informatics

MDPI

Article

From Service Accounts to Agentic Identities: A Zero Trust Governance Framework for Delegated Authority, Runtime Tool Control, and Accountable Non-Human Actors in Enterprise Cybersecurity

Mohammad Nizamuddin [1,*] and Ryana Sikder [2]

[1] Engineering, Physics, and Technology Department, Bronx Community College, City University of New York, New York, NY 1045, USA
[2] Medidata Solutions, Inc. Medidata (United States), New York, NY 10014, USA; ryanasikder1@gmail.com
* Correspondence: mohammad.nizamuddin@bcc.cuny.edu

Abstract

Agentic AI is changing enterprise cybersecurity as AI systems move beyond passive content generation toward autonomous planning, tool use, delegated execution, and operational action. As agents connect to email, code repositories, security operations center (SOC) platforms, finance workflows, cloud services, and enterprise application programming interfaces (APIs), they increasingly function as dynamic non-human identities rather than conventional software tools or service accounts. Existing identity and access management (IAM), Zero Trust, machine identity, and AI-governance approaches remain fragmented in their treatment of delegated authority, task intent, autonomy, runtime tool use, and auditable organizational consequences. This paper addresses these gaps by proposing the AIGATE (Agentic Identity Governance, Authority, Tool-Control and Evidence) Framework. AIGATE integrates eight governance layers: agent identity registration, lifecycle governance, delegated authority mapping, intent-bound access, least agency and least privilege, runtime tool-call control, audit evidence and accountability, and revocation and resilience. The framework treats agents as governed non-human enterprise identities whose actions remain attributable to designated human and organizational roles. AIGATE is developed through a structured critical synthesis of the recent literature on agentic AI security, machine identity, Zero Trust, runtime enforcement, and AI governance, with literature-derived governance requirements mapped explicitly to the eight framework layers. Three SOC, DevOps, and finance scenarios are used as illustrative applications rather than empirical validation. The contribution is an integrated governance architecture connecting identity, delegated authority, autonomy, runtime enforcement, evidence, and revocation across the agent lifecycle.

Keywords: agentic AI; non-human identity; machine identity governance; Zero Trust; runtime tool-call control

Check for updates

Academic Editor: Jiang Bian

Received: 9 July 2026
Revised: 25 August 2026
Accepted: 4 September 2026
Published: 8 September 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Generative AI has rapidly shifted from systems that mainly produce text, code or recommendations toward agentic systems that can plan, reason, use tools and act within digital environments [1]. This shift changes the cybersecurity problem because agentic AI is not limited to producing an output but can interact with APIs, access data, invoke tools

and execute multi-step tasks [2]. Large language model (LLM)-based agents are commonly built around planning, memory, tool use and feedback loops, which allows them to operate as task-oriented actors rather than passive conversational systems [2]. Agentic AI therefore creates a governance challenge because systems that act across organizational resources raise questions of authority, ownership, accountability and control [3].

In enterprise environments, AI agents are increasingly connected to email, calendars, databases, cloud platforms, software repositories, workflow systems, SOC tools and enterprise APIs [4]. This connectivity means that agents are no longer only assistants but can become non-human actors that read information, interpret instructions, call functions and trigger organizational actions [5]. The risk is amplified because agentic systems can reason over ambiguous instructions, process untrusted content, retain memory and adapt their actions during execution [6]. The enterprise security boundary is therefore moving from a simple user-application-system model toward a chain involving human users, AI agents, sub-agents, tools, APIs and organizational resources [7].

Traditional identity and access management was designed mainly for human users, applications, devices, service accounts and relatively stable machine identities [8]. This model becomes insufficient when AI agents operate dynamically, delegate tasks, change context, use multiple tools and act under varying levels of autonomy [8]. A service account normally performs a predefined technical function, while an AI agent may interpret a goal, choose actions, access different resources and create new operational consequences [5]. The governance question is therefore not only whether an identity has permission, but whether the agent's action is authorized, intent-aligned, necessary, auditable and revocable [9].

This problem is especially serious because tool-using agents often process trusted instructions, retrieved data and tool outputs through the same generative channel [9]. Indirect prompt injection shows that malicious instructions embedded in emails, documents, web pages or tool outputs can influence agents to perform actions that users did not intend [10]. Agent Security Bench further shows that vulnerabilities can occur across system prompts, user prompts, planning, memory retrieval and tool usage [11]. Prompt Flow Integrity frames this as a privilege-escalation issue because untrusted data may influence privileged tool use when instruction flow and data flow are not properly separated [12]. Runtime privilege-control research shows that agent security cannot rely only on model alignment because real-world effects occur through tool calls that require deterministic enforcement [13]. Tool-boundary approaches such as minimization, sanitization and runtime rule enforcement are therefore important because they control what enters tools, what returns to agents and what actions are allowed before execution [14].

Although recent research has examined agentic AI security, prompt injection, runtime enforcement, non-human identities, machine identity governance and AI oversight, these areas remain fragmented [15]. Security studies often focus on technical attacks and defenses, while governance studies focus on oversight and accountability without fully operationalizing agent identity, delegated authority and runtime access decisions [16]. Existing machine identity work recognizes that AI agents, service accounts, API tokens and automated workflows create a governance blind spot, but enterprise cybersecurity still lacks an integrated framework for governing agentic AI as dynamic non-human identity [5]. This gap matters because enterprises may deploy agents without clearly defining ownership, authority, autonomy level, tool boundaries, audit evidence, revocation rules and accountability mechanisms [3].

This paper addresses these gaps by developing the Agentic Identity Governance, Authority, Tool-Control and Evidence (AIGATE) Framework, a Zero Trust-aligned governance architecture for agentic AI as a non-human enterprise identity. It makes four contributions. First, it defines agentic identity as a governance extension of conventional machine identity

for systems that can interpret tasks, use tools, and act under delegated authority, while retaining organizational accountability with designated human roles. Second, it derives a set of agent-specific governance requirements from a critical synthesis of the fragmented literature on agent security, machine identity, IAM, Zero Trust, runtime enforcement, and AI governance. Third, it integrates these requirements into eight connected AIGATE layers and operationalizes least agency alongside least privilege so that both access and autonomy are bounded. Fourth, it examines the conceptual applicability of the framework through three enterprise scenarios involving SOC automation, DevOps, and finance workflows. These scenarios illustrate how the framework would constrain unsafe action paths but are not presented as empirical validation.

2. Conceptual Background: From Machine Identities to Agentic Non-Human Actors

Machine identities are digital identities used by non-human entities such as service accounts, API keys, certificates, workloads, bots, scripts, automated workflows and software services to authenticate and interact with enterprise systems [5]. Traditional machine identities normally support predefined technical functions, such as application-to-application communication, workload authentication, certificate-based trust or automated system tasks [8]. Non-human identities are already difficult to govern because they are often long-lived, overprivileged, poorly inventoried and weakly linked to a clear human owner [5]. Agentic AI expands this category because the identity is no longer attached only to a credential or automation process but to a system capable of reasoning, selecting tools and producing actions [8]. This makes AI agents more governance-sensitive than ordinary service accounts because they can interpret intent, change execution paths and act across systems in ways that may not be fully predefined at deployment time [7].

2.1. What Makes Agentic AI Different?

Agentic AI differs from traditional software automation because it combines autonomy, planning, memory, tool use, delegation and environmental interaction within a single operational system [17]. Autonomy matters because the agent may act without continuous human approval, which increases the need for bounded authority and escalation controls [4]. Planning matters because the agent may decompose a broad goal into multiple steps, creating security consequences that may not be visible from the original user instruction [2]. Tool use matters because agents can call APIs, browsers, databases, code interpreters, workflow tools and enterprise platforms, turning model output into operational action [10]. Memory matters because agents may retain context across interactions, which can improve task continuity but also introduce risks of memory poisoning, privilege accumulation and unintended data reuse [11]. Delegation matters because agents may interact with other agents or services, creating authority chains where responsibility becomes difficult to trace [7]. Environmental interaction matters because the agent does not merely produce text but acts on external systems, making confidentiality, integrity, availability and accountability directly relevant [6].

2.2. Why Is "Agent as Identity" the Right Framing?

The "agent as identity" framing is necessary because agentic AI systems can hold access rights, invoke tools, process enterprise data and produce organizational effects [5]. Treating agents only as applications or AI models underestimates their security role because their behavior is shaped by goals, prompts, retrieved content, tools, memory and runtime context [15]. A stronger approach is to treat AI agents as identity-bearing security principals with defined ownership, purpose, authority, autonomy level, tool permissions, monitoring

requirements and revocation conditions [8]. This framing allows enterprise governance to move from static credential control toward accountable control of agent behavior, delegated authority and runtime action [3].

The conceptual distinction between human, machine and agentic identity is summarized in Figure 1, which shows how enterprise identity governance evolves from user authentication to machine authentication and finally to accountable agentic action.

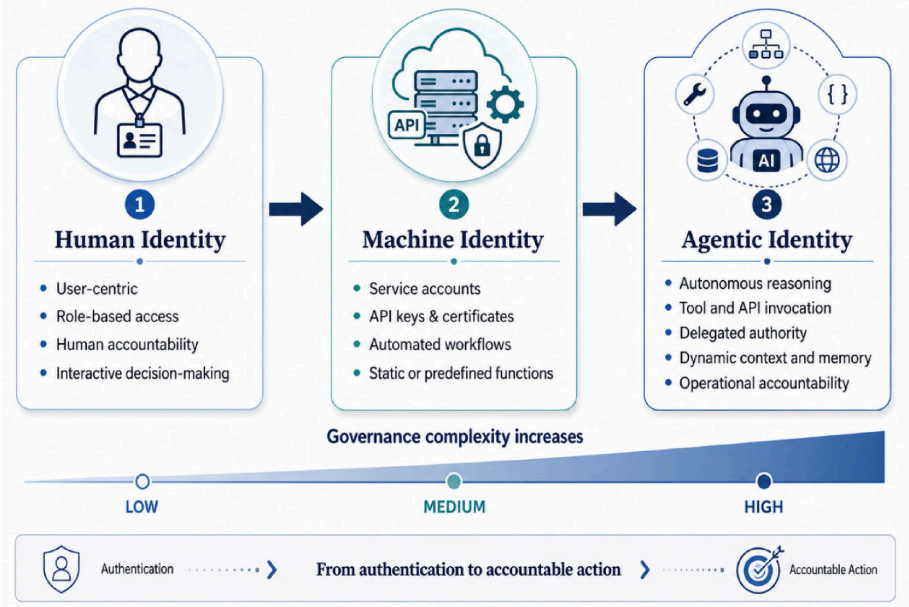


Figure 1. Evolution from human identity to machine identity to agentic identity.

