Securing Multi-Agent Agentic AI Systems With Design Principles and Prioritization Framework

Date: March 7, 2026

Amazon Web Services, Inc.
410 Terry Avenue North
Seattle, WA 98109-5210

Michael Medgyessy
Head of Cloud Innovation, DoW
Amazon Web Services
medgyess@amazon.com

Table of Contents

Introduction

Agentic AI systems represent a paradigm shift from traditional AI applications. They combine foundation model reasoning with autonomous action-taking capabilities, persistent memory, and multi-agent coordination. This shift introduces novel security challenges that require purpose-built frameworks and controls.

Agentic AI systems are fundamentally different from traditional AI applications. Where conventional AI provides responses to queries, agentic AI systems reason, plan, and execute multi-step tasks autonomously. These systems combine four core facets:

• Brain: The foundation model that provides reasoning, planning, and language understanding capabilities.

• Mind: The system prompt, persona, and contextual instructions that shape agent behavior and decision-making boundaries.

• Hands: The tools, APIs, and external systems that agents invoke to take actions in the real world.

• Role/Persona: The defined function, permissions, and operational scope assigned to each agent within a system.

When multiple agents work together—coordinating through orchestrators, sharing context, and delegating tasks—the security implications multiply. Multi-agent systems introduce communication protocols between agents, shared memory stores, dynamic tool delegation, and feedback mechanisms that create attack surfaces do not present in single-agent deployments.

Traditional application security practices—while necessary—cannot fully address these challenges. Static application security testing (SAST) tools designed for traditional code are largely ineffective for LLM-based reasoning, which operates on natural language rather than programmatic logic. Traditional authentication and authorization models assume deterministic request patterns and well-defined API boundaries. Agentic systems blur these boundaries, with agents making dynamic decisions about which tools to invoke and what data to access. A defense-in-depth strategy that combines deterministic controls with AI-specific safeguards is essential.

Control effectiveness varies based on several factors: model capability and version, orchestration framework design, tool complexity and number, deployment context (internal versus external facing), and the nature of multi-agent system interactions. Organizations must account for these variables when designing their security posture and recognize that security controls require continuous reassessment as any of these factors change.

Multi-agentic systems then have ways to communicate with each other and negotiate outcomes across each agent’s motivations, and abilities enabling focus on specific functionality and knowledge. AI requires efficient underlying advanced capabilities to be present foundationally. Finally, agents need to have feedback mechanisms. If an agent is underperforming that agent needs to be corrected or replaced. Agentic teams may need a missing agent added to improve team performance. Feedback from peer agents and humans continuously improve agent performance. Humans need defined checkpoints and ability to vector long-running agentic teams in their overall goal accomplishment. An agentic team leader with a focus on team management and interface with human leadership.

For those using AWS Dedicated Cloud regions it is important to note what services are missing from those that are needed to be advocated for prioritization in availability and accreditation for use to enable secure agentic AI governance. In parallel, planning for their use for when they do become available for immediate implementation is critical. Services available are always changing so this is a starting point and call to action for customer prioritization and demand signal.

Multi-agent AI systems consist of orchestrators, specialized worker agents, shared memory, tool registries, and feedback loops. Understanding this architecture is essential for identifying security boundaries and implementing appropriate controls at each layer.

A multi-agent AI system typically consists of an orchestrator agent that coordinates work across specialized worker agents. Each agent may have access to different tools, data sources, and permissions.

