

19. See https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/observability.html.
20. See https://docs.aws.amazon.com/bedrock/latest/userguide/data-protection.html and https://aws.amazon.com/blogs/security/securing-generative-ai-data-compliance-and-privacy-considerations/.
21. See https://docs.aws.amazon.com/bedrock/latest/userguide/data-encryption.html.

classifiers that handle challenges like prompt injection quite effectively. It also includes even more sophisticated guardrails based on adherence of model responses to a formal specification (more on that below). All of these tools that apply to direct human usage of LLMs are just as if not more important in an agentic context, and should be utilized therein to decrease undesirable behavior and increase security.

System prompts. System prompts are instructions to the LLM that exist outside the agent's control, but are added to its "conversations" with the LLM to guide the results towards more desirable outcomes. This approach can improve quality very generally, and that includes improved security-related behaviors. While in human usage system prompts are often large, monolithic documents that are added to the LLM's context window, for agentic systems it makes sense to break down such instructions into more documents that are more modular and contextual, delivered at relevant points during execution rather than as a single monolithic meta-prompt. This makes both functional and security-relevant instructions easier to maintain and validate, and also reduces the risk of seemingly conflicting guidance that can create exploitable ambiguity. In the Strands open source agentic framework, for example, "steering" documents can be utilized to continually update guidance to the LLM as an agent progresses through its workflow.[22] Injecting what amount to evolving system prompts at the relevant points in an agentic workflow will lead to better outcomes than loading all context upfront and hoping that the agent interprets follows those instructions at the right time in the course of its work.

Formal methods

Formal methods or, in AWS parlance, automated reasoning can provide additional capabilities in the agentic AI context that span both deterministic "outside the security box" controls as well as use cases inside the AI prompt/response flow. Automated reasoning is the use of tools that embody formal logic and mathematical principles to verify that system behavior conforms to specified formal model (not to be confused with an AI model), producing results that are deterministic, transparent, and provably correct.

Automated reasoning (AR) techniques can be applied in various ways to improve the behavior of LLMs and the agents that rely on them. AWS has applied it in two areas, both briefly noted above. First, the AR-based Cedar authorization language has been utilized as the engine for AgentCore Policy to strengthen the "security box." Second, AR has been applied as a new kind of Bedrock Guardrail to improve the behavior of LLMs and, derivatively, agents relying on those LLMs by guarding against hallucinations.

Cedar is an open source domain-specific language and runtime optimized for writing and evaluating authorization policies. Cedar is based on AR in two respects. First, the language itself is designed so that policies written in the language can be formally verified by its accompanying validator tools. Second, the Cedar runtime engine that evaluates policies is based on a formal model developed in the Lean language / interactive theorem prover accompanied by parallel production implementation written in Rust,[23] with differential testing used to prove the equivalence of the two.[24] The use of the Cedar language as the basis for AgentCore Policy to augment the power of the AgentCore Gateway brings these high assurance properties into the agentic AI domain. But AgentCore Policy also takes advantage of the power of LLMs by allowing Cedar policies to be drafted using natural language followed by human review.[25]

In the Bedrock Guardrails context, Automated Reasoning checks allow developers to create a formal model of a knowledge domain and pass all LLM output through an automated analysis that provides a

22. See https://strandsagents.com/latest/documentation/docs/user-guide/concepts/experimental/steering/.
23. See Cutler et al., "Cedar: A New Language for Expressive, Fast, Safe, and Analyzable Authorization (Extended Version)", https://arxiv.org/pdf/2403.04651 (2024).
24. See Hicks, "How we built Cedar with automated reasoning and differential testing," https://www.amazon.science/blog/how-we-built-cedar-with-automated-reasoning-and-differential-testing (2023).
25. See https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/policy-natural-language.html.

detection to the developer that the response is valid, invalid, or undecidable, with a reference to the relevant rules that led to the determination.[26] A developer can send valid responses back to the requester, and for invalid or undecidable responses, decide whether to inform the requester and stop, or ask clarifying questions. Similar to AgentCore Policy's authoring strategy, this feature also leverages LLMs to help users to create the formal model of the knowledge domain in which their application is operating by uploading an informal representation (such as corporate policy document) to generate the formal rules and then guiding them through resolution of remaining ambiguities. The Automated Reasoning checks approach clearly can benefit agents just as much as humans for general LLM output quality and correctness, including helping to reduce misbehaviors that could have a security impact.

