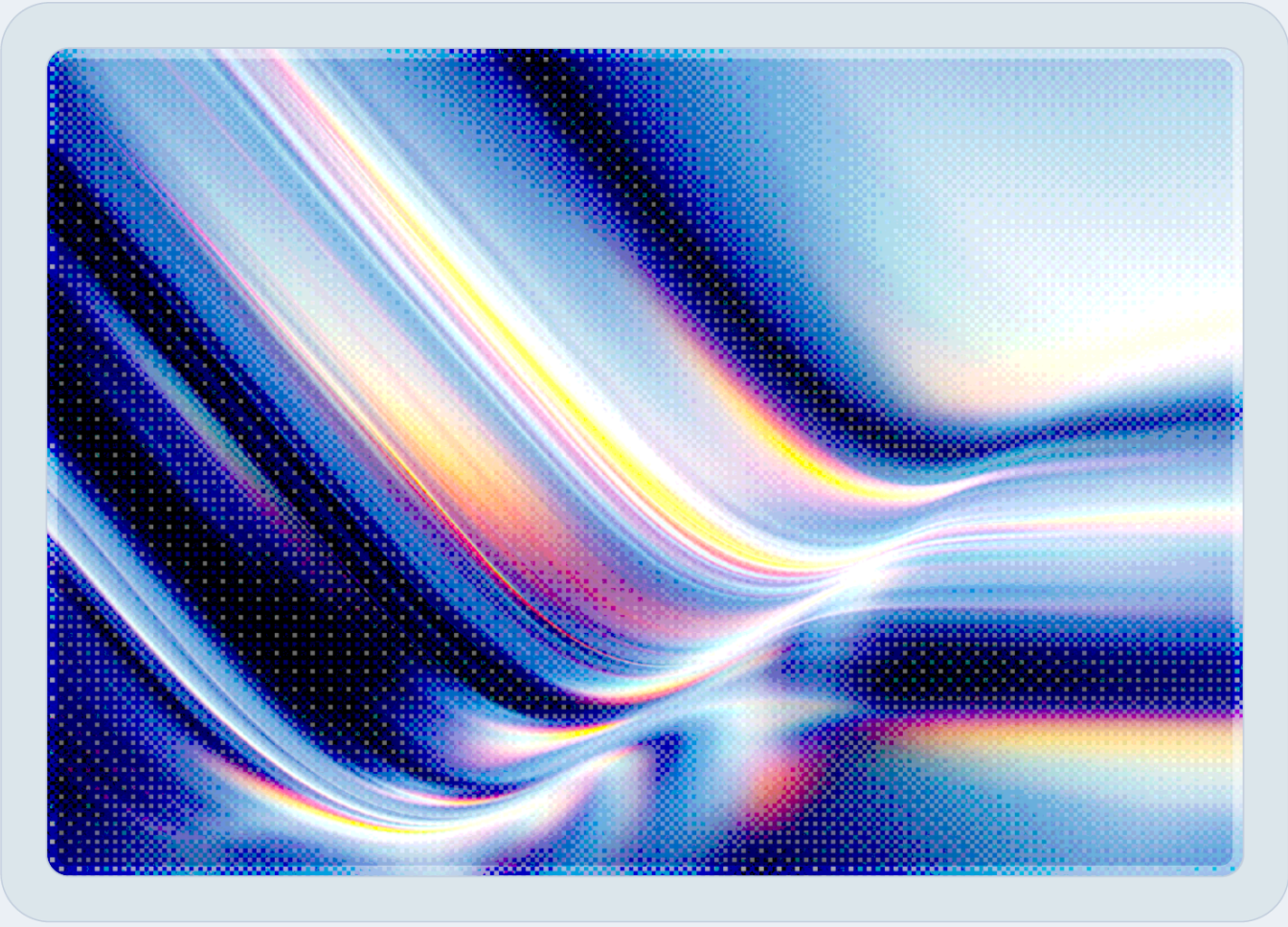
CSAI Foundation | Cloud Security Alliance

Cross-Agent Privilege Escalation in Agentic AI

The Hidden Attack Surface in Multi-Agent Workflows

2026-05-03

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Table of Contents

Executive Summary

The transition from single AI agents to coordinated networks of multiple autonomous agents is one of the most consequential architectural shifts in enterprise technology today. Security programs, however, have not kept pace. While organizations focus on preventing prompt injection and protecting the model layer, a different category of risk has been maturing in the space between agents: the exploitation of inter-agent trust relationships to achieve privilege escalation that neither the attacking agent nor any individual component would achieve alone.

Cross-agent privilege escalation describes a class of attack in which an AI agent — whether compromised by an adversary, manipulated through prompt injection, or behaving outside its intended scope — exploits the trust, credentials, permissions, or authority delegated to it by an orchestrating agent or principal to take actions it was not independently authorized to perform [16]. Unlike traditional privilege escalation, which typically exploits software vulnerabilities or misconfigured access controls, this class of attack exploits the architectural design choices of multi-agent systems: the way agents inherit permissions from their orchestrators, delegate authority to subagents, share context windows, and communicate through channels that carry no cryptographic guarantees of identity or integrity.

The empirical picture is concerning. A CSA survey conducted in January 2026 found that 65 percent of organizations reported at least one AI agent security incident in the prior twelve months, with 61 percent of those incidents involving data exposure or mishandling and 43 percent resulting in operational disruption [1]. A separate CSA survey on agentic identity found that only 18 percent of organizations reported high confidence in their identity and access management systems' ability to govern AI agents [2], and that 44 percent of organizations plan to rely on static API keys as their primary agent authentication mechanism — a credential model with documented, well-understood weaknesses [2]. The infrastructure for governing who agents are, what they are permitted to do, and when their permissions should be revoked does not yet exist at the scale and maturity that multi-agent deployment demands.

This paper examines the structural mechanics of cross-agent privilege escalation, traces the attack paths through which it manifests in current orchestration frameworks, documents the CVEs and research findings that have demonstrated its real-world exploitability, and provides actionable guidance for organizations seeking to build multi-agent architectures that resist this emerging threat class. The analysis is grounded in CSA's MAESTRO agentic threat model, the AI Controls Matrix, and established Zero Trust principles, connecting technical findings to the governance frameworks that enterprise security programs can implement today.