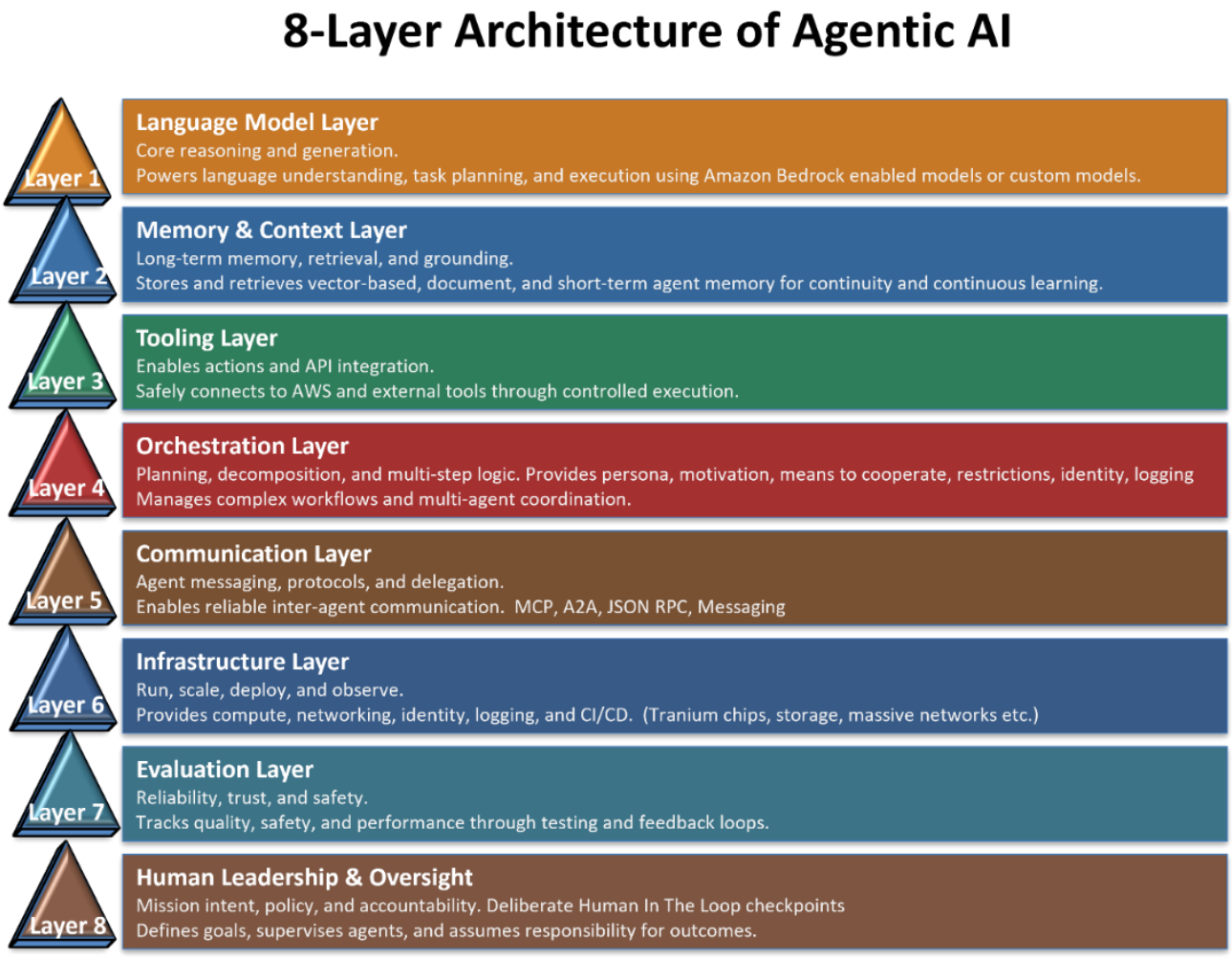

Figure 1:

The anatomy of a multi-agent system includes an orchestrator that receives user requests, decomposes them into subtasks, delegates to specialized worker agents, and synthesizes results through refinement loops. Each interaction point—between user and orchestrator,

between orchestrator and workers, between agents and tools, and between agents and memory—represents a security boundary that must be protected.

AWS provides dedicated infrastructure for hosting and managing these systems through Amazon Bedrock AgentCore, which offers isolated runtime environments (microVMs with dedicated CPU, memory, and filesystem), centralized tool management through AgentCore Gateway, identity brokering through AgentCore Identity, persistent memory management through AgentCore Memory, and comprehensive observability through AgentCore Observability.

Example multi-agentic system:

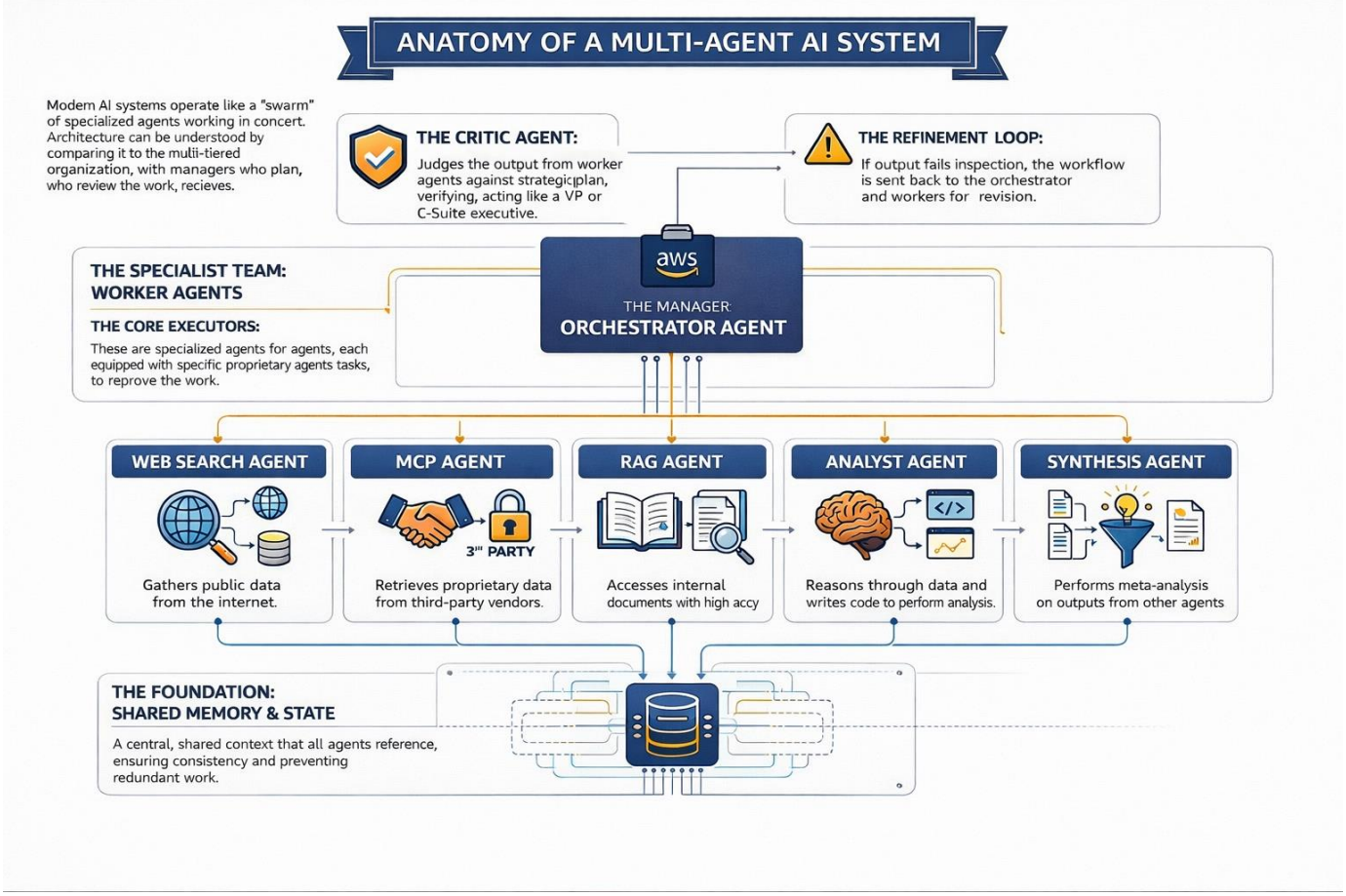

Figure 2:

Risk-Based Prioritization: The Four-Scope Framework

The Agentic AI Security Scoping Matrix provides a risk-based framework for prioritizing security investments based on agent autonomy levels. Organizations should begin with Scope 1 or 2 agents and progress to higher autonomy levels only after establishing robust security foundations.

Not all agentic AI deployments carry the same risk. The Agentic AI Security Scoping Matrix classifies agent systems into four scopes based on their level of autonomy, enabling organizations to prioritize security investments proportionally to risk.

The framework defines four distinct scopes with escalating risk profiles:

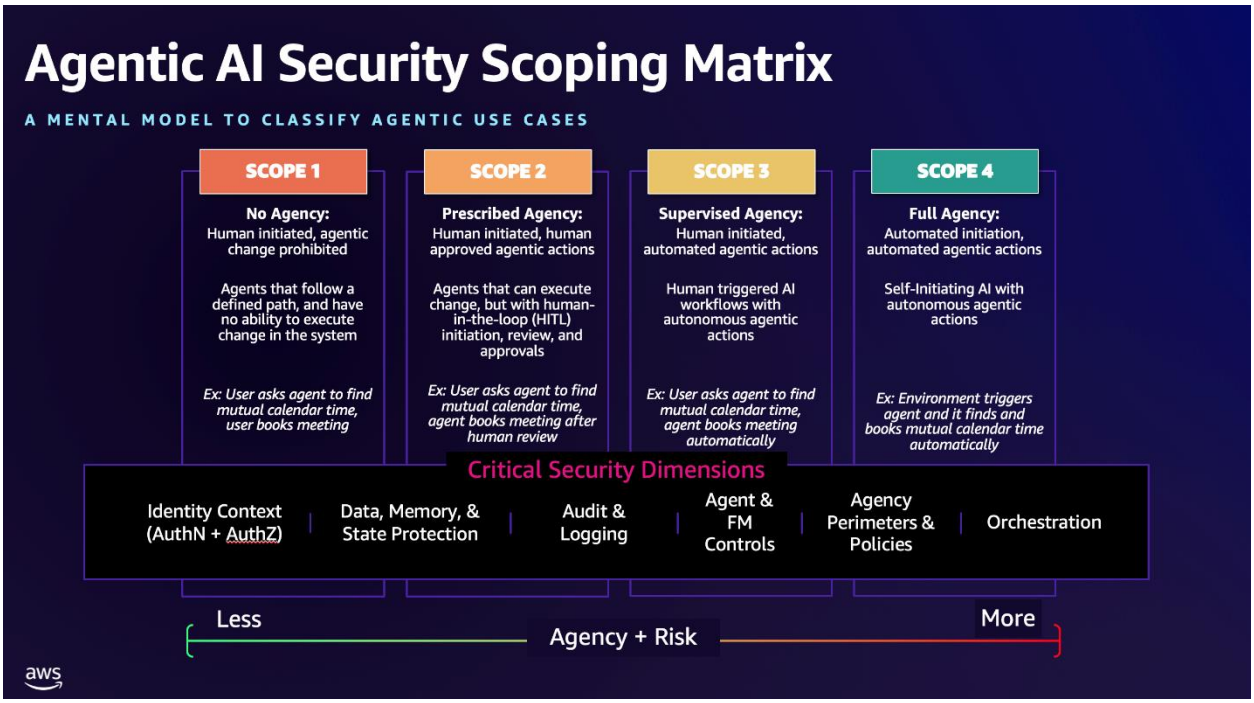

Figure 3:

Scope 1: No Agency (Lowest Risk)

- Characteristics: Human-initiated, read-only operations, predefined workflows
- Risk Level: Minimal - compromised interactions affect only specific requests
- Priority Focus: Process integrity, workflow boundary enforcement

Scope 2: Prescribed Agency (Low-Medium Risk)

- Characteristics: Human-initiated with mandatory human-in-the-loop (HITL) approval
- Risk Level: Controlled - all consequential actions require human authorization
- Priority Focus: Approval workflow security, authorization bypass prevention

Scope 3: Supervised Agency (Medium-High Risk)

- Characteristics: Human-initiated with autonomous execution
- Risk Level: Elevated - agents act independently after human trigger

- Priority Focus: Real-time monitoring, behavioral anomaly detection, scope management

Scope 4: Full Agency (Highest Risk)

- Characteristics: Self-initiating with autonomous execution
- Risk Level: Maximum - agents operate continuously without human instantiation
- Priority Focus: Advanced guardrails, continuous behavioral validation, fail-safe mechanisms

Key Insight: Organizations should start with Scope 1 or 2 implementations and progressively advance as security capabilities mature. Be cautious and selective when analyzing Scope 4 use cases, as the requirement to audit, assess, validate, and implement complex security controls is much higher.

AWS provides agents that are curated constantly enabling usage without the overhead of managing the lifecycle of the agent. Agents such as the policy agent and security agent on a team will enable security is involved from cradle to grave, and policy agents enforcing the policy requirements from government-imposed controls to mission owner designated guardrails on functionality. A user interface specific agent helps focus on communication of overall project management and reporting in and out of the agentic team to human leadership.

Before implementing specific security controls, organizations must adopt five foundational design principles that guide all agentic AI security implementations. These principles provide the philosophical foundation for building secure, trustworthy, and production-ready multi-agent systems. These principals lay the groundwork for agentic systems to securely move into high impact production capabilities with a routine process and framework which can continue to advance over time.

Foundational Design Principles

Five foundational design principles guide all agentic AI security implementations, regardless of scope level. These principles establish the philosophical foundation for secure agentic AI and should inform every architectural and operational decision.

Design Principle 1: Implement Progressive Autonomy with Bounded Agency

Start with minimal autonomy and agency, then gradually expand capabilities as security controls mature. Define clear boundaries for what actions agents can perform and what systems they can access, implementing defense-in-depth controls through network, application, agent, and data layers to ensure compromise at one level doesn't lead to complete system failure.

Key Implementation Requirements:

- Begin with Scope 1 or 2 implementations (read-only or human-approved actions)
- Establish clear operational boundaries before expanding autonomy
- Implement layered security controls at network, application, agent, and data layers
- Ensure single-point failures cannot compromise entire system
- Document and enforce agency boundaries for each agent deployment

Design Principle 2: Establish Continuous Monitoring and Behavioral Validation

Security should not be a boundary problem but rather handled through continuous monitoring by implementing behavioral analysis, anomaly detection, and audit trails. Monitor agent actions

during autonomous execution phases to detect unauthorized operations, scope creep, and deviations from intended behavior before they can cause significant impact.