Other considerations

Human-in-the-Loop techniques and their progressive development. As described in Principle 4, implementing progressive autonomy requires risk-based scoping of human oversight to avoid decision fatigue. The measurement infrastructure must systematically record decision outcomes, creating the evidentiary basis for expanding or restricting autonomy. As organizations gain confidence, agents can augment human oversight, but this transition must be gradual, evidence-based, and reversible. Approval mechanisms must be resistant to manipulation by the agents being overseen. Agentic systems generally provide a number of mechanisms for interfacing with humans and recording those interactions, from instant messaging systems to email to ticketing systems and just about any other interactive platform used by enterprises.

Continual evaluation and testing. The probabilistic nature of agentic systems demands evaluation beyond traditional testing. Four levels must operate simultaneously: behavioral testing to validate expected boundaries; adversarial testing to subject the agent to deliberate attacks; runtime validation to catch behavioral drift in production; and regression testing after every component change. Amazon Bedrock includes AgentCore Evaluations, an entire subsystem designed to assess the performance of AI agents.[27] It can compute metrics such as an agent's end-to-end task completion (goal attainment) correctness, the accuracy of a tool invoked by the agent while handling a user request, and any custom metric defined to evaluate specific dimensions of an agent's behavior. The AgentCore Evaluations can evaluate the AI agents that are hosted under AgentCore Runtime as well as AI agents hosted outside of AgentCore.

Updating and patching agentic system components. As discussed in Principle 1, agentic systems comprise multiple component types with independent update cycles, and each change can alter the system's security posture. A model update can weaken security defenses without any change to the surrounding system; a prompt update or a change of tool permissions can alter agent behavior as directly as a code change. Organizations should validate system behavior when any component changes, treat prompt and other AI-related updates with the same rigor as code changes, and maintain rollback capabilities at every level of the system. Rollback capabilities are of critical importance as they allow the rapid "undo" of systems changes when degraded performance of an agentic system is detected after an update.

Challenges

Several challenges complicate achieving the principles using the building blocks described above.

Managing of deterministic controls at scale. The "security box" architecture described in Principle 3 is sound, but its implementation becomes more complex as agentic systems scale in scope, tool access, and autonomy level. Granular limits and policy enforcement across large fleets of agents and complex tool

26. See https://docs.aws.amazon.com/bedrock/latest/userguide/guardrails-automated-reasoning-checks.html.
27. See https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/evaluations.html.

ecosystems requires significant engineering investment. As with any complex security and governance goal, organizations operating at Scope 3 or Scope 4 autonomy levels will face challenges in maintaining comprehensive, up-to-date deterministic constraints as their systems evolve.

Emergent behavior in complex, multi-agent systems. The security properties of individual agents do not simply aggregate into the security properties of multi-agent systems. Interactions between agents—including inter-agent message injection, trust boundary abuse, persistent memory corruption, and orchestrated distributed attacks among agents with overlapping privileges—may produce harmful outcomes that no individual agent's controls are able to prevent. This is analogous to emergent phenomena in other complex systems. Individually well-behaved components can produce collectively problematic behavior. Current security frameworks have limited tools for reasoning about emergent multi-agent behavior, so this is an area requiring further research and development.

Evaluation methodology maturity. The graduated autonomy model in Principle 4 depends on systematic evaluation, but the methods for evaluating agentic AI behavior are still maturing. Behavioral benchmarks, adversarial red-teaming protocols, and runtime monitoring approaches for agentic systems are less standardized than their equivalents for traditional software. Organizations seeking to implement evidence-based autonomy expansion will need to invest in developing their own evaluation infrastructure while the field matures.

Conclusion

Agentic AI is not a marginal extension of prior software paradigms. It is a qualitative shift that requires security frameworks to evolve; but not to start over. The foundational insight of this response is that existing security frameworks, including the NIST Cybersecurity Framework, NIST AI RMF, SSDF, and SP 800-63, provide the right foundation. The task ahead for the industry is to extend them for agentic-specific risks, not to replace them.