Introduction and Background

The Multi-Agent AI Revolution

Enterprise AI deployment has moved through several distinct phases in rapid succession. The first phase was the deployment of standalone AI models — large language models integrated into specific applications to perform bounded tasks such as summarization, code generation, or customer service. The second phase introduced AI agents: LLM-based systems equipped with tools that allow them to take actions in the world, access external data, execute code, browse the web, call APIs, and interact with enterprise systems. The third phase, which organizations are currently entering at scale, is the deployment of multi-agent systems: coordinated networks of specialized agents that communicate with each other, delegate tasks, spawn subagents, and collectively accomplish goals that would require complex multi-step reasoning and action across many systems.

This progression has been rapid and often unplanned. A CSA survey conducted in January 2026 found that 40 percent of organizations already had AI agents in production and that more than 70 percent expected to manage dozens to hundreds of agents within the following twelve months [2]. The scale of agent deployment is growing faster than organizations' capacity to govern it: the same survey found that only 23 percent of organizations had a formal, organization-wide agent governance strategy in place [2]. The remaining 77 percent are deploying multi-agent systems without a coherent framework for defining what those agents are authorized to do, how those authorizations are verified at runtime, or how they are revoked when no longer needed.

The practical consequence is that agents are operating in environments where the question of "what is this agent permitted to do, and how do we know?" is answered informally, inconsistently, or not at all. An orchestration agent may spawn a subagent, pass it a task description and a set of credentials, and trust that the subagent will operate within the intended scope — not because any technical mechanism enforces that scope, but because the orchestrator's prompt described it. A subagent receiving instructions from what it believes to be a trusted orchestrator may execute those instructions without any cryptographic verification that the instructions actually came from where they claim to have originated. These are not edge cases. They are the architectural baseline from which most current multi-agent systems operate.

Why Conventional Security Models Fail

The security models that govern traditional software systems were not designed for the properties that make AI agents distinctive: natural language instruction, context window inheritance, non-deterministic behavior, tool use, and the ability to operate autonomously across extended action sequences. Each of these properties creates attack surface that conventional security controls address poorly or not at all.

In traditional software, privilege escalation typically requires exploiting a specific technical vulnerability — a buffer overflow, a misconfigured permission, a token with excessive scope. The attack path is deterministic and, once discovered, the vulnerability can be patched. AI agents introduce a fundamentally different model: an agent's behavior is determined by a combination of its system prompt, its training, the instructions it receives at runtime, and the outputs of the tools it calls — none of which are signed, all of which can be influenced by an adversary with access to any point in the agent's input chain. The attack surface is not a specific line of code that can be patched; it is the entire surface of the agent's operating environment, including data it reads, messages it receives from other agents, and tool outputs it processes.

Conventional identity and access management (IAM) provides governance through verified identities with defined roles and permissions. When a human user authenticates, their identity is verified by an identity provider through cryptographic mechanisms, and their permissions flow from that verified identity. When an agent authenticates, the current norm is far more fragile. The majority of agent deployments use static API keys as their primary credential — tokens that can be copied, stolen, shared between agents, or extracted from a compromised orchestration context without any authentication ceremony that ties the credential to a specific, verifiable agent identity [2]. An agent that declares itself to be the orchestrator, the system administrator, or a trusted peer can, in most current deployments, make that claim through nothing stronger than the assertion itself — current standard practice provides no cryptographic verification ceremony for runtime inter-agent identity claims.

The research community has characterized this gap clearly. The problem of authenticated delegation — ensuring that when agent A grants authority to agent B, agent B both holds and can verifiably present that grant — was identified in an analysis published in early 2025 as one of the foundational unresolved problems in multi-agent security [3]. When that delegation cannot be cryptographically verified, the resulting trust relationships are vulnerable to spoofing, injection, and cascade exploitation.

Section 1: The Mechanics of Cross-Agent Privilege Escalation

Trust Inheritance in Agent Hierarchies

Multi-agent systems are typically organized hierarchically, with an orchestrator agent at the top directing specialized subagents to execute components of a broader task. This architecture is efficient: the orchestrator can decompose complex objectives into parallelizable subtasks, invoke specialist agents with appropriate capabilities, and aggregate results. But efficiency and security pull in opposite directions when it comes to permission inheritance. The orchestrator typically holds the highest privilege level in the system — it has credentials for enterprise systems, API tokens for external services, and access to sensitive context about the overall task. When it delegates to subagents, it must pass some portion of that context and those credentials to enable the subagents to function.

The mechanism by which this delegation occurs most commonly takes the same form: the orchestrator embeds credentials, context, or elevated instructions in the prompt or message it sends to the subagent. That prompt is processed by the subagent as input, not as a verified authentication event. An adversary who can inject into the subagent's input stream — through prompt injection, tool output manipulation, or retrieval-augmented generation (RAG) poisoning — can send the subagent instructions that appear to come from the trusted orchestrator, carrying the same implicit authority as legitimate orchestrator instructions. From the subagent's perspective, instructions from a legitimate orchestrator and instructions from an attacker impersonating that orchestrator are often indistinguishable.

Researchers analyzing the formal properties of multi-agent trust established in 2025 and 2026 that this architectural property creates an inherent privilege escalation path whenever an agent processes instructions from untrusted sources while simultaneously holding access to privileged systems or credentials [4]. The attack does not require exploiting a software vulnerability. It requires only that the attacker's instructions reach the agent through a pathway the agent treats as trustworthy — which, in current architectures, includes any data source the agent has access to, any tool output it processes, and any message that arrives in its context window bearing the formatting of a legitimate orchestrator message.

The Confused Deputy Pattern in Multi-Agent Systems

The confused deputy problem is a well-established concept in computer security, dating to the 1970s: it describes a situation in which a program with elevated privileges is tricked into misusing those privileges on behalf of an attacker who does not hold them directly. In classical computing, the confused deputy typically

emerges when a trusted service performs operations on behalf of arbitrary clients without adequately distinguishing between the service's own authority and the authority it should exercise on behalf of each client. In multi-agent AI systems, the confused deputy pattern recurs with a new mechanism and substantially greater potential for harm.