Figure 1 illustrates the progression from human identity to machine identity and agentic identity. In this paper, a machine or workload identity is the credentialed non-human principal used by a service, application, workload, or automation process for authentication and access. An agentic identity extends this concept to an AI system whose operational behavior can vary with goals, context, memory, tools, and delegated authority. An agent instance is a specific deployed or running instantiation of that agent and should therefore carry an instance or session identifier linked to its registered agentic identity. Delegated human authority is different from all three: it is the bounded authorization transferred by a named requester, owner, or organizational role for a defined task and does not become an inherent property of the agent. Accordingly, AIGATE uses the term accountable non-human actor in an attributional rather than legal or managerial sense: the agent and its actions must be identifiable, traceable, and auditable, while ultimate organizational accountability remains with designated human and organizational roles [5].

3. Methodological Approach

This study adopts a structured critical literature synthesis and conceptual framework-development approach rather than a systematic literature review. The objective is conceptual integration and framework construction, not exhaustive retrieval, meta-analysis, or estimation of an empirical effect. Critical literature synthesis is appropriate where dispersed and rapidly developing research streams must be compared to identify conceptual gaps and develop an integrative contribution [18]. Consistent with a concept-centric review approach, the analysis is organized around recurring security and governance problems rather than presenting prior studies sequentially [19]. The conceptual framework-development component follows the logic of clarifying con-structs, relationships, and mechanisms rather than testing empirical hypotheses [20]. This design is particularly suitable for agentic AI

security because relevant knowledge is currently distributed across agent security, machine identity, identity and access management (IAM), Zero Trust, runtime enforcement, and AI-governance research.

The substantive review was bounded primarily to the literature published from 2024 to July 2026, reflecting the rapid development of agentic AI and its associated security mechanisms. Earlier sources were retained where necessary to support the methodological approach or foundational concepts. The evidence base draws on peer-reviewed journal and conference publications together with recent preprints where a peer-reviewed version was not available. Targeted searches were conducted through IEEE Xplore, ACM Digital Library, Scopus, Web of Science, ScienceDirect, SpringerLink, and ACL Anthology, supplemented by backward and forward citation chaining. Sources were identified through targeted scholarly searching and citation chaining using combinations of terms including agentic AI, LLM agent, AI agent security, prompt injection, tool use, machine identity, identity and access management, Zero Trust, delegated authority, runtime enforcement, auditability, governance, and revocation. The search was purposive rather than exhaustive because the aim was to identify literature capable of informing framework constructs rather than to estimate the prevalence of findings across a complete research population. Representative search strings were adapted to the syntax and search fields of each database. Three principal search families were used: (1) ("agentic AI" OR "LLM agent*" OR "AI agent*") AND (security OR cybersecurity OR "prompt injection" OR "tool use"); (2) ("machine identity" OR "non-human identity" OR IAM OR "Zero Trust" OR "delegated authority") AND ("agentic AI" OR "AI agent*"); and (3) ("LLM agent*" OR "AI agent*") AND ("runtime enforcement" OR "tool-call control" OR "least privilege" OR auditability OR revocation). In Scopus these concepts were applied primarily to title, abstract, and keyword fields, in Web of Science to Topic fields, and equivalent title, abstract, metadata, or full-record searches were used in the other databases. The search process was iterative, with the final literature update completed in July 2026.

Sources were retained when they made a substantive contribution to at least one of five analytical themes: (1) agentic AI foundations and architecture, (2) agent-security threats, (3) tool-use and prompt-injection risk, (4) identity, IAM and Zero Trust, and (5) governance, oversight and auditability. Studies were considered particularly relevant where they identified agent-specific threats, identity or authorization limitations, runtime control mechanisms, delegation or autonomy concerns, audit requirements, or lifecycle-governance issues. General generative AI studies without direct relevance to autonomous or tool-using agents, sources that did not contribute to the security or governance problem, and duplicative sources that did not add a distinct conceptual or technical contribution were excluded. Because this is a conceptual synthesis rather than a systematic review, no claim of exhaustive coverage or PRISMA-style screening is made.

The final conceptual synthesis retained 20 substantive research sources [1–17,21–23]. Their primary analytical grouping was: agentic AI foundations and architecture [1,2,21]; agent-security threats [4,6,15,17]; tool-use and prompt-injection risk [9–14,22,23]; identity, IAM and Zero Trust [5,7,8]; and governance, oversight and auditability [3,16]. Some sources informed more than one theme, but they are listed here according to their principal role in the synthesis. References [18–20] support the literature-review and conceptual-framework methodology, while [24–27] provide standards and practitioner positioning rather than forming the core substantive evidence set. Because the search was purposive and no PRISMA-style screening log was maintained during the original framework development, an exact retrospective count of all initially encountered records is not reported.

Both authors participated in the targeted searches and concept-centric coding of the retained literature. Each author reviewed the analytical-theme assignments and literature-

to-layer mappings, and differences in interpretation were discussed and resolved through consensus before the final governance requirements and framework mappings were agreed. Preprints were retained only where they addressed rapidly emerging agentic-AI topics for which an equivalent peer-reviewed version was not available. Their credibility was assessed based on methodological transparency, technical specificity, author and institutional provenance, consistency with related peer-reviewed evidence, and the availability of later published versions; where an appropriate peer-reviewed version became available, that version was preferred.

Framework development followed three analytical stages. First, relevant evidence from the retained literature was coded concept-centrically within the five themes, focusing on the security problem identified, the control or governance mechanism proposed, and the limitation that remained unresolved at the enterprise level. Second, recurring limitations were synthesized into eight governance requirements concerning identity, lifecycle, delegation, intent, autonomy, runtime control, accountable evidence, and containment/recovery. Third, each requirement was translated into a corresponding AIGATE layer. The literature-to-requirement-to-layer relationship is made explicit in the derivation matrix in Section 5, allowing the framework structure to be traced back to the underlying evidence rather than being presented as an arbitrary eight-layer design. This follows conceptual framework-development logic in which the contribution lies in clarifying constructs, relationships, and mechanisms rather than testing empirical hypotheses [20].

Finally, the resulting framework was applied to three enterprise scenarios involving SOC automation, DevOps, and finance workflows. These scenarios were selected because they represent distinct high-impact contexts in which agents may combine sensitive resources, delegated authority, untrusted inputs, powerful tools, and varying levels of autonomy. The scenarios are used to examine the framework's conceptual applicability and control coverage, not to demonstrate empirical effectiveness. Accordingly, conclusions concerning effectiveness, quantitative risk reduction, or generalizability are reserved for future empirical evaluation.

4. Literature Review

4.1. Architectural Foundations of Agentic AI

Agentic AI is grounded in the movement from passive language generation toward systems that can plan, remember, coordinate, use tools and act across environments [1]. Earlier generative AI systems were mainly evaluated by output quality, but agentic systems must also be evaluated by the consequences of their actions, because their design includes operational loops rather than one-off responses [2]. LLM-based autonomous agents commonly combine a language model with planning modules, memory structures, tool interfaces and feedback mechanisms, which enables them to perform multi-step tasks with reduced human involvement [2]. This architecture is security-relevant because each component creates a different point of control, failure and attack, especially when memory, tool use and environmental feedback are combined into one workflow [17].

The literature also shows that agentic AI should not be treated as a single technical category. Ali et al. distinguish between symbolic or classical agentic systems and neural or generative agentic systems, which is important because the governance risks of LLM-based agents are different from those of rule-based automation [1]. Symbolic agents may be more predictable because their behavior is shaped by explicit rules, while neural agents are more flexible but also more difficult to constrain because their behavior is generated through probabilistic reasoning [1]. Current agentic frameworks such as CrewAI, LangGraph, AutoGen, Semantic Kernel, Google ADK and MetaGPT also differ in orchestration, memory, communication, modularity and guardrail support, which makes standardized enterprise

governance difficult [21]. This lack of architectural consistency matters because organizations may deploy agents with different levels of observability, access control and safety enforcement while assuming that all "AI agents" can be governed in the same way [21].

4.2. The Expanding Security Threat Landscape of AI Agents

The threat landscape of agentic AI is broader than traditional LLM security because agents can convert model reasoning into external actions [6]. He et al. show that LLM agents inherit ordinary LLM risks but intensify them through reasoning, tool usage and environmental interaction, which can create cascading consequences beyond text generation [6]. Chhabra et al. classify agentic AI threats across prompt injection, tool abuse, memory poisoning, agent identity misuse, multi-agent threats, environmental risks and governance concerns, showing that agent security is not reducible to prompt safety alone [17]. This is important because many enterprise AI controls still focus on model output moderation, while the more serious risk may occur when an agent calls a privileged tool or acts on manipulated context [17].

A system-level view is also necessary because agentic risk is shaped by design choices. Kim et al. identify agent design dimensions such as input trust, access sensitivity, workflow, action, memory, tool and user interface, which together define the agent's attack surface [15]. This perspective is useful because an agent with low autonomy and read-only access does not create the same risk as an agent with persistent memory, sensitive data access and write privileges [15]. Lazer et al. further show that agentic AI creates dual-use cybersecurity dynamics because the same capabilities that support SOC automation, threat hunting and incident response can also support reconnaissance, exploitation, social engineering and oversight evasion [4]. Zhang et al. demonstrate that vulnerabilities can occur across system prompts, user prompts, planning, memory retrieval and tool execution, which confirms that agentic AI requires workflow-level security rather than isolated model-level protection [11].