The most important area of extension relates to the architectural considerations discussed above. Security for agentic AI must be enforced primarily through deterministic, infrastructure-level controls external to the agent and the LLM. LLMs are not security enforcement mechanisms, and treating them as such—that is, relying on prompting, alignment, or internal guardrails as primary security controls—is a category error that will likely produce insecure systems. The "security box" is not optional. It is the precondition for responsible deployment. In addition, as discussed above the proper usage of features and capabilities within the probabilistic AI context can improve security as well, and those techniques should also be implemented wherever possible. Finally, security frameworks need to be extended to take into account the evolution of agentic systems from limited to greater autonomy. Autonomy must be earned through demonstrated performance, not granted by default. The mechanism is systematic evaluation, and the vehicle is the progressive evolution of human-in-the-loop oversight. This approach transforms the expansion of agent autonomy from a judgment call into an evidence-based decision, including the ability to reduce autonomy when evidence warrants. Providers should build capabilities and features that make it easier for builders and users to carry out their respective responsibilities when building and using agentic systems.

NIST and CAISI can make the useful contributions by extending existing frameworks rather than creating parallel ones, by encouraging model cards and security documentation from AI providers that give builders the visibility they need to manage inherited risk, and by examining the development of evaluation methodology standards for agentic AI behavior. The security community has many of the necessary tools and is rapidly developing others. The task is to apply them effectively to this new class of systems.

Appendix: Response to Specific Questions from NIST

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

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

(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?

(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?

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

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

Amazon response

(a) Unique security threats in AI agent systems include the threat of prompt injection, both direct and indirect, memory poisoning, inserting information that modifies behavior in unexpected ways, and probabilistic behavior itself, making it challenging to adequately test overall system behavior. In addition, other security threats for traditional systems such as confused deputy attacks may be elevated in risk in AI agent systems.

(b) An exhaustive analysis of models, scaffolding, tools, and deployment mechanisms is beyond the scope of this document, but suffice it to say that each additional tool deployed in an AI agent system increases both the potential utility of the system, as well as the complexity and risk profile. Two additional ways to impact the risk profile of an AI agent system are the mechanisms the agents use for identity and access management, and the level of human oversight built in to the system. Systems with greater autonomy have more opportunities for surprising and undesirable behavior than systems that require human approval gates for agentic actions.

(c) AI agent systems are seeing rapid adoption today. To the extent that customers use strong security implementations and defense in depth philosophies when designing their AI agent systems, we will continue to see successful projects and rapid growth of these systems. That said, as with any technology in a relatively early phase, organizations may struggle accurately to assess the risks of agentic systems, and will thus find it hard to make a well-informed risk/benefit analysis. This is not so much a literal barrier to adoption but a typical phase in a technology adoption life cycle.

(d) We are early enough in the adoption life cycle that "change over time" is hard to assess. As to future evolution, customers will gradually reduce the amount of human oversight that they apply to their AI agent systems, as well as increasing the complexity of the systems. The underlying models in these systems are becoming more capable, allowing for more extended autonomous operations. If done carefully with the right kinds of "security box" and other constraints, we foresee continued positive growth with minimal downsides. Some organizations, however, will likely move too quickly without adequate safeguards, with some resulting bad outcomes. The resulting news cycle will result in a spike in associated "fear, uncertainty, and doubt." Again, this is typical of any adoption of new technology.

(e) Multi-agent systems must deal with inter-agent message injection, trust boundary abuse, unexpected results from persistent shared memory, and orchestrated distributed attacks using agents with distinct privileges. Adequate testing and evaluation of such systems will also be a challenge.

2. Security Practices for AI Agent Systems

(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? Categories may include:

i. Model-level controls, such as measures to enhance model robustness to prompt injections;

ii. Agent system-level controls, such as prompt engineering, data or tool restrictions, and continuous monitoring methods;

iii. Human oversight controls, such as approvals for consequential actions, management of sensitive and untrusted data, network access permissions, or other controls.

(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?

(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?

(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?