An AI agent acting as an orchestrator is, by construction, a privileged principal: it holds credentials, tools, and system access that subagents may not possess individually. When an adversary manipulates a subagent into sending the orchestrator a crafted instruction — through prompt injection in data the subagent has retrieved, through a malicious tool output, or through direct message injection — the orchestrator may execute that instruction using its elevated privileges. The attacker never directly accessed the orchestrator. They manipulated a less-privileged agent in the trust chain to do so on their behalf. The orchestrator, acting in good faith, becomes the confused deputy.

CSA's research program has identified the confused deputy attack as a priority threat in agentic AI environments, noting that it exploits not a vulnerability in any single component but a structural property of systems in which trusted agents process untrusted data [5]. The attack is particularly difficult to detect because the orchestrator's actions appear legitimate: it is using its authorized credentials, operating within its defined role, and taking actions that look exactly like the actions it was designed to take. The malicious instruction that triggered those actions was processed several steps earlier in the pipeline, often by an agent that itself showed no anomalous behavior. Attribution requires tracing the full action chain backward from the harmful outcome to the point of injection — a capability that few current multi-agent monitoring systems provide.

Role Elevation Through Delegation Chains

Beyond individual confused deputy attacks, multi-agent systems are vulnerable to a more systematic form of privilege escalation that researchers have termed role elevation through delegation chains. In this pattern, an attacker does not need to compromise a high-privilege orchestrator directly. Instead, they manipulate a series of agents, each of which has modest privileges, in a sequence that cumulatively achieves escalated access — much as a lateral movement campaign in traditional enterprise networks can traverse multiple intermediate systems to reach a high-value target.

The mechanics of this escalation follow from the trust model that most multi-agent frameworks implement. When agent A spawns agent B and delegates a task, agent B inherits some portion of agent A's context and credentials. When agent B spawns agent C for a subtask, agent C may inherit from both. In a sufficiently complex agent network, the transitive delegation graph can create privilege paths that no individual authorization decision was intended to enable. An agent at the edge of the network, interacting with external data sources, may be several delegation steps removed from high-privilege core systems — but if the delegation chain is not explicitly bounded, a sufficiently motivated adversary can potentially traverse it.

Formal analysis of this problem, published in 2025 as part of a comprehensive threat model for agentic AI systems [6], characterized delegation chain exploitation as a "reasoning-layer attack" distinct from the code-layer vulnerabilities that conventional security testing addresses. Because the escalation path traverses semantic relationships between agents — which agent trusts which, what authority each agent believes it holds, which instructions each agent will act upon without further verification — it does not produce the technical signatures that intrusion detection systems are designed to recognize. An agent exercising delegated authority that was transitively obtained through a chain of manipulated delegations looks, from a logs and access-pattern perspective, exactly like an agent exercising legitimate authority.

Section 2: The Agentic Identity Gap

Current State: Credential Chaos

The foundational technical precondition for preventing cross-agent privilege escalation is the ability to verify which agent is claiming to hold which authority. Without verified agentic identity, every instruction an agent receives from a purported orchestrator is ultimately an unauthenticated claim: "I am your orchestrator; do this." Current enterprise deployments overwhelmingly fail this basic test.

The CSA 2025 Agentic Identity Survey, conducted among 285 respondents in September–October 2025, documented the credential landscape in detail [2]. Among organizations with agents in production, 44 percent planned to use static API keys as their primary agent authentication mechanism; 43 percent relied on username and password credentials; and 35 percent used shared service accounts [2]. These are not the weakest available mechanisms by accident. They reflect the fact that modern identity frameworks — including OIDC and OAuth 2.0, the protocols that govern human and application identity in most enterprises — were not designed with autonomous agents as a principal type. The protocols assume that a human is present either to authenticate or to authorize a delegation. When an agent spawns a subagent at runtime, the parent agent is acting autonomously, and there is no human present to approve the delegation through a standard authorization ceremony.

The consequence of this credential gap extends beyond the authentication layer. Agents using static API keys cannot be bound to a specific task scope or time window at the cryptographic level. A key issued for one purpose can be used, if it is exfiltrated or passed to an unintended recipient, for any purpose within its scope — potentially for as long as the key remains valid. The OWASP Top 10 for Agentic Applications, published in 2026, identifies this pattern explicitly: agents often inherit user or system identities including high-privilege credentials and session tokens, and those inherited credentials can be exploited if the agent is compromised or manipulated [7]. The static credential model does not bound the damage that any single compromised agent can cause, because the credential carries the same authority regardless of which agent holds it.

The organizational picture compounds the technical problem. Only 39 percent of organizations assign clear ownership of agent identity to a security function; 32 percent assign it to general IT operations; and 13 percent have created a dedicated AI Security function [2]. In the remaining organizations, agent identity ownership is ambiguous, distributed, or unassigned. When no one owns the problem, credential rotation does not happen on any consistent schedule, compromised agents are not deprovisioned promptly, and the inventory of what agents exist — let alone what credentials they hold — is incomplete. The CSA January 2026

survey found that 82 percent of organizations had discovered previously unknown agents in their environment in the prior twelve months [1], which suggests that the premise of governing agent identity requires first solving the more basic problem of knowing what agents are deployed.

Emerging Standards and Their Current Gaps

The identity standards community has recognized the agentic identity problem and begun addressing it. The OpenID Foundation's October 2025 white paper on Identity Management for Agentic AI proposed several extensions to existing protocols that would make agent authentication more robust: a proposed OIDC-A (OpenID Connect for Agents) specification that includes agent attestation information and agent capability endpoints; requirements for strong asymmetric authentication methods such as JWT Client Authentication or mutual TLS; and a Just-in-Time Provisioning model that would create agent identities on demand with embedded attributes including time-to-live, purpose, risk level, and delegation context [8]. These proposals directly address the agentic authentication gap, but they remain at the proposal stage. Standardization timelines in the identity space are measured in years, not months.

NIST's National Cybersecurity Center of Excellence published a concept paper in February 2026 that identified the absence of standard mechanisms for carrying agent permissions across organizational boundaries as a critical gap [9]. The paper noted that when an agent registered in one organization calls a service operated by another, neither OAuth 2.1 nor OIDC currently provides a standard mechanism to carry the agent's scoped permissions across that boundary — a gap that is particularly consequential in enterprise workflows where AI agents routinely interact with third-party services, SaaS platforms, and partner organization APIs. NIST announced its AI Agent Standards Initiative in February 2026 to address these gaps, but characterized the program as a multi-year effort [10].