4.3. Tool Use, Prompt Injection and Privilege Escalation

Tool use is the point where AI output becomes enterprise action. AgentDojo shows that agents operating in dynamic tool-calling environments can be attacked through untrusted data returned by tools, which can cause the agent to leak information or perform unintended actions [10]. This finding is important because enterprise agents commonly operate on emails, documents, web pages, tickets and databases that may contain attacker-controlled content [10]. Agent Security Bench extends this concern by formalizing multiple attack types, including direct prompt injection, indirect prompt injection, memory poisoning, Plan-of-Thought backdoors and mixed attacks across several scenarios [11]. This suggests that agent security cannot be based only on detecting obvious malicious prompts, because attacks may be embedded in memory, observations, retrieved records or workflow context [11].

The deeper problem is that agents may process trusted instructions, retrieved records, and tool outputs through the same generative workflow [9]. Alpay and Alpay [9] formalize this problem through intent-to-execution noninterference and distinguish descriptive prompt constraints from enforceable policy projection, capability restriction, and output validation [9]. Kim, Choi, and Lee [12] frame this weakness as a privilege-escalation problem because untrusted data may influence privileged tools when the agent cannot reliably separate instruction flow from data flow. This argument is central to enterprise governance because attackers may not need to compromise the IAM system directly if they can influence the agent that already holds access to enterprise tools [12].

Recent defenses increasingly focus on deterministic control at the tool boundary. Progent enforces least privilege by checking every tool call against a symbolic policy and re-

quiring explicit approval when privilege expansion is needed [13]. CLAWGUARD similarly argues that tool-call boundaries should be governed through deterministic and auditable enforcement before real-world effects are produced [22]. Bhagwatkar et al. propose a tool-input minimizer and tool-output sanitizer, which is useful because it controls both unnecessary data exposure and malicious content returning to the agent [14]. AgentSpec strengthens the same direction by externalizing runtime safety constraints from the LLM, making enforcement more inspectable, modular and auditable [23].

4.4. Identity, Access Management and Zero Trust for Agentic Systems

Traditional IAM is not sufficient for agentic AI because it was designed mainly for human users, applications and relatively static machine identities [8]. Huang et al. argue that OAuth 2.0, OpenID Connect (OIDC), and Security Assertion Markup Language (SAML) have limitations in multi-agent systems because they provide coarse-grained controls, assume single-entity access and lack the contextual adaptability needed for dynamic agent behavior [8]. Their proposed direction includes verifiable agent identities, decentralized identifiers, verifiable credentials, agent naming, fine-grained access control, global session management and policy enforcement [8]. This is valuable because it shifts IAM from static authentication toward dynamic governance of agent capability, provenance, behavioral scope and revocation [8].

Syros et al. offer a more concrete architecture through Security Architecture for Governing Agentic systems (SAGA), which uses agent registration, provider-maintained metadata, user-defined contact policies and cryptographic access-control tokens for inter-agent communication [7]. SAGA is important because it treats agent identity and communication as governable objects rather than assuming that agents can freely discover and interact with each other [7]. Kurtz and Krawiecka extend the identity argument by describing AI agents as active, identity-bearing enterprise actors with access footprints, behavioral trajectories and accountability structures [5]. Their work is useful, but it also reveals the need for a narrower enterprise cybersecurity framework that connects machine identity governance with runtime tool control and delegated authority [5]. Raza et al. add that agentic AI requires lifecycle governance, security, privacy, explainability and operational risk management, which aligns with Zero Trust but requires more agent-specific control design [3].

4.5. Continuous Governance, Oversight and Auditability

Agentic AI governance must be continuous because agents operate through runtime decisions, tool interactions and changing context rather than through static deployment conditions [3]. Raza et al. adapt Trust, Risk and Security Management to agentic systems by emphasizing explainability, ModelOps, lifecycle governance, privacy, security and regulatory compliance [3]. This is useful, but enterprise cybersecurity also requires controls that show who authorized the agent, what tools it used, what data influenced the action and whether the action remained within approved intent [9]. Schmitz et al. show that agent oversight requires cross-departmental implementation, comprehensive evaluation, enhanced security protocols, operational visibility and systematic auditing [16]. Their public-sector analysis is relevant to enterprises because it shows that episodic approvals are weak when systems act continuously and require integrated supervision [16].

The literature therefore points toward a governance model that integrates identity, authority, runtime control, and audit evidence. Runtime-enforcement research shows how tool actions can be constrained, but it does not fully address ownership, accountability, and lifecycle governance [23]. IAM research shows how agent identity may be represented, but it does not fully address prompt injection, memory poisoning, and tool-boundary manipu-

lation [8]. The governance literature highlights oversight and accountability but requires operational translation into agent registration, delegated-authority chains, least-agency levels, tool-call policies, audit registers, and revocation procedures [3]. The literature review supports the central claim that agentic AI should be governed as a dynamic non-human identity whose security depends on identity assurance, intent-bound access, runtime tool control and accountable evidence across the full agent lifecycle.

5. Research Gaps and Framework Derivation

The literature indicates that agentic AI security is not only a problem of prompt injection, unsafe outputs, or vulnerable tools, but a governance problem created by the transition from relatively static machine identities to dynamic non-human actors. Existing work addresses important components of identity, delegation, prompt and tool security, runtime enforcement, and oversight, but these controls are generally treated as separate problems. To make the transition from the literature review to the proposed framework transparent, Table 1 maps the principal literature-derived governance issues to the corresponding remaining governance requirements and shows how these requirements inform the eight AIGATE layers.

Table 1. Literature-derived governance requirements and derivation of the eight AIGATE layers.

Literature Evidence/ Existing Emphasis	Remaining Governance Requirement	Derived AIGATE Response	Layer
Machine identity and agent-registration research establishes identity, provenance, and fine-grained access requirements [5,7,8].	Enterprise agents still require a persistent governance record linking identity to owner, purpose, configuration, scope, and risk.	Agent Identity Registration	1
Agent governance and machine identity research identifies changing tools, models, ownership, and deployment conditions [3,5,16].	Identity and authority must be governed from request and approval through monitoring, suspension, and retirement.	Lifecycle Governance	2
SAGA and governance research address agent registration, interaction, and authorization [7,16].	Authority provenance must remain visible across human → agent → sub-agent → tool/action chains.	Delegated Authority Mapping	3
AgentDojo, AgentSecBench, and Prompt Flow Integrity show that legitimate permissions can be redirected by untrusted context [9,10,12].	Authorization must consider the current task purpose and execution necessity, not identity and permission alone.	Intent-Bound Access	4
Least-privilege and runtime-security work constrains permissions and tool use [4,13].	Access restriction alone does not determine how independently an agent may decide, sequence, escalate, or execute actions.	Least Agency and Least Privilege	5
Progent, CLAWGUARD, input/output minimization, and AgentSpec demonstrate the value of external runtime constraints [13,14,22,23].	High-impact tool/API actions require deterministic checking before external effects occur.	Runtime Tool-Call Control	6

Table 1. *Cont.*

Literature Evidence/ Existing Emphasis	Remaining Governance Requirement	Derived AIGATE Response	Layer
The oversight and runtime-security literature emphasizes visibility, auditability, and policy enforcement [5,16,23].	Organizations need reconstructable evidence connecting requester, agent, intent, policy decision, approval, tool call, and outcome.	Audit Evidence and Accountability	7
The identity, runtime-control, and governance literature recognizes revocation, incident response, and lifecycle risk [3,8,16].	Compromised or unsafe agents require rapid containment followed by controlled recovery to a trusted operating state.	Revocation and Resilience	8

Table 1 makes the synthesis-to-framework transition explicit. AIGATE does not claim that each underlying security mechanism is individually novel; rather, its contribution is to organize these literature-derived requirements into a connected enterprise governance architecture spanning identity, authority, intent, autonomy, runtime enforcement, evidence, and recovery.

The provenance of the framework elements can be distinguished at three levels. First, identity registration, lifecycle governance, delegated authorization, least-privilege enforcement, runtime tool controls, auditability, and revocation draw directly on mechanisms or requirements established in prior research [5,7,8,13,16,22,23]. Second, AIGATE provides the synthesis that connects these previously fragmented mechanisms across a single governance lifecycle linking identity, authority, intent, autonomy, enforcement, evidence, and recovery. Third, AIGATE introduces specific normative operationalizations within this integrated architecture, particularly the four-condition intent-bound access test and the six-level autonomy ladder used to operationalize least agency. The exact six-level ladder is therefore proposed in this paper as an AIGATE normative taxonomy rather than as an established taxonomy derived from a single prior source.

6. Proposed AIGATE (Agentic Identity Governance, Authority, Tool-Control and Evidence) Framework

6.1. Framework Overview

The proposed AIGATE is designed to govern AI agents as active enterprise identities rather than as ordinary applications, chatbots or background automation scripts [5]. The framework extends traditional machine identity governance by adding controls for autonomy, delegated authority, task intent, runtime tool use, audit evidence and emergency revocation [8]. Its central assumption is that an AI agent becomes a cybersecurity-relevant identity when it can access enterprise resources, interpret instructions, invoke tools or produce operational consequences [7]. The framework therefore combines identity governance, Zero Trust access logic, runtime enforcement and accountability into one layered model [3].

Threat Model and Trust Assumptions

AIGATE assumes that an enterprise-managed agent may hold legitimate credentials and approved tool access while still being redirected by malicious users, attacker-controlled documents or tool outputs, poisoned memory, compromised peer agents, or configuration errors. Protected assets include enterprise data, credentials, APIs, workflow state, delegated authority, and audit evidence. Prompts, retrieved content, tool responses, sub-agents, and network location are therefore not treated as inherently trustworthy. The framework does not rely on the LLM alone to distinguish trusted instruction from untrusted data; privileged actions must instead pass external policy checks binding registered identity, delegated

authority, approved intent, autonomy level, and tool/data scope before execution [10,12,13]. Model-weight compromise and compromise of the underlying model provider are outside AIGATE's direct enforcement scope.