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

i. What is the extent of adoption by AI agent system developers and deployers of these relevant guidelines, frameworks, and best practices?

ii. What are impediments, challenges, or misconceptions about adopting these kinds of guidelines, frameworks, or best practices?

iii. Are there ways in which existing cybersecurity best practices may not be appropriate for the security of AI agent systems?

Amazon response

(a) Security can be enhanced through deterministic, infrastructure-level controls external to the agent's reasoning loop and tool access; what we have called the "security box." These controls cover compute isolation, tool access restrictions, IAM with least privilege, and observability infrastructure. Additional controls, such as deterministic or AI-based guardrails on model prompts and responses, within the AI loop can also help. As discussed above, automated reasoning and formal methods can further strengthen these controls by verifying that system behavior conforms to specified policies in a provably correct manner.

(b) The effectiveness of controls varies significantly with autonomy level and tool access. We recommend that customers use the Agentic AI Scoping Matrix (Scopes 1–4) to calibrate the required rigor of deterministic constraints. At higher autonomy levels (Scope 3–4), maintaining comprehensive, up-to-date deterministic constraints becomes both more important but also more complex and engineering-intensive. In multi-agent systems, the security properties of individual agents do not easily aggregate into system-level security, requiring additional architectural separation and inter-agent trust controls.

(c) We recommend a progressive model in which deterministic constraints are relaxed over time only as demonstrated performance warrants, with certain hard outer boundaries remaining permanent for high-risk decisions regardless of the agentic system's track record. As agentic systems mature and agents begin overseeing other agents, architectural separation must continue evolve to prevent recursive trust failures.

(d) Building on traditional software, agentic systems comprise multiple component types: models, prompts, tools, retrieval pipelines, each with independent update cycles, and any change may alter the system's security posture without any change to surrounding components. We recommend treating prompt and AI-related updates with the same rigor as code changes, validating system behavior after any change, and maintaining rollback capabilities at every level of the system.

(e) The existing NIST Cybersecurity Framework, NIST AI RMF, SSDF (SP 800-218), and SP 800-63 are the most relevant existing frameworks. They should be extended for agent-specific risks rather than replaced.

3. Assessing the Security of AI Agent Systems

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

i. What methods could be used to detect security incidents after an AI agent system has been deployed?

ii. How do these align (or differ) from traditional information security practices, including supply chain security?

iii. What is the maturity of these methods in research and applied use?

iv. What resources or information would be useful for anticipating, identifying, and assessing security threats, risks, or vulnerabilities?

(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?

(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?

i. Does this data or documentation vary between open-source and closed-source AI models and AI agent systems, and if so, how?

ii. What kinds of disclosures (if made mandatory or public) could potentially create new vulnerabilities?

iii. How should such, if any, disclosures be kept secure between parties to protect system integrity?

iv. What is the state of practice for user-facing documentation of AI agent systems that support secure deployment?

Amazon response

(a) We recommend applying a full SDL discipline to both traditional software and AI components, supplementing traditional threat modeling and static analysis with behavioral testing, adversarial evaluation, and red team exercises specifically designed for agentic systems. Four levels of evaluation must operate simultaneously: behavioral testing to validate expected boundaries, adversarial testing to subject the agent to deliberate attacks, runtime observation and validation to catch behavioral drift in production, and regression testing after component changes. The Agentic AI Security Scoping Matrix provides a useful framework for matching risk assessment to the autonomy level of the system under evaluation.

i. We recommend deploying observability infrastructure that captures sufficient context of agent operations for investigation and response, with real-time anomaly detection complementing historical logging. Critically, this observability infrastructure must be protected from the

agents it monitors — just as organizations would not allow employees to edit their own audit logs, agents must not be able to alter the records of their own actions.

ii. We view the changes needed for agentic systems as an extension of existing security and governance practices of traditional human-software hybrid systems that agentic systems will gradually augment or replace. We recommend treating existing security practices such as threat modeling, SDL, IAM, and supply chain security as the foundation, extended rather than replaced for agentic contexts. The key difference is that agentic systems have a broader supply chain surface than traditional software, consuming not only third-party foundation models but also AI-specific plugins, tool servers, etc., each of which introduces risk that builders must assess. Traditional regression testing remains necessary but is insufficient for AI components, which require supplemental behavioral and adversarial evaluation due to their probabilistic nature.