The OIDC Client-Initiated Backchannel Authentication (CIBA) protocol offers a partial near-term mitigation for some delegation scenarios: it allows agents to pause and request human approval through asynchronous channels, resulting in short-lived, single-purpose tokens rather than persistent broad-scope credentials. But CIBA requires human availability and is incompatible with the real-time, low-latency operation of many production multi-agent workflows. For the window between now and the eventual maturation of purpose-built agentic identity standards — a window that could span several years — organizations must implement cross-agent privilege escalation controls using imperfect tools, compensating controls, and architectural constraints.

Section 3: Attack Patterns and Documented Vulnerabilities

Orchestration Framework Vulnerabilities

The research and vulnerability disclosure record through 2025 and early 2026 established that cross-agent privilege escalation paths are not purely theoretical. They have materialized in production-quality orchestration frameworks used by enterprises at scale.

The most significant disclosed vulnerability was CVE-2025-68664, publicly named "LangGrinch," which affected LangChain Core and received a CVSS score of 9.3 [11]. The vulnerability combined a prompt injection pathway with a serialization/deserialization flaw to create an end-to-end privilege escalation chain. An attacker could steer an AI agent, through crafted input, into generating structured outputs containing LangChain's internal serialization marker key. Because the marker was not properly escaped during serialization, attacker-controlled data could be later deserialized and interpreted as trusted LangChain objects rather than untrusted input. The practical consequence was that an attacker who could influence an agent's input — through a malicious document, a crafted web page, or a poisoned tool output — could cause that agent to serialize and ultimately expose all environment variables: cloud credentials, database connection strings, LLM API keys, and vector database secrets. The vulnerability encompassed twelve distinct attack vectors across common use cases including event streaming, logging, message history, memory, and caches [11]. For multi-agent systems, the implications were compound: an agent whose environment variables were exfiltrated through this chain became a stepping stone to any system those credentials could access.

Prior LangChain vulnerabilities suggest a recurring tension between framework expressiveness and security: CVE-2024-36480's remote code execution path and CVE-2023-44467's prompt injection vulnerability follow a pattern in which the mechanisms that enable flexible agent behavior also create exploitable attack surface [11]. Each of these vulnerabilities reflects the same underlying design tension in orchestration frameworks: the features that enable agents to process, execute, and act on diverse inputs create attack surface alongside capability.

Academic research published on arXiv in 2025 and 2026 documented additional attack classes specific to multi-agent environments. A comprehensive survey of agentic AI security found that systems composed of multiple coordinated agents can conduct reconnaissance across environments, progress through privilege escalation sequences, and coordinate data exfiltration while adapting to defensive measures — all without human intervention [4]. In multi-agent settings where agents share context windows or operate in overlapping memory spaces, a compromise or manipulation at one agent can propagate through the shared

context to affect the behavior of agents that were never directly targeted. The researchers characterized this as "cascading failure" — a term borrowed from infrastructure reliability engineering that captures the way a single exploitation event at one node can propagate through the network of trust relationships to undermine the security posture of the entire system.

Tool Poisoning and Function Call Injection

Tool poisoning is a related attack class that exploits the mechanism by which agents invoke external capabilities. In a typical agent architecture, tools — functions that the agent can call to interact with external systems, search the web, query databases, execute code, or communicate with other agents — are described to the agent through a tool manifest or system prompt. When an adversary can influence the content of that manifest, they can inject malicious instructions that appear to be legitimate tool descriptions. An agent processing a poisoned tool manifest may invoke a function that it believes performs one action while it actually performs another, potentially with elevated permissions.

The OWASP Top 10 for LLM Applications 2025 identified Excessive Agency (LLM06:2025) as one of the most prevalent risk categories in deployed AI systems, distinguishing three contributing root causes: agents having access to tools beyond their task scope (Excessive Functionality), tools operating with broader privileges than necessary (Excessive Permissions), and high-impact actions proceeding without human-in-the-loop controls (Excessive Autonomy) [12]. All three conditions are frequently present in multi-agent systems, and their combination creates conditions under which tool poisoning can have consequences far beyond what an injection into a single, bounded agent could achieve.

Function call injection — a variant in which the attacker targets the structured data formats used to specify tool invocations rather than the tool descriptions themselves — has emerged as a particularly practical attack vector. The arXiv paper analyzing privilege escalation in LLM-based agent systems documented a mandatory access control framework designed to address this class of attack, noting that current orchestration frameworks do not implement access control at the function call level: an agent's authority to call a function is determined by whether the function is listed in its tool manifest, not by whether the specific call it is making is within the scope of the agent's current task or authorization context [13]. An agent manipulated into making a function call outside its intended task scope has, from a logging perspective, made an apparently authorized function call. The authorization mismatch is semantic, not syntactic.

Cross-Agent Lateral Movement and Agent Spoofing

In systems where agents communicate with each other through message-passing interfaces, an attacker who compromises one agent gains the ability to interact with the full set of agents that the compromised agent could legitimately reach. Because inter-agent messages in most current frameworks carry no cryptographic

proof of origin, a compromised agent can send messages that purport to come from any other agent in the network — including the orchestrator. This is agent spoofing, and it enables lateral movement through the agent network that closely parallels the lateral movement phase of a conventional enterprise intrusion.

The research literature on open challenges in multi-agent security, published in mid-2025, characterized this problem as foundational: "Multi-agent systems rely on unauthenticated, unencrypted, or unvalidated messages. A spoofed instruction can mislead multiple agents, trigger privilege confusion, or collapse workflows" [14]. The researchers noted that addressing this challenge requires cryptographic signing of inter-agent messages — a capability that, to the authors' knowledge, no widely deployed multi-agent framework currently provides as a default, a gap the research literature has characterized as foundational [14]. Without it, the boundary between a trusted agent and an attacker impersonating a trusted agent is maintained only by prompt-level guardrails, which have been repeatedly demonstrated to be bypassable under adversarial conditions.