Key Implementation Requirements:

- Deploy real-time behavioral monitoring from day one
- Implement anomaly detection for agent decision patterns
- Establish baseline behaviors for normal agent operations
- Create automated alerting for deviations from expected behavior
- Monitor for scope creep during autonomous execution phases

Design Principle 3: Maintain Human Oversight Through Strategic Intervention Points

Even in highly autonomous systems, preserve relevant human control through approval workflows or override capabilities. Design systems to automatically reduce autonomy levels when security events are detected (graceful degradation), and ensure humans can validate agent alignment with organizational objectives.

Key Implementation Requirements:

- Establish strategic human intervention points appropriate to autonomy level
- Implement graceful degradation mechanisms that reduce autonomy when anomalies detected
- Maintain tamper-proof human override capabilities
- Design approval workflows that don't overwhelm human reviewers
- Ensure humans can validate agent alignment with organizational goals

Design Principle 4: Secure Identity, Authorization, and Tool Access

Implement identity management for both human and machine actors, with authentication, authorization, and delegation mechanisms for autonomous actions. Apply least-privilege principles to tool access, API permissions, and other integrations, reducing the risk of agents escalating privileges or accessing resources beyond their defined scope.

Key Implementation Requirements:

- Implement separate identity models for agents and human users
- Apply least-privilege principles to all tool access and API permissions
- Use temporary credentials with just-in-time access patterns
- Prevent confused deputy problems where lesser-privileged entities elevate permissions through agents
- Conduct regular permission audits and access reviews

Design Principle 5: Design for Explainability and Auditability

Build transparency into agent decision-making processes by logging actions and rationale. Maintain audit trails showing actual versus expected execution paths and provide visibility into tool orchestration and multi-agent interactions to support incident investigation and response.

Key Implementation Requirements:

- Log all agent decisions, actions, and reasoning paths to immutable storage
- Capture complete decision artifacts including prompts, model responses, and tool calls
  Implement distributed tracing for multi-agent interactions

- Maintain correlation identifiers linking related actions across agents
- Store decision context sufficient to reconstruct agent reasoning processes

Critical Security Dimensions: Prioritized Implementation

Six security dimensions provide the operational framework for implementing agentic AI security. Each dimension has progressive requirements based on the agent's autonomy scope level, with Identity Context as the highest priority across all scopes.

The following six security dimensions, prioritized by implementation urgency, provide the operational framework for securing agentic AI systems. Each dimension maps to specific AWS services and has progressive requirements that increase with agent autonomy level.

1) Identity Context (Authentication & Authorization) - Highest Priority

Why This Comes First: Identity and authorization concerns must be addressed for both machines and humans to prevent the "confused deputy problem" where lesser-privileged entities elevate permissions through agents. This dimension directly implements Design Principle 4 (Secure Identity, Authorization, and Tool Access).

AWS Services:

- Amazon Bedrock AgentCore Identity: Agent-specific identity and credential management
- AWS IAM: Human administrator identity with least-privilege roles with dynamic permission boundaries
- Amazon Cognito: Customer identity and M2M authentication
- AWS Certificate Manager: Automates certificate lifecycle for certificate-based authentication for agent-to-agent communications

Implementation Priority by Scope:

- Scope 1-2: User authentication, service authentication, limited read-only permissions
- Scope 3: Add agent authentication, identity delegation for autonomous actions
- Scope 4: Dynamic identity lifecycle, federated authentication, cross-agent identity verification

Key Requirements:

- Implement separate permission models for agents and human users
- Use temporary credentials with just-in-time access patterns
- Enforce certificate-based authentication for agent-to-agent communications
- Conduct regular permission audits using AWS IAM Access Analyzer

2) Threat Modeling & Risk Assessment - Critical Foundation

Why This Is Essential: Threat modeling should be a high priority to directly align security controls with zero-trust principles.

Prioritization Approach (aligned with NIST AI Profile):

- Priority 1 (High): Controls addressing the most critical challenges - implement immediately

- Priority 2 (Moderate): Next priority after high-priority controls
- Priority 3 (Foundational): Important but may not require same urgency

Key Risk Vectors to Prioritize:
Cascading compromises in multi-agent workflows where single agent breach propagates
Memory poisoning attacks corrupting decision-making across interactions
Unauthorized capability expansion in autonomous agents
Data exfiltration through external connectivity
Uncontrolled operations from self-directed behavior

3) Audit & Logging - Immediate Deployment Requirement

This dimension directly implements Design Principle 5 (Design for Explainability and Auditability) and supports Design Principle 2 (Continuous Monitoring and Behavioral Validation).