AIGATE's trusted computing base (TCB) comprises the enterprise identity provider and credential service, agent registry, authority and delegation records, policy-decision function, policy-enforcement point, authorized human approval channel, and protected audit-evidence repository. These components are assumed to authenticate principals correctly, preserve approved identity and authority state, enforce policy without unauthorized modification, and produce trustworthy evidence. Downstream tools and APIs are treated as execution targets rather than governance authorities; their responses remain potentially untrusted and should not independently alter agent authority. If the registry, credential service, policy-decision or policy-enforcement component, approval channel, or audit repository cannot be trusted, high-impact agent actions should fail closed or be suspended until trust is restored.

Revocation should propagate across the full delegated-authority graph rather than terminating only the parent agent. When an agent or delegated capability is revoked, the registry should mark the affected identity or capability as inactive, the credential service should invalidate active tokens and session credentials, and the policy-enforcement point should deny subsequent tool calls. Child-agent delegations and derived capabilities should be withdrawn, active workflows paused or terminated where feasible, cached credentials invalidated, and the revocation event recorded in the audit-evidence register. Recovery should require explicit reauthorization or re-registration rather than automatic continuation from the previously trusted state.

AIGATE applies Zero Trust as an action-governance principle rather than merely a network-access principle. Consistent with NIST SP 800-207, no agent is treated as implicitly trustworthy because it has already authenticated or operates inside an enterprise environment; instead, identity, authority, context, and policy are evaluated before access or action is permitted [24]. Table 2 therefore summarizes the eight AIGATE layers and maps each layer to its corresponding Zero Trust control role.

Table 2. AIGATE Framework and its alignment with Zero Trust control logic.

Layer	Name	Purpose	Zero Trust Role
1	Agent Identity Registration	Assigns every agent a unique identity, owner, business purpose and risk class	Verify the acting non-human identity and bind it to an accountable owner and approved purpose.
2	Lifecycle Governance	Controls agent request, approval, deployment, monitoring, suspension and retirement	Continuously re-evaluate trust as model, tools, context, purpose, and risk change.
3	Delegated Authority Mapping	Records human-to-agent, agent-to-agent and agent-to-tool authority chains	Make delegated authorization explicit rather than inheriting implicit trust from a user or parent agent.
4	Intent-Bound Access	Links access decisions to task intent, approved purpose, data scope and tool scope	Evaluate each action against current identity, authority, task purpose, and context.
5	Least Agency and Least Privilege	Limits both permissions and autonomy so agents do not receive unnecessary operational freedom	Grant only the minimum access and minimum autonomy necessary for the approved task.

Table 2. *Cont.*

Layer	Name	Purpose	Zero Trust Role
6	Runtime Tool-Call Control	Enforces policy at tool, API and action boundaries before external effects occur	Enforce the policy decision at the tool/API boundary before an external action occurs.
7	Audit Evidence and Accountability	Captures traceable evidence of decisions, tool calls, approvals, exceptions and outcomes	Continuously monitor decisions and actions and preserve evidence for review and accountability.
8	Revocation and Resilience	Provides kill switch, token expiry, rollback, credential rotation and incident response controls	Revoke sessions/credentials, contain unsafe behavior, and restore operation only after trust is re-established.

The overall structure of the proposed framework is presented in Figure 2, which integrates the eight governance layers into a single model for securing AI agents as dynamic non-human identities in enterprise cybersecurity. The figure shows that the framework progresses from foundational identity and lifecycle controls toward decision and access controls, and finally toward assurance, accountability and recovery controls [8].

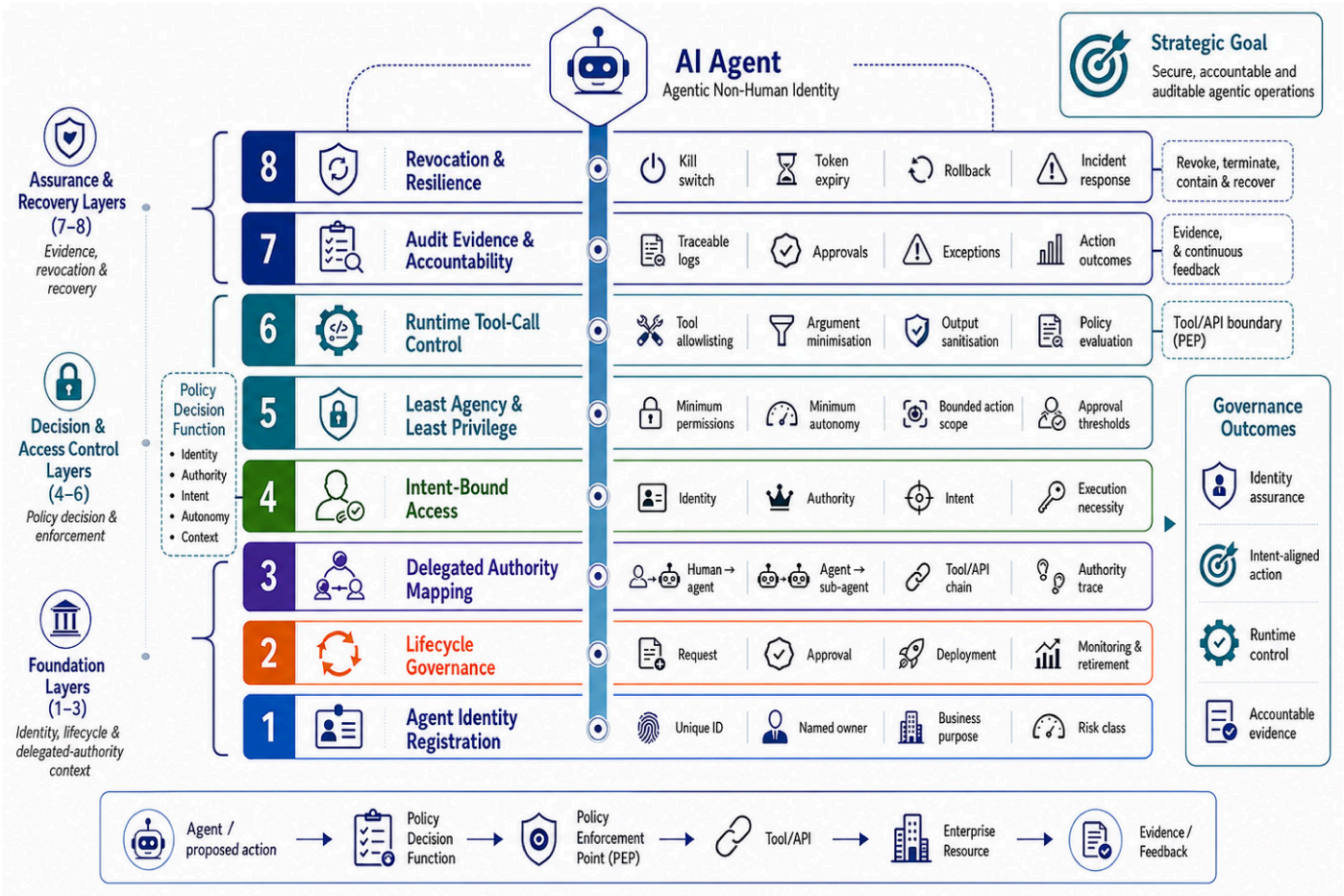


Figure 2. Eight-layer AIGATE governance and runtime enforcement architecture.

Figure 2 presents AIGATE as both a layered governance framework and a reference architecture for runtime action control. Layers 1–3 provide the identity, lifecycle, and delegated-authority context used in access decisions [5,7]. Layers 4–5 form the policy-

decision function by evaluating registered identity, delegated authority, task intent, access scope, and permitted autonomy [9,13]. Layer 6 acts as the policy enforcement point at the tool/API boundary, where a proposed action can be allowed, blocked, modified, or escalated before an external effect occurs [13,22]. Layers 7–8 provide evidence feedback, revocation, containment, and controlled recovery [8,16]. This arrangement adapts the Zero Trust policy-decision/policy-enforcement logic to agentic actions rather than assuming that prior authentication creates continuing trust [24]. The figure therefore illustrates the paper's core argument that secure agentic AI governance requires more than authentication or isolated technical safeguards; it requires a layered model linking identity assurance, delegated authority, intent-aligned execution, runtime control, accountable evidence, and revocable authority across the agent lifecycle [3].

The following subsections explain each layer of the framework in detail, beginning with agent identity registration as the foundational governance requirement [7].

6.2. Layer 1: Agent Identity Registration

The first layer requires every enterprise AI agent to be formally registered before it is connected to tools, data sources or business workflows [7]. This registration should not be limited to a technical name or API credential, because the security issue is not only authentication but also ownership, purpose, risk and accountability [5]. Agent identity registration creates an accountability anchor by linking the agent to a named human owner, approved business function, model configuration, permitted tools, data scope and autonomy level [8]. This is essential because agentic AI systems may act dynamically and cannot be governed safely if the enterprise does not know which agent exists, why it exists, what it can access and who is responsible for its behavior [3].

To make agent identity registration operational without conflating persistent identity with individual runtime executions, the framework requires a minimum identity register that separates the registered agent identity from its deployment, runtime session, credential, delegation-chain, and task identifiers, while also recording ownership, purpose, tool/data scope, autonomy, and risk classification as presented in Table 3.

Table 3. Minimum agent identity register.

Field	Example
Registered agent identity	Persistent governed identifier for the agent or workload across approved deployments [7].
Deployment/configuration ID	Identifier linking the approved model version, orchestration configuration, tool set and deployment state.
Runtime instance/session ID	Ephemeral identifier for a specific running agent instance or session.
Credential/token ID	Identifier or fingerprint of the credential/token used by the instance; secret credential values should not be stored in the register.
Delegation-chain ID	Correlation identifier linking the human requester, parent agent, sub-agent and delegated capability chain.
Task/request ID	Identifier linking the runtime action to the initiating request and approved task purpose.
Agent owner	Named human, team or business owner responsible for the agent [5].
Business purpose	SOC triage, HR support, DevOps automation or compliance mapping [4].
Model/version	Model, configuration, orchestration framework and deployment version [21].
Tools allowed	Security information and event management (SIEM), ticketing, email, database, browser, code tool or workflow API [10].