MITRE's ATLAS framework, updated to version 5.4.0 in February 2026, added specific techniques targeting the agentic tool ecosystem, including "Publish Poisoned AI Agent Tool" and "Escape to Host" [15]. These additions reflect the security research community's recognition that agentic AI systems create attack surfaces that do not map cleanly onto the existing ATT&CK framework for enterprise systems: the reasoning layer of an AI agent, the space in which instructions are interpreted and trust decisions are made, represents a new attack surface that requires new taxonomic categories.

Section 4: Architectural Factors That Amplify Risk

The Shared Context Window Problem

In most multi-agent orchestration frameworks, agents that are working on a shared task have access to a shared or inherited context window — a record of prior agent outputs, tool calls, retrieved data, and messages that enables each agent to understand the state of the overall task. This shared context is architecturally necessary: without it, agents cannot coordinate effectively, because each would be working without knowledge of what other agents have already done. But the shared context window is also an attack surface, because it means that any attacker who can inject content into any part of the shared context can potentially influence the behavior of all agents that will subsequently process that context.

This is the mechanism by which indirect prompt injection — in which an attacker embeds malicious instructions in data that an agent will retrieve and process, rather than sending instructions directly to the agent — can achieve cross-agent impact. If the retrieved data ends up in a shared context, a single injection point can influence an entire agent network. Research on indirect prompt injection in LLM-integrated applications has documented this attack pathway extensively, demonstrating how retrieved adversarial content can redirect agent behavior in ways that are indistinguishable from legitimate instructions [19]. The implications for multi-agent networks with shared context are a direct extension of those findings: the blast radius of a single injection grows with the number of agents sharing the affected context.

Agent Sprawl and Retirement Debt

The governance data on agent lifecycle management reveals an architectural risk factor that is distinct from technical attack vectors but operates as a force multiplier for them. The CSA January 2026 survey found that only 21 percent of organizations have formal decommissioning processes for AI agents, and only 19 percent expressed high confidence that their agents are fully retired when no longer needed [1]. The remaining organizations face what the survey characterized as "retirement debt" — a growing accumulation of agents that are technically still active, still hold credentials, and can still take actions, even though they are no longer being actively maintained, monitored, or governed.

Retired agents that retain credentials present a distinct and often overlooked attack surface. They represent lateral movement targets that will not receive the security attention of actively governed agents, may hold credentials that have not been rotated since the agent was last actively used, and may have permission profiles that were appropriate for a task that is no longer being performed. An attacker who identifies a deprecated orchestration agent in an organization's multi-agent infrastructure may find a credential-rich, low-surveillance entry point into the agent network — a privileged principal that nobody is watching. The

combination of excessive privilege (credentials and tool access that were appropriate during active use) and reduced oversight (monitoring and anomaly detection focused on currently active agents) makes retired agents a particularly attractive target for cross-agent privilege escalation campaigns.

The Aggregation Problem in Multi-Agent Networks

As organizations scale their multi-agent deployments, an emergent risk property becomes increasingly relevant: the aggregation problem. Individual agents may each hold limited, apparently reasonable levels of access. An HR automation agent may have access to employee records. A finance automation agent may have read access to budget systems. A communications agent may have permission to draft and send emails on behalf of users. None of these permission profiles, considered individually, represents an obviously excessive grant. But a multi-agent network in which these agents can interact, share context, and coordinate creates a system whose aggregate access is substantially higher than what any individual authorization decision contemplated.

Formal analysis published in 2026 identified this aggregation problem as a distinct challenge for least-privilege enforcement in multi-agent systems [13]. Traditional IAM frameworks evaluate whether a given principal's permissions are appropriate to that principal's function. In a multi-agent system, the relevant question is whether the emergent capabilities of the agent network — what the network can collectively accomplish through agent cooperation — are appropriate and intended. This is a harder question to answer, because it requires reasoning about agent interaction patterns that may not have been explicitly designed and that may not be visible to the administrator reviewing any individual agent's permissions. The attack surface, in this sense, is not any individual agent but the graph of trust and cooperation relationships among agents.

Section 5: Toward Secure Multi-Agent Architectures

The Zero Trust Foundation

The architectural principles required to address cross-agent privilege escalation are not new inventions. They are applications of Zero Trust principles — never trust, always verify; least privilege; assume breach — to the agent layer of enterprise infrastructure. What is new is the need to apply these principles to a class of principal that lacks the properties that existing Zero Trust implementations assume: a verified identity, a defined role, and a stable relationship between credentials and the principal that holds them. Building a Zero Trust architecture for agents requires solving the identity problem first, then building authorization, monitoring, and response capabilities on top of a verified identity foundation.

The initial practical step for most organizations is agent discovery and inventory. Organizations cannot apply least privilege to agents they do not know exist, and they cannot revoke credentials for agents they have not inventoried. The CSA January 2026 survey found that 82 percent of organizations discovered previously unknown agents in their environments in the prior year — a figure that indicates how far most organizations are from having a complete agent inventory [1]. Achieving a reliable agent inventory requires active discovery tooling, not just documentation of intentional deployments. Shadow agents — those deployed without knowledge of security teams, often by developers or business units — must be included in the inventory before any coherent privilege governance program can function.

With a complete inventory, organizations can begin applying the principle of minimum viable privilege: each agent should hold exactly the credentials, tool access, and permissions required to accomplish its specific, defined function, and no more. This principle is straightforward to state but challenging to implement in practice, because agent function often evolves, and in practice developers building agent systems often request broader permissions than strictly necessary to avoid authorization errors during development — a pattern familiar from service account provisioning in traditional IAM programs. Operationalizing minimum viable privilege requires treating agent permission profiles with the same rigor applied to service account privileges in traditional IAM programs — with regular review cycles, documented justifications for each permission grant, and automated alerts when agents attempt to use capabilities outside their defined scope.

Implementing Agentic Identity

While formal standards for agentic identity are still maturing, organizations can take immediate steps to improve the quality of agent authentication using available mechanisms. The SPIFFE/SVID workload identity framework, originally designed for service-to-service authentication in microservices architectures, is architecturally appropriate for agent identity in multi-agent systems: it provides cryptographically verifiable

identities for compute workloads, short-lived certificates that expire without rotation burden, and an attestation model that can bind an identity to the specific agent process rather than to a shared credential. Among organizations planning to use modern authentication mechanisms for agents, 19 percent of CSA survey respondents indicated plans to implement SPIFFE/SVID [2], suggesting awareness but far from universal adoption.