iii. The methods for evaluating agentic AI behavior are still maturing. Behavioral benchmarks, adversarial red-teaming protocols, and runtime monitoring approaches for agentic systems are less standardized than their equivalents for traditional software. Organizations seeking to implement evidence-based autonomy expansion will need to invest in developing their own evaluation infrastructure while the field matures. Investment in standardized evaluation methodology will contribute greatly to the field.

iv. The Agentic AI Security Scoping Matrix [28]provides a practical starting point for assessing a particular system, enabling organizations to classify their deployment by autonomy level and calibrate their deterministic constraints accordingly.

(b) We recommend using the Agentic AI Security Scoping Matrix as the primary tool for scoping a threat assessment to a particular system, classifying the system by autonomy level (Scopes 1–4) based on its connectivity, tool access, and degree of autonomous action, and then calibrating the required deterministic constraints accordingly. For model assessment, responsible providers generally provide "system cards" or "model cards" that give developers insight into the training data, training techniques, and guardrails built into the models. The "dangerous triad" framework discussed in the main text provides a complementary lens.[29] Together, these information sources and frameworks allow organizations to focus their security investment on the threat vectors that are actually present in their deployment, rather than treating every agentic system as if it carries the full threat surface of the most complex possible configuration.

(c) The references given in the previous paragraph are relevant and helpful here. The product / service documentation of the capabilities and features of a platform designed to support building and running agents, especially those related to the "security box" and its observability framework, are necessary inputs to the secure design and operation of an agentic system. Automated evaluation and testing will also be of great importance.

4. Limiting, Modifying, and Monitoring Deployment Environments

(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?

(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?

28. See *supra*, note 8.
29. See *supra*, note 9.

(c) What is the state of managing risks associated with interactions between AI agent systems and counterparties? Practices, their adoption, and their relative maturity may differ according to the counterparty in the interaction, including:

i. Interactions with humans who are not using the AI agent system directly;

ii. Interactions with digital resources, including web services, servers, and legacy systems;

iii. Interactions with mechanical systems, machinery, or Internet-of-Things (IoT);

iv. Interactions with authentication mechanisms, operating system access, source code access, or similar network-level access vectors;

v. Interactions with other AI agent systems.

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

i. What challenges exist to deploying traditional methods of monitoring threats, risks, or vulnerabilities?

ii. Are there legal and/or privacy challenges to monitoring deployment environments for security threats, risks, or vulnerabilities?

iii. What is the maturity of these methods in research and practice?

(e) Are current AI agent systems widely deployed on the open internet, or in otherwise unbounded environments? How could the volume of traffic be tracked on the open internet or in otherwise unbounded environments over time?

Amazon response

(a) Deployment environments can be constrained through deterministic, infrastructure-level controls external to the agent's reasoning loop. This is the "security box" that includes compute isolation, least-privilege IAM, and granular tool access policies that restrict not just which tools an agent can invoke but how it can use them, down to individual parameters and operations, as discussed above. The Agentic AI Security Scoping Matrix (Scopes 1–4) provides a practical framework for calibrating the need for rigor of these constraints based on the autonomy level of the specific deployment.

(b) Virtual and physical environments can be hardened by ensuring all components, including tools, go through security review and vulnerability management processes. Tool credentials should be kept out of agentic context windows and protected from prompt-based attacks, managed by systems that support regular credential rotation, and scoped to the minimum set of permissions needed for business purposes. Component changes such as model updates, prompt revisions, and tool permission changes should be treated with the same rigor as a code change. The progression from human-in-the-loop approval to post-hoc review to full autonomy is itself a form of environmental modification, expanding or contracting the agent's operational envelope based on demonstrated performance.

(c) An AI Agent system utilizing counterparty interactions is similar in many ways to a human utilizing counterparty interactions. Humans are generally (although not universally) well intentioned, but can make mistakes, or be socially engineered even if well intentioned. These risks are managed through layered defenses, with strong audit capabilities and the potential to require escalation for approval or two-party authorization on particularly risky transactions. Traditional IAM controls, including least-privilege access, credential rotation, and centralized credential management also mitigate risk of undesired actions by agents as well as inadvertent credential exposure.