Table 3. *Cont.*

Field	Example
Data scope	Logs only, HR records, finance data, customer records or repository data [15].
Autonomy level	Levels 0–5: No agentic action, recommendation-only, draft action, human-approved action, conditional autonomous action, or full autonomous action, as defined in Section 6.6.
Risk class	Organization-assigned risk classification based on data sensitivity, tool/action impact, autonomy, and business consequence (e.g., low, medium, high, or critical) [3].

Table 3 shows that agent registration is not simply a naming exercise; it creates the accountability anchor needed to govern the agent's access, autonomy and operational behavior throughout its lifecycle.

6.3. Layer 2: Lifecycle Governance

The second layer controls the agent from initial request to retirement, because agentic risk changes as tools, data sources, model versions and business use cases evolve [3]. Lifecycle governance should begin with a formal request, followed by risk assessment, owner approval, tool and data approval, deployment, monitoring, access review, incident handling, suspension and retirement [16]. This prevents agents from becoming unmanaged non-human identities that remain connected to systems after their business purpose, owner or risk profile has changed [5]. Lifecycle governance also supports Zero Trust because trust in an agent should not be permanent but continuously re-evaluated according to behavior, scope and context [8].

Once an agent has been registered, it should be governed through a lifecycle process that controls how the agent is requested, assessed, approved, deployed, monitored, reviewed, suspended and retired as mentioned in Table 4.

Table 4. Agent lifecycle governance stages.

Stage	Governance Purpose
1. Request	Defines the business need and proposed agent function
2. Risk Assessment	Evaluates data sensitivity, tool access, autonomy and business impact
3. Owner Approval	Assigns human accountability before deployment
4. Tool/data Approval	Confirms which tools, APIs and datasets the agent may use
5. Deployment	Connects the agent under approved identity, access and monitoring conditions
6. Monitoring	Tracks behavior, tool calls, exceptions and policy violations
7. Access Review	Revalidates permissions, autonomy and continued business need
8. Incident Handling	Defines response steps for unsafe, compromised or abnormal behavior
9. Suspension	Temporarily disables the agent or selected capabilities
10. Retirement	Removes access, archives evidence and closes the agent identity

Table 4 demonstrates that agent governance must be continuous rather than one-time, because changes in tools, data access, model configuration or business purpose can alter the agent's risk profile after deployment.

6.4. Layer 3: Delegated Authority Mapping

The third layer records the authority chain behind agentic action, because agentic systems may act directly, call tools or delegate subtasks to other agents [7]. A basic access log may show that an API was called, but it may not show whether the action originated from a human request, a parent agent, a child agent, a tool observation or an injected instruction [9]. Delegated authority mapping should therefore preserve the full chain: Human Requester → Agent Identity → Sub-agent Identity → Tool/API → Data/Resource → Action Outcome [5]. This mapping is a governance requirement because accountability becomes weak when organizations can observe the final action but cannot reconstruct the authority path that produced it [16].

Figure 3 illustrates the delegated authority chain that must be captured when an AI agent acts across enterprise systems. Unlike ordinary access logs, which may only record the final API call or resource access, agentic governance must preserve the full path from the human requester to the primary agent, any delegated sub-agent, the tool or API interface, the enterprise resource and the final action outcome [7].

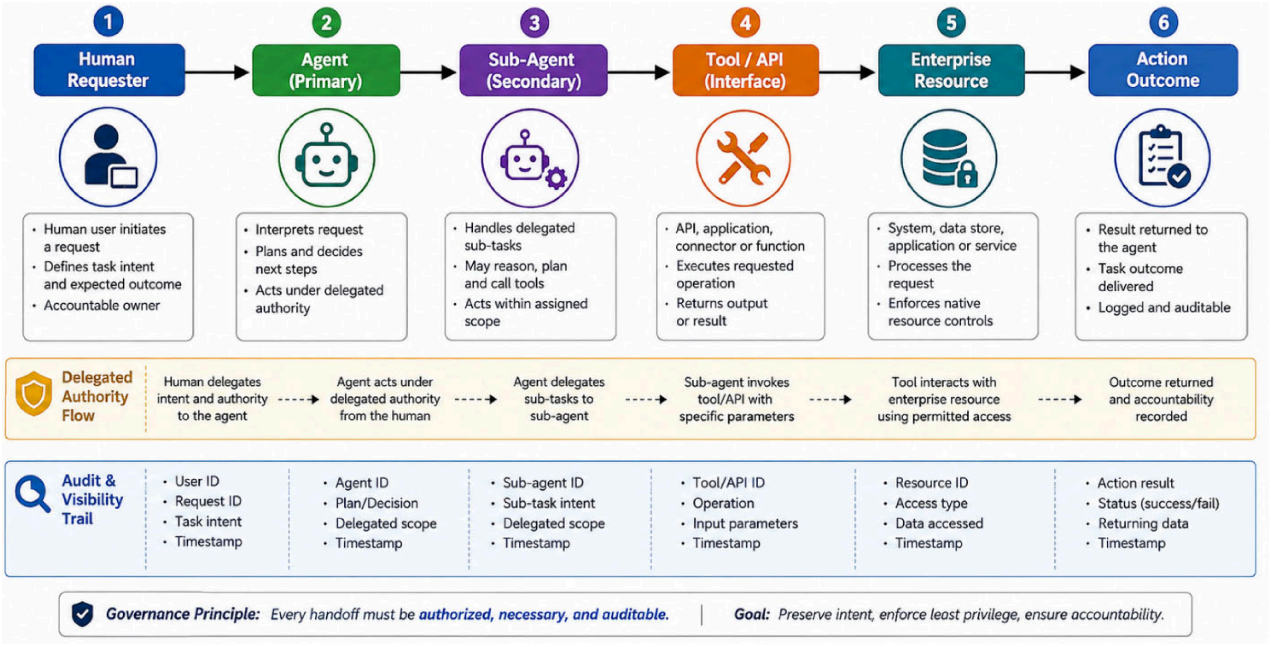


Figure 3. Delegated authority and accountability chain in agentic enterprise systems.

Figure 3 shows that agentic action is not a single access event but a sequence of delegated decisions and operational handoffs. The human requester initiates the task and defines the intended outcome, while the primary agent interprets the request, plans the next steps and acts under delegated authority. Where complex tasks are divided, a sub-agent may handle a narrower delegated function before invoking a tool, connector or API. The tool then interacts with an enterprise resource such as a database, application, workflow system or service, producing an action outcome that should be returned, logged and made auditable. This chain is important because accountability can be weakened if the organization records only the final tool action without preserving the user intent, agent identity, sub-agent scope, tool parameters, resource accessed and final result [5]. The

governance principle is that every handoff in the chain should be authorized, necessary and auditable, so that delegated authority does not become an uncontrolled privilege path [9].

6.5. Layer 4: Intent-Bound Access

The fourth layer is one of the framework's central contributions because agentic AI requires access decisions to be bound to task intent, not only to identity and permission [9]. Traditional access control asks whether an entity has permission to use a resource, while intent-bound access asks whether the current action is necessary for the approved task [9]. This is important because an agent may legitimately have access to email, files or ticketing systems, yet still misuse that access if a malicious document, email or tool output changes the task direction [10]. Intent-bound access therefore requires four AIGATE conditions to align before action is permitted, as explained below.

- **Identity**: Which agent is acting and under which registered identity.
- **Authority**: Which human owner, requester or policy authorized the action.
- **Intent:** What task is the agent approved to complete in this session.
- **Execution:** Is the proposed tool call necessary and proportionate for the approved task.

A compact machine-oriented example illustrates how these conditions can be enforced externally. Consider the policy context *{agent_id = AGT-SOC-017; requester = analyst-42; task_id = INC-4821; approved_task = investigate suspicious privileged sign-in; permitted_tool = IAM; resource_scope = account ACC-204; action_scope = read account and authentication state; autonomy = Level 2; validity = 30 min; approval = not required for read-only queries; evidence = record policy input, decision, tool call and outcome}*. A proposed *query_account_status* action matching these attributes is ALLOW. If the same agent proposes *disable_account*, the policy function returns ESCALATE because the action exceeds the approved read-only scope and requires Level 3 human approval. An expired validity period or a mismatch in agent, requester, task, resource or tool results in BLOCK, while unnecessary tool arguments may be MODIFIED/MINIMIZED before execution. This example illustrates how approved intent can be translated into deterministic runtime policy without relying on the language model alone.

6.6. Layer 5: Least Agency and Least Privilege

The fifth layer extends the established least-privilege principle [13] by adding least agency as an AIGATE governance principle. Least privilege asks what minimum access is needed, while least agency asks what minimum autonomy is needed. This distinction matters because an agent may have limited access but still cause harm if it is allowed to decide, sequence and execute actions without sufficient approval gates [4]. In this framework, agent autonomy can be understood as a progression from no agentic action to recommendation-only support, draft action, human-approved action, conditional autonomous action and, finally, full autonomous execution. Within AIGATE, high-impact actions such as disabling accounts, transferring funds, changing code, deleting records or submitting compliance evidence should normally remain at recommendation-only, draft, human-approved or conditional autonomy levels rather than full autonomy.

The autonomy level is assigned during agent approval by the business owner together with the relevant security or risk owner and is revalidated during access review. The default is the lowest level capable of completing the approved task. Recommendation or draft modes are appropriate where information is uncertain or consequences are high; human-approved action is required where an agent changes security state, code, financial value, identity, or regulated data; and conditional autonomy is appropriate only where predefined policy thresholds, approved tools/data, monitoring, and rollback controls are in place. Autonomy must be reduced, suspended, or revoked when task scope changes,

new tools or sensitive data are introduced, risk classification increases, anomalous behavior or policy violations occur, or incident response is triggered. Any increase in autonomy requires documented re-approval.

The relationship between least privilege and least agency is illustrated in Figure 4. Least privilege, consistent with Progent [13], controls what an agent is permitted to access or execute, while AIGATE's least-agency principle controls how independently the agent may decide and act within that access scope.