For organizations not yet ready to implement workload identity frameworks, intermediate improvements include eliminating shared service accounts for agent authentication (each agent should have a distinct credential), enforcing credential rotation on a defined schedule, and implementing token-binding mechanisms that tie API credentials to specific agent contexts where the infrastructure supports it. These measures do not solve the fundamental problem — without a verified identity bound to a specific agent instance, credentials remain portable and spoofable — but they reduce the persistence of any credential compromise and provide clearer audit trails when agent behavior requires investigation.

The emerging proposals for OIDC-A and purpose-built agentic authorization represent the longer-term path toward verified agentic identity [8]. Organizations should engage with these standards development processes, provide implementation feedback from their own multi-agent deployments, and plan their identity architecture with the intent to adopt purpose-built agentic identity standards when they are finalized. Building agent identity infrastructure on top of extensible, standards-based identity foundations — rather than ad hoc credential management — minimizes the cost of migration when formal standards arrive.

Securing Inter-Agent Communication

The absence of cryptographic signing on inter-agent messages is the technical mechanism through which agent spoofing and much of cross-agent privilege escalation becomes possible. While no widely deployed framework currently provides this capability as a default, organizations building custom multi-agent orchestration can implement message signing using standard cryptographic libraries. Each agent in the network should hold a signing key, and messages from that agent to other agents should be signed with that key. Receiving agents should verify signatures before acting on instructions, and should refuse to process unsigned messages or messages signed with unrecognized keys.

This approach requires a key distribution infrastructure — a registry that maps agent identities to their public keys and that receiving agents can query — but the operational infrastructure required (key distribution, per-agent key lifecycle management, and signature verification at message receipt) is architecturally similar to the PKI and certificate management infrastructure that enterprises with mature IAM programs already operate for service-to-service authentication. Cloud-native equivalents such as AWS KMS, Azure Key Vault, and GCP Cloud KMS provide accessible implementations for organizations without on-premises CA infrastructure. Signed inter-agent messages substantially reduce agent spoofing as an attack vector, shift the attacker's required capability from message injection to key compromise, and create an audit trail that attributes every instruction in the agent network to a specific, verified source.

For organizations using managed orchestration platforms rather than custom-built systems, the near-term mitigation is isolation: reducing the blast radius of any single agent compromise by partitioning agents into network segments with limited inter-segment communication paths. An agent that cannot reach agents outside its designated segment cannot be used as a pivot point for lateral movement through the broader agent network, even if it is compromised or manipulated. Privilege boundaries should be designed so that no agent has direct network access to agents with substantially higher privilege levels than its own function requires.

Human-in-the-Loop as Security Architecture

Human oversight is often discussed in the context of AI safety as a correctness mechanism — a way to ensure that AI agents make good decisions. It is equally important as a security mechanism. The OWASP Excessive Agency finding identified the absence of human-in-the-loop controls as one of the three root causes of agent privilege misuse [12], and the CSA January 2026 survey found that the majority of organizations (53 percent) operate agents autonomously for low-risk tasks while requiring human review for high-risk actions — a tiered autonomy model that, when properly implemented, constrains the blast radius of any privilege escalation event [1].

Effective human-in-the-loop security controls for multi-agent systems require more than a general policy of "humans review high-risk actions." They require a precise definition of which agent actions constitute high risk, a technical mechanism by which agents can recognize when they are approaching a high-risk action and pause for approval, and an asynchronous approval channel that allows humans to respond without requiring real-time availability. The OIDC CIBA protocol supports this model at the authentication layer, and several orchestration frameworks have begun implementing configurable approval gates for specific action types. The effectiveness of these controls depends on the accuracy of the action risk classification: an approval gate that triggers on the wrong threshold — too high, too often, or not at all for genuinely dangerous actions — either creates operational friction without security benefit or provides false assurance while dangerous actions proceed unchecked.

The practical recommendation is to define, for each multi-agent workflow, a specific set of irreversible or high-impact actions that require human approval before execution. Typical examples include sending communications on behalf of humans, writing or deleting records in authoritative data stores, initiating financial transactions, escalating privileges or modifying access control configurations, and invoking external APIs with write permissions. These actions should be explicitly enumerated in the agent's tool configuration, and the approval gate should be enforced at the tool level — not at the prompt level, which is bypassed by adversarial instructions — so that no combination of prompt injection or context manipulation can cause the agent to bypass the approval requirement.

Section 6: A Governance Framework for Multi-Agent Security

Risk Assessment Across the Agent Network

The first step in governing cross-agent privilege escalation risk is a structured assessment of the agent network as a whole rather than individual agents in isolation. This assessment should map the trust relationships between agents — which agent can instruct which, what authority is delegated from orchestrators to subagents, and what the transitive privilege graph looks like — and should identify the paths through that graph that would, if traversed by an attacker, result in the highest-impact outcomes. The paths connecting low-privilege agents with access to external data sources to high-privilege agents with access to sensitive enterprise systems are the ones that require the most rigorous controls.

The table below provides a framework for prioritizing controls based on the intersection of an agent's privilege level (what it can do) and its exposure surface (how likely it is to encounter adversarial input).

Agent Privilege Level	External Data Exposure	Recommended Control Tier
High (direct access to sensitive systems or credentials)	Any	Tier 1: Signed messages, approval gates for all writes, continuous monitoring
High	Isolated (no external input)	Tier 2: Signed messages, approval gates for irreversible actions, event-driven alerts
Medium (limited system access)	High (web browsing, external APIs, user-provided content)	Tier 2: Output validation, sandboxed execution, approval gates for high-risk actions
Medium	Low	Tier 3: Standard credential management, periodic review
Low (read-only, sandboxed)	High	Tier 2: Output validation, escalation detection
Low	Low	Tier 3: Inventory and periodic review

This tiering model is not intended to eliminate security controls for lower-risk agents; it is intended to ensure that the highest-risk combinations — high privilege paired with external exposure — receive the most intensive protection. An agent with administrative credentials that processes documents from external sources is a critical risk, regardless of its intended function, because the combination of administrative access and externally influenced behavior is exactly the condition that cross-agent privilege escalation attacks exploit.