AWS Services:
- AWS CloudTrail: Immutable audit trails with log file validation
- Amazon CloudWatch Logs: Comprehensive logging with encryption
- Amazon S3 with Object Lock: Long-term logs in compliance mode*
- Amazon Bedrock AgentCore Observability: Agent reasoning chains and decisions
- AWS X-Ray: Distributed tracing across agent interactions

Progressive Requirements:
- Scope 1: Local activity logs, change tracking, integrity monitoring
- Scope 2: Human decision audit trails, agent recommendation logging
- Scope 3: Comprehensive action logging, reasoning chain capture, extended session tracking
- Scope 4: Behavioral analysis, anomaly detection, automated incident correlation

Key Requirements:
- Log all agent decisions, actions, and reasoning paths to off-app storage
- Use write-once storage with cryptographic signing to prevent tampering Maintain correlation identifiers to link related actions across multiple agents
- Store decision artifacts including prompts, model responses, and tool calls

4) Data, Memory, & State Protection

AWS Services:
Amazon Bedrock AgentCore Memory: Isolated short-term and long-term memory
AWS KMS: Encryption for messages and signing keys
Amazon SQS: Server-side encryption for agent messaging

Escalating Requirements:
Scope 1: Local resource permissions, file system access controls
Scope 2: Role-based access control, read-mostly permissions
Scope 3: Context-aware authorization, just-in-time privilege elevation
Scope 4: Dynamic permission boundaries, continuous validation

Key Requirements:
Isolate agent memory between sessions and users to prevent cross-contamination
Validate and sanitize all memory inputs before storage

Monitor for hallucination propagation across agent interactions

- Implement integrity checks and versioning for stored memories

5) Agent & Foundation Model Controls

AWS Services:

- Amazon Bedrock Guardrails: Operational boundaries, content filters, denied topics
- AWS WAF: Malicious request filtering
- Amazon EventBridge: Guardrail violation alerts

AWS Lambda: Timeout limits and memory constraints

Implementation Priorities:

- Scope 1: Process isolation, input/output validation, basic guardrails
- Scope 2: Approval gateway enforcement, extended session management
- Scope 3: Container isolation, long-running process monitoring, tool invocation sandboxing
- Scope 4: Behavioral analysis, anomaly detection, automated containment

Key Requirements:

- Implement constitutional AI principles with multiple validation layers
- When applicable configure content filtering to block PII, profanity, and sensitive data
- Use automated reasoning to validate tool inputs and outputs
- Implement human-in-the-loop for critical decisions using AWS Step Functions

6) Agency Perimeters & Policies

This dimension directly implements Design Principle 1 (Progressive Autonomy with Bounded Agency).

AWS Services:

- Amazon Bedrock AgentCore Gateway: Fine-grained tool access control
- AWS Organizations with Service Control Policy (SCPs): Maximum permission boundaries
- AWS Config: Permission change detection

Progressive Controls:

- Scope 1: Fixed execution boundaries, hard-coded constraints
- Scope 2: Approval-based boundary modification, human-validated constraints
- Scope 3: Dynamic boundary adjustment, runtime constraint evaluation
- Scope 4: Self-securing boundaries, automated safety checks, graceful degradation

Key Requirements:

- Maintain an approved tool registry with security assessments
- Implement role-based access control (RBAC) for tool usage
- Define clear execution boundaries and resource quotas
- Use policy-as-code for automated governance enforcement

By breaking down the complexity into priorities and dimensions as applied in context of scope of the agentic AI the roadmap becomes easily contextualized for action. Multi-agentic AI systems are being deployed currently, and security and governance must be applied in as rapid a pace as the capability so technical debt in rework of these systems can be reduced. Artificially limiting use introduces as much risk to mission if not more as the world continues to evolve. It is imperative to put as much priority in laying the framework first for long-term production success past pilots and build a organizational structure which empowers and automates the framework as it continuously evolves as well.

DEFENSE-IN-DEPTH: AWS SECURITY ARCHITECTURE

Multi-Agent Orchestration - Specialized Priority

AWS Services:

- AWS Step Functions: Multi-agent workflow coordination with state validation
- amazon SQS: Secure asynchronous messaging
- AWS PrivateLink: Secure agent-service communication
- Amazon GuardDuty: Unusual API call pattern detection

Critical Requirements:

- Encrypt and authenticate all inter-agent communications using TLS 1.3
- Establish trust boundaries between agents based on roles and risk profiles
- Implement circuit breakers to prevent cascading failures
- Monitor coordination patterns for anomalies indicating compromised agents

Progressive Autonomy Deployment Strategy

This strategy directly implements Design Principle 1 (Progressive Autonomy with Bounded Agency).