(d) Deployment environments should be monitored through observability infrastructure that captures sufficient context for real-time anomaly detection and post-hoc investigation, with that infrastructure architecturally protected from the agents it monitors, just as employees cannot edit

their own audit logs. Four levels of evaluation must operate simultaneously: behavioral testing, adversarial testing, runtime validation to catch behavioral drift in production, and regression testing after every component change.

(e) AI agent systems are already deployed at meaningful and growing scale across the open internet, in customer service automation, software development pipelines, research tools, and enterprise workflows. Characterizing that deployment as "wide," however, would probably overstate the current picture, as most production deployments remain bounded by human oversight gates or constrained tool access rather than operating in fully unbounded environments. When it comes to data collection on public networks by AI systems, site owners should have clear visibility into when an agent is accessing their site on a user's behalf, and a way to communicate how they want agents to interact with their content. AI developers, in turn, should identify their agents so they can be distinguished from malicious bots and follow guidance from site owners so they can make informed decisions about access. Tracking traffic volume over time will likely require a combination of approaches: network-level telemetry from major cloud and CDN providers along with standardized agent identity headers or signatures that could be logged at API gateways, and industry reporting mechanisms that trigger off of timing or pattern signatures analogous to how bot traffic is currently measured. The IETF working group on control of AI web behavior will likely result in reasonable standards in this area.[30]

5. Additional Considerations

(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?

(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?

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

i. Where should future research be directed in order to unlock the benefits of adoption of secure and resilient AI agent systems?

ii. Which research approaches should be prioritized to advance the scientific understanding and mitigation of security threats, risks, and vulnerabilities affecting AI agent systems?

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

(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?

Amazon response

(a) The AI agent ecosystem is still evolving, and as such it is likely premature to create a new security framework specifically for agentic systems. Extending the NIST CSF, AI RMF, SSDF, and SP 800-63 for agent-specific risks rather than creating parallel ones makes sense. Working with industry to develop best practice documents and guidance, as well as example systems illustrating those principles, could have significant practical value. The current work on agentic identity at the NIST Cybersecurity Center is a good example of this.[31]

30. See IETF, AI Preferences Working Group, https://datatracker.ietf.org/wg/aipref/about/.

31. See https://www.nccoe.nist.gov/news-insights/new-concept-paper-identity-and-authority-software-agents.

(b) Reasonable extensions of existing cyber security standards to take into account agentic AI risk factors, while remaining careful not to be too prescriptive at the early stage of a technological revolution, is the best policy approach at this time.

(c) There is a tremendous amount of both scientific research as well as practical R&D going on in the industry at this time due to industry and market forces demanding greater understanding of risks and satisfaction of emerging requirements. We do not see a need for government direction or guidance in this space for the time being.

(d) Countries around the world are currently considering how to best address a range of AI related challenges and opportunities. Given that agentic AI is an emerging and rapidly evolving field, we are supportive of national and regional approaches that leverage and extend existing security frameworks rather than building entirely new paradigms, which allows governments and companies alike to benefit from decades of established security practices and lessons learned. We also support approaches that recognize the distinct roles that providers who build the underlying AI platforms, builders who develop agentic applications and workflows, and end users who deploy and interact with these systems each have distinct but interconnected security obligations. We are seeing countries working to balance the speed of innovation with security—the most successful approaches recognize these goals are complementary, not competing, implementing safeguards that manage risk while enabling progress and access.

(e) Potentially useful fields to study to develop AI agent capabilities are human factors engineering and aviation safety design. Human factors engineering deals with humans as components that make probabilistic decisions, and while AI agents are not subject to decision fatigue, there still should be a mechanism to measure and reduce the overall error rate of the decisions made by AI agents. Aviation safety design similarly focuses on minimizing human errors, with an added approach to earned autonomy, where new aircraft systems are certified through progressive demonstration of reliability before being granted operational authority. Also, as noted above, the use of formal methods (historically considered another branch of the computer science of artificial intelligence) across multiple dimensions of the problem space is a fruitful way to enhance the security and reliability of AI systems.