Incident Response for Agent Networks

The existing data on AI agent security incidents indicates that organizations regularly encounter agent behavior that requires investigation and response, yet most have not adapted their incident response procedures for agent-specific scenarios. The CSA January 2026 survey found that 65 percent of organizations had experienced an AI agent security incident in the prior twelve months, and that 61 percent of those involved data exposure or mishandling [1]. Virtually all organizations reporting incidents described material business impact [1] — a finding that underscores the operational severity of agent security failures even in environments where defenders had not yet developed incident response procedures tailored to agent architectures.

Agent security incident response differs from conventional IR in several ways. First, the evidence trail is different: agent behavior is captured in context windows, tool call logs, and message queues rather than in traditional network logs and endpoint telemetry. Effective agent IR requires capturing and preserving these artifacts, which many current monitoring systems do not prioritize. Second, the scope of investigation is different: because agents can act autonomously across many systems in a short time, the timeline of a compromised agent's activity may span thousands of actions over minutes. Reconstructing that timeline requires tooling designed for the speed and scale of agent operations. Third, the containment strategy is different: stopping a compromised agent requires revoking its credentials and removing it from the orchestration network, not merely isolating a network endpoint. If the compromised agent had already passed credentials or context to downstream agents, those must be identified and assessed as well.

Organizations should develop agent-specific playbooks that address: indicators of agent compromise (anomalous tool calls, out-of-scope actions, unexpected data access patterns); procedures for pausing or terminating an agent mid-operation without corrupting the broader workflow; credential revocation workflows for agents with dynamic, delegated credentials; and post-incident analysis procedures for tracing the chain of causation from attacker input to compromised action through the full agent network.

CSA Resource Alignment

CSA's framework portfolio provides substantive guidance across the dimensions of cross-agent privilege escalation risk, and organizations building multi-agent security programs should draw on these resources as foundational references rather than developing bespoke frameworks from scratch.

The MAESTRO agentic AI threat model, developed under the CSA AI Safety Initiative, provides the most directly applicable threat taxonomy for multi-agent security. MAESTRO's seven-layer model specifically addresses the agent layer, the orchestration layer, and the trust relationships between them, and it includes threat categories that map directly to the attack patterns described in this paper: goal hijacking, context poisoning, tool misuse, and inter-agent exploitation. Organizations using MAESTRO as their threat modeling framework have a structured vocabulary for the risk categories this paper addresses and a path for mapping those risks to the control domains of the AI Controls Matrix.

The AI Controls Matrix (AICM) provides the control framework most directly applicable to multi-agent security governance. Several AICM domains are particularly relevant: the Identity and Access Management domain addresses credential management, authentication standards, and permission governance for AI systems; the Monitoring and Logging domain addresses the audit trail requirements for agent behavior; the Supply Chain and Third-Party domain addresses the risks created by multi-agent systems that interact with external agents or third-party agent platforms; and the Incident Response domain addresses the procedures required to detect, contain, and recover from agent security incidents. AICM's Orchestrated Service Provider (OSP) category, with its dedicated implementation and auditing guidelines, is specifically designed for organizations operating multi-agent orchestration infrastructure.

CSA's Zero Trust guidance, particularly the work on Zero Trust for automation and orchestration environments, provides architectural principles applicable to the agent communication layer. The core Zero Trust principle — that no message, instruction, or identity claim should be trusted without verification, regardless of network origin — is precisely the principle that most current multi-agent systems violate when they process inter-agent instructions without cryptographic authentication. Applying Zero Trust to the agent message layer, as this paper recommends, is a direct application of the principles CSA's Zero Trust program has developed for microservices, APIs, and service-to-service communication.

The CSA Agentic AI Red Teaming Guide [17] provides testing methodologies specifically designed for multi-agent environments, covering twelve threat categories including agent authorization hijacking, permission escalation, multi-agent exploitation, and supply chain attacks. Organizations that have not yet red-teamed their multi-agent deployments against the specific attack patterns described in this paper should use the guide as a testing curriculum; those that have conducted conventional penetration testing of their agent infrastructure should use it to ensure that agentic-specific attack vectors — which do not appear in standard penetration testing frameworks — have been adequately assessed.

The STAR for AI program provides an assessment and assurance pathway for organizations seeking to evaluate or communicate their multi-agent security posture. As multi-agent deployments become ubiquitous, customer security teams will increasingly ask about the security controls governing agent behavior, and the STAR for AI framework provides a standardized vocabulary and assessment structure for answering those questions systematically.

These CSA frameworks operate alongside other relevant governance resources. NIST's AI Risk Management Framework (AI RMF 1.0) [18] provides foundational risk management principles broadly applicable to AI systems. MITRE ATLAS [15], cited throughout this paper for its adversarial technique taxonomy, offers a complementary view from the attacker's perspective. Emerging regulatory frameworks, including the EU AI Act's requirements for high-risk AI systems, establish compliance obligations that multi-agent deployments must address. CSA's MAESTRO, AICM, and related guidance represent the most directly applicable resources for organizations focused specifically on multi-agent security governance, but they are most effective when used in conjunction with these broader frameworks.

Conclusions and Recommendations

Multi-agent AI systems are neither more nor less secure than the trust relationships they embody. When agents inherit, delegate, and transfer authority through channels that carry no cryptographic guarantees of identity or integrity, the security of the overall system is bounded by the security of its weakest trust relationship — which, in complex agent networks, may be a low-privilege agent at the network's edge that is exposed to adversarial external data. Organizations that deploy multi-agent AI without addressing this foundational architectural property are not simply accepting theoretical risk; they are operating systems with documented, proven attack paths against a threat actor community that has begun cataloging and exploiting them.

The recommendations that follow are organized in three tiers — immediate actions that can be implemented within the next ninety days using existing tools and processes; medium-term architectural improvements requiring planning and investment over a six-to-twelve month horizon; and strategic posture changes that position organizations for the emerging standards landscape. None requires waiting for formal agentic identity standards to mature; all are actionable now.