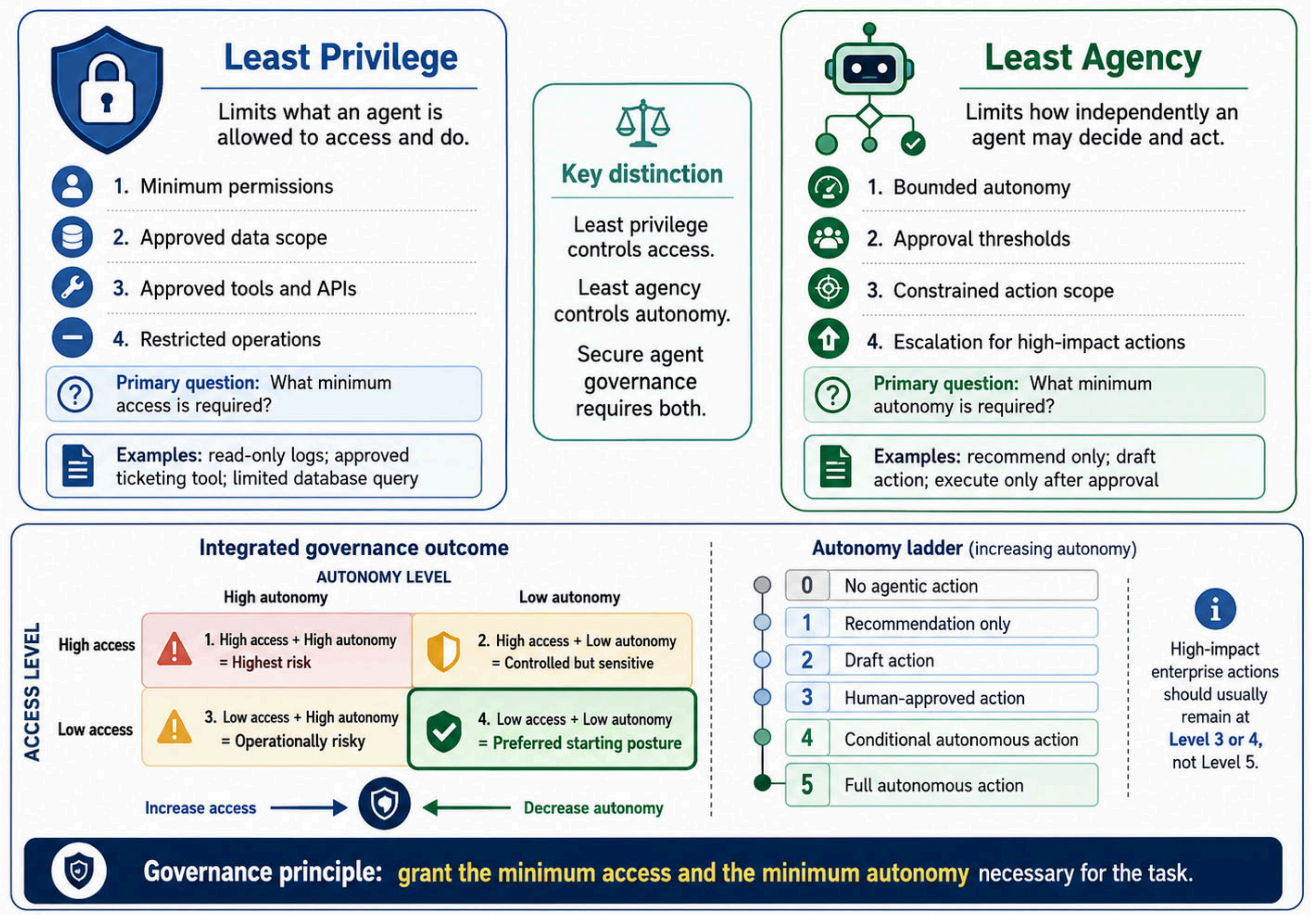


Figure 4. Least privilege versus least agency models.

Figure 4 shows that secure agent governance requires both minimum access and minimum autonomy. Least privilege, consistent with privilege-control work such as Progent [13], limits the tools, data, APIs, and operations an agent may use, whereas AIGATE's least-agency principle limits how independently the agent may decide and act within that access scope. The model highlights that high access combined with high autonomy creates the highest-risk posture because the agent can both reach sensitive resources and act with limited human intervention [4]. In contrast, low access combined with low autonomy represents the preferred starting posture because it gives the agent only the access and independence necessary for the approved task [3]. As an AIGATE normative recommendation, high-impact enterprise actions, such as account disablement, financial approval, code deployment or sensitive data modification, should normally remain at human-approved or conditional autonomy levels rather than full autonomous execution. Within AIGATE, least agency therefore extends least-privilege reasoning from access control to the agent's discretion, sequencing, escalation, and operational judgment. The high/low labels in Figure 4 indicate relative access and autonomy positions for conceptual comparison and do not represent calibrated enterprise risk thresholds.

6.7. Layer 6: Runtime Tool-Call Control

The sixth layer controls the point where agent reasoning becomes operational action [10]. Tool-call control is essential because external effects occur when an agent invokes an API, sends an email, queries a database, executes code, updates a ticket or changes a workflow state [13]. This layer therefore combines tool allowlisting and policy evaluation [13,22,23] with argument minimization and output sanitization [14], together with approval gates and audit logging for high-impact actions. Tool allowlisting restricts the agent to approved tools, while argument minimization prevents unnecessary sensitive data from being passed into external systems. Output sanitization reduces the risk that malicious tool responses or indirect prompt injections re-enter the agent's reasoning process. Policy evaluation checks whether each proposed tool call matches the approved task, data scope and autonomy level. Approval gates route uncertain or high-impact actions to a human decision point, while audit logging records allowed, blocked, escalated and rejected actions. This layer is strongly aligned with CLAWGUARD because deterministic enforcement at the tool-call boundary can block unsafe actions before real-world effects are produced [22].

6.8. Layer 7: Audit Evidence and Accountability

The seventh layer creates a structured audit evidence register because ordinary system logs are not enough to explain agentic decisions [23]. Agent accountability requires evidence of the registered agent identity, owner, user request, task intent, data accessed, tool invoked, policy decision, human approval, output, exception and revocation record [5]. This evidence is necessary because an enterprise must be able to reconstruct whether an action was authorized, intent-aligned, proportionate and compliant [16].

To make runtime accountability operational, the framework requires an Agentic Audit Evidence Register that captures not only what action occurred, but also which agent acted, why it acted, what data influenced the action, which tool was used, what policy decision was applied and whether human approval was required.

Table 5 shows that agentic auditability must go beyond ordinary access logs by preserving the full evidence chain from user intent and agent identity to tool invocation, policy decision, action outcome and revocation record.

Table 5. Agentic audit evidence register.

Evidence Item	Why It Matters
Registered agent identity and runtime instance/session ID	Links the persistent governed actor to the specific execution that produced the event
Delegation-chain and task/request IDs	Correlate the action with its initiating request, delegated authority path and approved task
Human owner	Assigns organizational accountability
User Request	Shows the initiating instruction
Task Intent	Defines the authorized purpose of action
Data Accessed	Shows information exposure and retrieval scope
Tool Invoked	Shows the operational action channel
Policy Decision	Explains why access was allowed, blocked or escalated
Human Approval	Demonstrates oversight for high-impact actions
Output/Action	Records the organizational consequence
Exception	Captures deviation, violation or blocked behavior
Revocation Record	Shows containment and closure of unsafe access

Because the Agentic Audit Evidence Register may contain sensitive prompts, resource identifiers, data-access information, approval decisions and security events, the evidence itself requires protection. Records should be append-only or otherwise tamper-evident, integrity-protected through cryptographic hashing, signing or equivalent controls, encrypted in transit and at rest, and accessible only to authorized roles under least-privilege principles. Retention periods should be defined according to legal, regulatory, incident-response and organizational requirements, while privacy minimization or redaction should prevent unnecessary prompt, payload or personal-data content from being retained. Chain-of-custody metadata should record the creation, access, export and authorized amendment of evidence; where correction is required, the original record should be preserved and the change recorded as an auditable version rather than silently overwritten.

6.9. Layer 8: Revocation and Resilience

The final layer is designed to enable rapid containment of unsafe, compromised, or unnecessary agents [8]. Revocation must be designed into agent governance because agentic systems may operate with tool access, memory, delegated authority and live sessions that cannot be left active after abnormal behavior is detected [5]. The AIGATE design therefore includes a kill switch, token expiry, session termination, credential rotation, rollback where feasible, forensic preservation, and post-incident review. These controls are important because agentic AI incidents may unfold through delegated authority chains and runtime tool interactions rather than through a single static credential or application event [7].

In this layer, revocation and resilience perform different but connected functions. Revocation is the containment mechanism: it stops or narrows the agent's authority through session termination, token expiry, credential withdrawal, or capability suspension. Resilience concerns restoration to a trusted operating state after containment through evidence preservation, rollback of reversible actions, credential rotation, clean session or configuration recovery, post-incident review, and controlled reauthorization before the agent resumes operation. Revocation therefore limits continuing harm, whereas resilience provides the recovery and safe-return capability.

Together, the eight AIGATE layers provide a governance model that treats agentic AI as a dynamic non-human identity across its full operational lifecycle. The framework is broader than tool-level security because enterprise risk does not arise only from prompt injection or unsafe tool calls, but from the combination of identity, authority, autonomy, runtime action and accountability [15]. It is also more specific than general AI governance because it translates trust, risk and oversight principles into concrete controls for agent registration, delegated authority, intent-bound access, runtime enforcement, audit evidence and revocation [3]. This makes the framework suitable for enterprise cybersecurity settings where AI agents may support SOC automation, DevOps workflows, finance processes, HR operations and compliance evidence generation.