Recommended Implementation Path:

Phase 1: Foundation (Months 1-3)

• Deploy Scope 1 agents with read-only capabilities
• Establish identity management and audit logging
• Build security monitoring baseline
• Implement all five design principles at foundational level

Phase 2: Controlled Expansion (Months 4-6)

• Introduce Scope 2 agents with HITL approval workflows
• Implement comprehensive approval audit trails
• Train human approvers on agent capabilities
• Strengthen continuous monitoring capabilities

Phase 3: Supervised Autonomy (Months 7-12)

• Deploy Scope 3 agents for specific use cases
• Implement real-time behavioral monitoring
• Establish automated kill switches and intervention mechanisms
• Deploy graceful degradation mechanisms

Phase 4: Full Autonomy (Year 2+)

• Selectively deploy Scope 4 agents for proven use cases
• Implement advanced AI safety techniques and reward modeling
• Maintain tamper-proof human override mechanisms
• Ensure all five design principles are fully operational at highest maturity level

Critical Success Factor: Human oversight doesn't reduce from Scope 1 to 4—it shifts focus. While instantiation and approval requirements decrease, the need to audit, assess, validate, and implement complex security controls increases dramatically in Scopes 3 and 4. This aligns with Design Principle 3 (Maintain Human Oversight Through Strategic Intervention Points).

Gaps Technology Does Not Fully Address

While AWS provides comprehensive services, it is recommended that Chief AI Officers utilize AWS ProServe consultants which can embed in your organization to help fill in the below gaps alongside partners to ensure all the aspects are in fact covered.

Agent Lifecycle Management

Gap: Central agent registry for discovery, reuse, and governance
Chief AI Officer Responsibility: Implement internal "app store" for AI agents with version control, credential management, and approval workflows

Simulation Testing

Gap: Statistical evaluation of agent behavior across synthetic scenarios
Solution: Move beyond unit tests to simulation testing integrated into agent CI/CD pipelines

Rainbow Deployments

Gap: Safe rollout strategies for stateful, long-running agents
Solution: Implement gradual traffic shifting with easy rollback capabilities

Outcome-Based Evaluation

Gap: Quality assessment beyond deterministic testing
Solution: Implement LLM-as-a-judge setups combined with human annotations

Cost & Latency Monitoring

Gap: Real-time financial and performance oversight
Solution: Deploy dashboards tracking token consumption, API utilization, and latency per agent, what is the Return on Investment for each agent added to a team and how to determine this.

Conclusion

Securing production ready multi-agent agentic AI systems requires a risk-based, progressive approach that matches security investment to autonomy levels. The companion paper to this one is Establishing an AI Cybersecurity Organization. It is important to have the organizational structure alongside the technology together for Chief AI Officer roadmap success.
Agentic AI scoping from Scope 1 to Scope 4 represents a fundamental shift in AI security—each scope requires specific capabilities that organizations must build systematically to support their

agentic ambitions safely. The five design principles provide the guiding philosophy that ensures security controls remain aligned with organizational objectives throughout this progression. Long-running agentic AI teams are only getting more capable, and leaders will be provided with a new challenge in understanding leadership and management of human and agentic workforces together. Agentic AI represents both a transformative opportunity and a significant security challenge for enterprises. The framework presented in this document provides a structured path forward, enabling organizations to capture the value of autonomous AI systems while managing risk appropriately. How this evolves over time is an exciting journey but one which cannot wait to begin. The strategic thought leadership and guidance is critical for an organization to be able to have as not only a guiding light but one to collaborate on evolving together.

APPENDIX 1:
Next Steps

Immediate Actions:

- Assess current AI agent deployments against the four-scope framework
- Evaluate current implementation maturity against the five design principles
- Identify gaps in identity management and audit logging capabilities
- Establish governance committee for scope progression decisions
- Develop 12-month roadmap for progressive autonomy deployment

Strategic Investments:

- Build internal agent lifecycle management platform
- Implement simulation testing infrastructure
- Deploy comprehensive monitoring and anomaly detection systems
- Establish multi-agent coordination protocols and standards
- Embed all five design principles into organizational security culture

APPENDIX 2:
Design Principles Implementation Checklist

Use this checklist to ensure all five design principles are properly implemented:

Design Principle 1 - Progressive Autonomy with Bounded Agency:
☐ Started with minimal autonomy (Scope 1 or 2)
☐ Defined clear agency boundaries for each agent
☐ Implemented defense-in-depth across network, application, agent, and data layers
☐ Documented progression criteria for advancing through scopes

Design Principle 2 - Continuous Monitoring and Behavioral Validation:
☐ Deployed real-time behavioral monitoring
☐ Implemented anomaly detection systems
☐ Established baseline behaviors for normal operations
☐ Created automated alerting for deviations

Design Principle 3 - Human Oversight Through Strategic Intervention Points:
☐ Established appropriate human intervention points
☐ Implemented graceful degradation mechanisms
☐ Maintained tamper-proof human override capabilities
☐ Designed approval workflows that prevent cognitive overload