Immediate Actions (0–90 days)

Organizations should conduct a complete inventory of deployed AI agents, including agents deployed outside IT governance processes. Shadow agent discovery must precede any coherent privilege governance program. Each discovered agent should be classified by its privilege profile (what systems and data it can

access) and its exposure profile (whether it processes external or user-provided content). Agents combining high privilege with external exposure should be prioritized for immediate review.

Organizations should eliminate shared service accounts for agent authentication where technically feasible, replacing them with individual credentials per agent instance. Existing static API keys should be documented, rotated, and placed under credential management tooling that enforces rotation schedules. Any agent with credentials to sensitive systems that is not actively maintained should be deprovisioned immediately, as inactive agents with live credentials represent the risk category this paper describes as retirement debt.

Organizations should establish, in writing, the specific action types for each multi-agent workflow that require human approval before execution. These definitions should be enforced at the tool level — through approval gates in the orchestration framework — not merely through prompt instructions. This human-in-the-loop control is the highest-leverage near-term security control for limiting the damage of any successful privilege escalation attempt.

Medium-Term Architecture (6–12 months)

Organizations should develop a roadmap for implementing workload identity (SPIFFE/SVID or equivalent) for agent authentication, and should begin piloting it in their highest-risk multi-agent deployments. The operational infrastructure for workload identity — key distribution, certificate lifecycle management, attestation — should be treated as foundational IAM infrastructure for the agentic layer, not as a specialized project.

Organizations should implement message integrity mechanisms for inter-agent communication in new multi-agent deployments, and should retrofit them in existing deployments where the operational risk justifies the engineering investment. The minimum viable implementation is cryptographic signing of inter-agent messages using per-agent keys, with signature verification enforced at message receipt. This substantially reduces agent spoofing as an attack vector and creates an attribution foundation for incident investigation.

Organizations should conduct adversarial testing of their multi-agent deployments using the CSA Agentic AI Red Teaming Guide [17] as a testing framework, with specific focus on the cross-agent privilege escalation patterns described in this paper: confused deputy attacks, delegation chain exploitation, shared context injection, and tool poisoning. Results should inform control prioritization and architectural remediation.

Strategic Posture

Organizations should monitor the development of purpose-built agentic identity standards — including OIDC-A proposals from the OpenID Foundation and the NIST AI Agent Standards Initiative — and plan their agent identity architecture to accommodate adoption of those standards as they mature. Building agent identity infrastructure on extensible, standards-based foundations now is substantially less costly than migrating from ad hoc credential management later.

Organizations should engage with CSA's MAESTRO and AICM working groups to contribute operational experience from their own multi-agent deployments to the development of the frameworks that will govern this technology at industry scale. The standards bodies developing agentic AI security frameworks have recognized that they need operational data from deployed multi-agent systems to produce guidance that is both comprehensive and practically applicable. Organizations that contribute to this process will help shape frameworks that reflect the real attack surfaces they are defending.

Finally, organizations should treat multi-agent security as a discipline requiring dedicated ownership within the security organization. The convergence of identity, orchestration, and AI-specific attack vectors that characterizes cross-agent privilege escalation does not map cleanly onto any existing security team's charter. Creating explicit ownership — whether in an IAM team, a security architecture team, or an emerging AI Security function — is a prerequisite for the coherent, sustained attention that this attack class demands.

References

[1] Hillary Baron, Marina Bregkou, Josh Buker, Ryan Gifford. "Autonomous but Not Controlled: AI Agent Incidents Now Common in Enterprises." Cloud Security Alliance, 2026. (Survey conducted January 2026, 418 respondents.)

[2] Hillary Baron, Marina Bregkou, Josh Buker, Ryan Gifford, John Yeoh. "Securing Autonomous AI Agents." Cloud Security Alliance, 2026. (Survey conducted September–October 2025, 285 respondents.)

[3] Guangpeng Li et al. "Authenticated Delegation and Authorized AI Agents." arXiv:2501.09674, January 2025.

[4] Adversa AI Research. "Agentic AI Security: Threats, Defenses, Evaluation, and Open Challenges." arXiv:2510.23883, October 2025.

[5] CSA AI Safety Initiative. "Confused Deputy Attacks on Autonomous AI Agents." CSA Lab Space, 2025.

[6] Research Team. "Securing Agentic AI: A Comprehensive Threat Model and Mitigation Framework for Generative AI Agents." arXiv:2504.19956, April 2025.

[7] OWASP Gen AI Security Project. "OWASP Top 10 for Agentic Applications 2026." OWASP, 2026.

[8] OpenID Foundation. "Identity Management for Agentic AI." OpenID Foundation, October 2025.

[9] NIST National Cybersecurity Center of Excellence. "Accelerating the Adoption of Software and AI Agent Identity and Authorization." NIST NCCoE Concept Paper, February 2026.

[10] NIST. "Announcing the AI Agent Standards Initiative for Interoperable and Secure Innovation." NIST, February 2026.

[11] Cyata Security Research. "All I Want for Christmas is Your Secrets: LangGrinch Hits LangChain Core (CVE-2025-68664)." Cyata, December 2025. See also: NIST National Vulnerability Database. "CVE-2025-68664 Detail." NIST NVD.

[12] OWASP Gen AI Security Project. "OWASP Top 10 for LLM Applications 2025." OWASP, November 2024.

[13] Research Team. "Taming Various Privilege Escalation in LLM-Based Agent Systems: A Mandatory Access Control Framework." arXiv:2601.11893, January 2026.

[14] Research Team. "Open Challenges in Multi-Agent Security: Towards Secure Systems of Interacting AI Agents." arXiv:2505.02077, May 2025.

[15] MITRE. "MITRE ATLAS v5.4.0." MITRE Corporation, February 2026.

[16] Simon Willison. "Cross-Agent Privilege Escalation: When Agents Free Each Other." Simon Willison's Weblog, September 2025.

[17] Ken Huang et al. "Agentic AI Red Teaming Guide." Cloud Security Alliance, 2025.

[18] NIST. "AI Risk Management Framework 1.0." NIST, January 2023.

[19] Kai Greshake et al. "Not what you've signed up for: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injections." arXiv:2302.12173, 2023.