7. Scenario-Based Application and Qualitative Risk Analysis

The framework is applied to three high-impact enterprise scenarios to examine its control coverage and the unsafe action paths it is designed to constrain. These scenarios are illustrative analytical applications rather than empirical validation. Risk is characterized qualitatively using four factors derived from the conceptual synthesis: (1) sensitivity of affected data or resources, (2) exposure and impact of available tools, (3) agent autonomy, and (4) influence of untrusted inputs. These factors are not multiplied or converted into a numerical risk score because this study does not establish calibrated scales, weights, or empirical thresholds. AIGATE does not prescribe universal categorical thresholds; organizations may map these factors to their established enterprise risk scales. Instead,

they provide a transparent heuristic for explaining why a scenario has elevated agentic risk and for identifying the AIGATE controls expected to intervene. Quantitative calibration and effectiveness testing are reserved for future empirical evaluation.

7.1. Scenario 1: SOC Triage and Containment Agent

In this scenario, an AI security agent reads SIEM alerts, checks endpoint telemetry, correlates authentication events and recommends or triggers containment actions. The scenario presents elevated risk because the agent combines sensitive security telemetry and privileged security-tool access with substantial autonomy and potentially noisy or adversarial input [4]. The critical failure path occurs when poisoned logs, false correlation or attacker-shaped evidence causes the agent to disable a legitimate privileged account, isolate a production endpoint or escalate a false incident. This is not merely a detection error, because an agentic SOC system can convert uncertain interpretation into operational containment through endpoint detection and response (EDR), IAM or ticketing tools [17].

The AIGATE Framework addresses this risk by requiring registered agent identity, declared SOC purpose, approved SIEM/EDR tool scope and a delegated authority map showing whether containment was requested by a human analyst, parent agent or automated policy. Intent-bound access requires identity, delegated authority, approved task intent, and execution necessity to align before containment is permitted, meaning the agent may query telemetry but cannot disable accounts unless the action matches the approved incident objective [9]. Least agency places account disablement and endpoint isolation at Level 3 or Level 4, meaning human approval or strict conditional rules are required for high-impact containment. Runtime tool-call control further requires allowlisted EDR actions, argument minimization, policy evaluation and audit logging before the containment tool executes [22]. This scenario illustrates AIGATE's control logic: useful triage remains available, while the transition from analysis to containment is subject to explicit authority, intent, autonomy, and runtime-policy checks.

7.2. Scenario 2: DevOps Code and Deployment Agent

In this scenario, an AI DevOps agent reads GitHub issues, analyzes repository files, edits code, runs tests, creates pull requests and may trigger deployment workflows. The scenario presents elevated risk because the agent combines sensitive source-code and secret access, powerful development and deployment tooling, substantial autonomy, and potentially untrusted repository content. A poisoned GitHub issue, malicious README, compromised dependency instruction or adversarial tool output may redirect the agent into executing unsafe commands, leaking secrets or merging insecure code. This risk is structurally similar to indirect prompt injection because attacker-controlled content enters the agent context as if it were task-relevant information [10].

Within AIGATE, this scenario is governed through tool and repository scoping, least privilege, least agency, and runtime policy enforcement. The agent may be authorized to read issues and draft code, but merge approval, deployment execution and secret access should require additional approval gates. Progent's logic is relevant here because tool calls should be checked against symbolic policy and privilege expansion should require explicit approval rather than occurring silently during execution [13]. Each proposed action should therefore use an approved tool and repository, match the approved task intent, and remain within the agent's assigned autonomy level. Tool-output sanitization and prompt-flow separation are also required because untrusted repository content should not be allowed to become an instruction path for privileged shell, continuous integration/continuous delivery (CI/CD) or deployment tools [12]. The scenario illustrates how AIGATE separates

productive coding assistance from privileged software-supply-chain actions by applying additional authority, autonomy, and runtime controls before merge or deployment.

7.3. Scenario 3: Finance Workflow and Payment Agent

In this scenario, an AI finance agent reads invoices, validates supplier information, checks purchase orders, prepares payment workflows and may initiate payment requests. The scenario presents particularly elevated risk because sensitive financial data, payment-capable tools, substantial autonomy, and untrusted external documents combine with potentially irreversible monetary consequences. The critical failure path occurs when a malicious invoice embeds instructions that cause the agent to change beneficiary details, bypass verification or initiate payment to an attacker-controlled account. This is especially dangerous because the harmful action may appear as a legitimate finance workflow rather than as a conventional malware event.

AIGATE requires the finance agent to have a registered identity, named business owner, approved payment purpose, restricted tool set and explicit autonomy level. Intent-bound access means that the agent can extract invoice data but cannot alter beneficiary details unless the action is consistent with verified supplier records and approved payment intent [9]. Least agency requires payment initiation, beneficiary modification and exception approval to remain at Level 3 or Level 4 rather than full autonomy. Runtime control should include payment-tool allowlisting, beneficiary verification, argument minimization, output sanitization, approval gates and audit evidence for every payment-relevant action [14]. The audit evidence register must capture the registered agent identity and runtime instance/session ID, human approver, supplier record, invoice source, tool invoked, policy decision and final payment status. This scenario illustrates how AIGATE separates document processing from financial authority and makes high-impact payment actions contingent on verified identity, task intent, policy enforcement, and accountable approval.

To make the scenario analysis systematic and comparable across domains, Table 6 summarizes the protected assets, principal failure paths, proposed high-impact actions, applicable AIGATE layers, expected policy outcomes, approval requirements and remaining limitations.

Table 6. Cross-scenario comparison of AIGATE control coverage.

Scenario/ Protected Asset	Untrusted Input or Failure Path	Proposed High-Impact Action	Applicable AIGATE Layers	Expected Policy Outcome	Approval Requirement	Remaining Limitation
SOC: Privileged accounts, endpoints and security telemetry	Poisoned logs, false correlation or attacker-shaped evidence	Disable account or isolate endpoint	1, 3, 4, 5, 6, 7, 8	Escalate or block unless authority, intent and policy conditions are satisfied	Human approval or predefined Level 4 conditional policy	False correlation and compromised telemetry/tools may still influence analysis
DevOps: Source code, secrets and CI/CD environment	Malicious issue, README, dependency instruction or tool output	Merge code, deploy software or access secrets	1, 3, 4, 5, 6, 7	Escalate high-impact actions; block policy mismatch; allow bounded read/draft operations	Human approval for merge, deployment or privilege expansion	Compromised repositories, dependencies or policy configuration remain residual risks
Finance: Financial data, supplier records and payment workflows	Malicious invoice or manipulated supplier information	Modify beneficiary details or initiate payment	1, 3, 4, 5, 6, 7, 8	Escalate or block unless supplier, intent and approval conditions are verified	Human approval for beneficiary changes, payment execution and exceptions	Upstream supplier fraud or compromise of the approval channel remains possible

8. Discussion

AIGATE's contribution is primarily integrative and architectural rather than a claim that each component control is individually new. Existing approaches provide strong but partial solutions. Huang et al. [8] focus on Zero Trust identity and fine-grained authorization; SAGA [7] addresses agent registration, lifecycle oversight, and inter-agent access; Progent [13], CLAWGUARD [22], and AgentSpec [23] provide forms of runtime constraint; and benchmark/security research characterizes prompt, memory, identity, and tool-related attack surfaces [9–12,15,17]. AIGATE links these otherwise fragmented elements into an enterprise identity-governance sequence spanning registration, lifecycle, delegated authority, intent, autonomy, runtime tool enforcement, evidence, and revocation/resilience.

The distinction between least privilege and least agency is particularly important within this integration. Least privilege constrains what resources and operations an agent can access, whereas least agency constrains how independently it can decide and act within that access. AIGATE operationalizes this distinction through the autonomy ladder and approval conditions in Section 6.6 and connects autonomy decisions to registered identity, delegated authority, task intent, runtime policy, and revocation. The contribution is therefore not the creation of every individual security mechanism, but the specification of how these mechanisms should interact when an AI agent becomes a delegated non-human enterprise actor.

Table 7 positions AIGATE against the principal technical approaches and practitioner references most relevant to its scope. The comparison is intended to clarify complementarity and the specific integration provided by AIGATE rather than imply that AIGATE replaces these mechanisms or standards.

Table 7. Positioning of AIGATE against existing agent-security approaches and established governance references.

Reference/Approach	Primary Contribution	Relationship to AIGATE
Huang et al. [8]	Zero Trust identity, verifiable credentials, fine-grained agent access	Provides identity/access foundations; AIGATE extends them into lifecycle, intent, autonomy, runtime action, evidence, and recovery.
SAGA [7]	Agent registration, lifecycle oversight, inter-agent access policies and cryptographic authorization	Closely related identity/delegation architecture; AIGATE broadens governance toward enterprise task intent, least agency, tool enforcement, audit evidence, and resilience.
Progent [13]	Programmable least-privilege control of tool calls	Supplies runtime privilege-control logic; AIGATE embeds such enforcement within ownership, delegation, intent, autonomy, and lifecycle governance.
CLAWGUARD [22]	Deterministic runtime control against indirect prompt injection	Complements AIGATE Layer 6; AIGATE additionally addresses identity, authority, lifecycle, evidence, and recovery.
AgentSpec [23]	Customizable runtime safety rules for LLM agents	Provides a mechanism for runtime constraints; AIGATE specifies the wider governance context in which such rules are authorized, audited, and revoked.
NIST SP 800-207 [24]	Zero Trust resource access, explicit authorization, policy decision and enforcement	Supplies the Zero Trust control logic adapted by AIGATE to delegated agent actions and tool/API execution.

Table 7. *Cont.*

Reference/Approach	Primary Contribution	Relationship to AIGATE
NIST AI RMF 1.0 [25]	Govern, Map, Measure, and Manage functions across AI risk management	Provides broad AI-risk governance outcomes; AIGATE supplies narrower agent-identity, authority, autonomy, and tool-control mechanisms.
OWASP Top 10 for Agentic Applications 2026 [26]	Agent-specific threats including goal hijacking, tool misuse, identity/privilege abuse, context poisoning, and rogue behavior	Provides agentic threat guidance; AIGATE organizes preventive governance and runtime controls around identity, authority, least agency, evidence, and revocation.
MITRE ATLAS [27]	Adversary tactics and techniques targeting AI-enabled systems	Provides an adversary knowledge base that can inform AIGATE threat modeling and future control testing; AIGATE is not itself a threat catalog.