Design Principle 4 - Secure Identity, Authorization, and Tool Access:
☐ Implemented separate identity models for agents and humans
☐ Applied least-privilege principles to all tool access
☐ Used temporary credentials with just-in-time access
☐ Conducted regular permission audits

Design Principle 5 - Explainability and Auditability:
☐ Logged all agent decisions and reasoning paths
☐ Captured complete decision artifacts
☐ Implemented distributed tracing for multi-agent interactions
☐ Maintained audit trails showing actual vs. expected execution paths

APPENDIX 3:
Securing Model Context Protocol (MCP) as described by Anthropic

MCP has become the standard for AI models to communicate with the non-AI cyber world around them. This standard of communication brings considerations for security as well in which best practices are still emerging every day. Below is information from Anthropic, the creator of the MCP standard, to consider securing your implementations of MCP as well.

The Model Context Protocol (MCP) represents a fundamental shift in how AI agents interact with external tools and data sources. This security framework addresses the unique challenges of securing AI-mediated systems where large language models make security-critical decisions based on natural language inputs. Organizations implementing agentic AI must understand that traditional security approaches are insufficient—MCP introduces 12 core threat categories encompassing nearly 40 distinct threats that require coordinated defense strategies across protocol, application, and organizational layers.

The Three-Tier Threat Model

Tier 1: MCP-Specific Threats (Novel Risks)
These seven threats are unique to MCP architecture and represent entirely new attack surfaces:

- Identity Spoofing: Weak authentication mechanisms allow attackers to impersonate legitimate clients or agents, gaining unauthorized access to tools and data sources.
- Tool Poisoning: Malicious modification of tool metadata causes the LLM to invoke compromised tools with incorrect parameters or dangerous configurations.
- Full Schema Poisoning: Attackers compromise entire tool schema definitions at the structural level, fundamentally altering how the AI understands available capabilities.
- Resource Content Poisoning: Malicious instructions embedded in backend data sources (databases, file systems) manipulate LLM behavior when retrieved and processed.
- Typosquatting/Confusion Attacks: Similar-named malicious servers or tools exploit naming conventions to trick users and systems into connecting to compromised endpoints.
- Shadow MCP Servers: Unauthorized, unmonitored server instances operate outside governance frameworks, creating security blind spots.
- Overreliance on LLM: Organizations incorrectly assume LLMs will invoke tools safely despite known vulnerabilities, failing to implement necessary guardrails.

Tier 2: MCP-Contextualized Threats - Eight conventional threats significantly amplified by AI mediation:

- 1. Privilege Escalation: Exploitation of authentication/authorization flaws, including confused deputy attacks where the AI is tricked into performing privileged operations on behalf of unauthorized users.
- Insecure Human-in-the-Loop: Missing or insufficient consent checks allow autonomous 2. AI actions without appropriate human oversight for sensitive operations.
- Improper Multitenancy: Weak tenant isolation in cloud deployments leads to data leakage between organizations sharing infrastructure.
- 3. Prompt Injection: Malicious inputs manipulate LLM behavior, causing it to ignore instructions, leak data, or perform unauthorized actions.
- 4. Man-in-the-Middle (MITM): Insecure transport layers enable interception and modification of communications between MCP components.

- 5 Consent/Approval Fatigue: Excessive security prompts condition users to blindly approve requests without proper review.
- 6. Resource Exhaustion/Denial of Wallet: Excessive API calls cause cost overruns or denial-of-service conditions.
- 7. Invisible Agent Activity: Covert malicious operations occur without user awareness or audit trails.

Tier 3: Conventional Threats

Traditional security threats include credential theft, command injection, file system exposure, supply chain compromise, insufficient logging, CSRF, CORS bypass, and others that remain relevant in MCP contexts.

Strategic Recommendations for AI Security Leaders related to MCP

Immediate Actions:

- Conduct inventory of all MCP deployments and shadow servers
- Implement authentication and authorization for all MCP connections
- Deploy logging and monitoring across MCP infrastructure
- Establish allowlists of approved MCP servers
- Implement sandboxing for all MCP server executions

Short-Term Initiatives:

- Deploy cryptographic verification for all MCP components
- Implement human-in-the-loop controls for sensitive operations
- Establish centralized governance framework
- Conduct security training for development teams
- Implement prompt injection detection systems

Long-Term Strategy:

- Implement comprehensive AI supply chain security program

Additional Information:

Link to the latest AWS AI Security Reference Architecture
https://docs.aws.amazon.com/prescriptive-guidance/latest/agentic-ai-security/introduction.html

For builders there is also AWS AI Well Architected Tool support