These references also clarify the scope of AIGATE. It is not proposed as a replacement for Zero Trust architecture, AI risk management, agent-security benchmarks, or runtime enforcement technologies. Its narrower contribution is to connect these concerns through a governance model in which a registered agent acts under explicit delegated authority, bounded task intent and autonomy, externally enforced tool policies, reconstructable evidence, and revocable authority.

9. Conclusions, Limitations and Future Research

9.1. Conclusions

Agentic AI changes enterprise cybersecurity because AI systems are no longer limited to generating outputs; they can access systems, invoke tools, delegate tasks and produce operational consequences [17]. This shift makes traditional IAM and machine identity governance insufficient because AI agents may operate with changing context, autonomy, tool access and delegated authority during execution. The core argument of this paper is therefore that AI agents should be governed as dynamic non-human identities, not merely as applications, chatbots or automation scripts.

The proposed AIGATE Framework addresses this challenge by integrating agent identity registration, lifecycle governance, delegated authority mapping, intent-bound access, least agency, runtime tool-call control, audit evidence and revocation into one enterprise governance model. Its contribution is to connect fragmented research on agentic AI threats, prompt injection, runtime enforcement, machine identity and AI governance into a structured framework for controlling agentic action across the identity lifecycle. AIGATE illustrates how technical risks such as indirect prompt injection, memory poisoning, and unsafe tool use become governance risks when an authorized agent can convert manipulated context into enterprise action.

The paper also operationalizes least agency alongside least privilege by distinguishing restrictions on access from restrictions on independent decision-making and action. Least privilege limits access, while least agency limits how independently an agent may decide and act. This distinction strengthens Zero Trust for agentic AI by requiring every agentic action to be identity-assured, authority-bound, intent-aligned, minimally autonomous, runtime-enforced, auditable and revocable.

9.2. Limitations

The scope of this paper is conceptual and framework-oriented because agentic AI security, machine identity governance and runtime enforcement are still emerging areas.

A conceptual synthesis is appropriate at this stage because these research streams have not yet stabilized into a mature enterprise governance discipline [6]. Some of the reviewed literature is very recent, including advanced work on agent identity, prompt-flow integrity, runtime tool-call enforcement and machine identity governance. This reflects the early-stage nature of the field and the need for structured conceptual work before large-scale empirical validation can be meaningfully designed.

AIGATE is also enterprise-focused, so its direct application to SMEs, public-sector bodies or highly regulated critical infrastructure may require contextual adaptation. The paper does not implement a technical enforcement system because its purpose is to define the governance model that such systems should support. This means the framework should be understood as a governance architecture for structuring identity, authority, autonomy, runtime control and accountability, rather than as a finished software control mechanism.

The scenario analyses therefore illustrate conceptual applicability and expected control coverage rather than empirical effectiveness, quantified risk reduction, or generalizability. The qualitative risk factors used in Section 7 are descriptive heuristics and require empirical calibration before they can support quantitative risk scoring.

9.3. Future Research

Future research should empirically evaluate AIGATE through expert review, Delphi studies, enterprise case studies, and benchmark-based implementation in settings such as SOC automation, DevOps, and finance. Such evaluations should establish explicit baselines and measurable criteria, for example policy-violation rates, blocked unsafe tool calls, approval burden, audit completeness, revocation latency, and task utility. Technical work should also prototype the policy-decision and tool-enforcement functions represented in Figure 2 and test whether explicit identity, delegated-authority, intent, and least-agency controls reduce unsafe action paths relative to baseline agent configurations. Mapping to broader compliance regimes such as ISO/IEC 27001 [28] and sector-specific requirements can subsequently build on the Zero Trust and AI-risk positioning established in the present study.

Author Contributions: Conceptualization, M.N.; methodology, M.N. and R.S.; investigation, M.N. and R.S.; writing—original draft preparation, M.N. and R.S.; writing—review and editing, M.N. and R.S.; visualization, M.N. and R.S.; project administration, M.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable. This study is based on literature synthesis and conceptual framework development and did not involve human participants or the collection of human-subject data.

Informed Consent Statement: Not applicable. This study did not involve human participants.

Data Availability Statement: No new empirical data were created or analyzed in this study. Data sharing is therefore not applicable.

Conflicts of Interest: R.S. is employed by Medidata Solutions, Inc. The authors declare no other conflicts of interest. The views, opinions, and conclusions expressed in this paper are those of the authors and do not necessarily reflect the official position, policy, or views of Medidata Solutions, Inc. The company had no role in the conceptualization of the study, literature synthesis, development of the AIGATE Framework, preparation of the manuscript, or decision to submit the work for publication.

References

1. Ali, M.A.; Dornaika, F.; Charafeddine, J. Agentic AI: A Comprehensive Survey of Architectures, Applications, and Future Directions. *Artif. Intell. Rev.* **2025**, *59*, 11. [CrossRef]
2. Wang, L.; Ma, C.; Feng, X.; Zhang, Z.; Yang, H.; Zhang, J.; Chen, Z.; Tang, J. A Survey on Large Language Model Based Autonomous Agents. *Front. Comput. Sci.* **2024**, *18*, 186345. [CrossRef]
3. Raza, S.; Sapkota, R.; Karkee, M.; Emmanouilidis, C. TRiSM for Agentic AI: A Review of Trust, Risk, and Security Management in LLM-Based Agentic Multi-Agent Systems. *AI Open* **2026**, *7*, 71–95. [CrossRef]
4. Lazer, S.J.; Aryal, K.; Gupta, M.; Bertino, E. A Survey of Agentic AI and Cybersecurity: Challenges, Opportunities and Use-Case Prototypes. Available online: https://arxiv.org/abs/2601.05293 (accessed on 12 June 2026).
5. Kurtz, A.; Krawiecka, K. Who Governs the Machine? A Machine Identity Governance Taxonomy (MIGT) for AI Systems Operating Across Enterprise and Geopolitical Boundaries. Available online: https://arxiv.org/abs/2604.06148 (accessed on 9 June 2026).
6. He, F.; Zhu, T.; Ye, D.; Liu, B.; Zhou, W.; Yu, P. The Emerged Security and Privacy of LLM Agent: A Survey with Case Studies. *ACM Comput. Surv.* **2025**, *58*, 1–36. [CrossRef]
7. Syros, G.; Suri, A.; Oprea, A.; Ginesin, J.; Nita-Rotaru, C. SAGA: A Security Architecture for Governing AI Agentic Systems. In Proceedings of the Network and Distributed System Security (NDSS) Symposium 2026, San Diego, CA, USA, 23–27 February 2026.
8. Huang, K.; Narajala, V.S.; Yeoh, J.; Ross, J.; Lambe, M.; Raskar, R.; Harkati, Y.; Huang, J.; Habler, I.; Hughes, C. A Novel Zero-Trust Identity Framework for Agentic AI: Decentralized Authentication and Fine-Grained Access Control. In Proceedings of the 2026 International Conference on AI x Data and Knowledge Engineering (AIxDKE) 2026, Laguna Hills, CA, USA, 2–4 February 2026. [CrossRef]
9. Alpay, F.; Alpay, T. AgentSecBench: Measuring Prompt Injection, Privacy Leakage, and Tool-Use Integrity in LLM Agents. Available online: https://arxiv.org/abs/2605.26269 (accessed on 1 July 2026).
10. Debenedetti, E.; Zhang, J.; Balunovic, M.; Beurer-Kellner, L. AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents. In Proceedings of the 38th Conference on Neural Information Processing Systems (NeurIPS 2024), Vancouver, BC, Canada, 10–15 October 2024.
11. Zhang, H.; Huang, J.; Mei, K.; Yao, Y.; Wang, Z.; Zhan, C.; Wang, H.; Zhang, Y. Agent Security Bench (ASB): Formalizing and Benchmarking Attacks and Defenses in LLM-Based Agents. Available online: https://arxiv.org/abs/2410.02644 (accessed on 1 July 2026).
12. Kim, J.; Choi, W.; Lee, B. Prompt Flow Integrity to Prevent Privilege Escalation in LLM Agents. Available online: https://arxiv.org/abs/2503.15547 (accessed on 12 June 2026).
13. Shi, T.; He, J.; Wang, Z.; Wu, L.; Li, H.; Guo, W.; Song, D. Progent: Programmable Privilege Control for LLM Agents. Available online: https://arxiv.org/abs/2504.11703 (accessed on 12 June 2026).
14. Bhagwatkar, R.; Kasa, K.; Puri, A.; Huang, G.; Rish, I.; Taylor, G.W.; Lacoste, A. Indirect Prompt Injections: Are Firewalls All You Need, or Stronger Benchmarks? Available online: https://arxiv.org/abs/2510.05244 (accessed on 1 July 2026).
15. Kim, J.; Guo, W.; Song, D. SoK: Attack and Defense Landscape of Agentic AI Systems. In Proceedings of the 35th USENIX Security Symposium; USENIX, Baltimore, MD, USA, 12–14 August 2026.
16. Schmitz, C.; Rystrøm, J.; Batzner, J. Oversight Structures for Agentic AI in Public-Sector Organizations. In *Proceedings of the 1st Workshop for Research on Agent Language Models (REALM 2025)*; Association for Computational Linguistics: Vienna, Austria, 2025; pp. 298–308.
17. Chhabra, A.; Datta, S.; Nahin, S.K.; Mohapatra, P. Agentic AI Security: Threats, Defenses, Evaluation, and Open Challenges. *IEEE Access* **2026**, *14*, 49455–49482. [CrossRef]
18. Snyder, H. Literature Review as a Research Methodology: An Overview and Guidelines. *J. Bus. Res.* **2019**, *104*, 333–339. [CrossRef]
19. Webster, J.; Watson, R.T. Analyzing the Past to Prepare for the Future: Writing a Literature Review. *MIS Q.* **2002**, *26*, xiii–xxiii. [CrossRef]
20. Jaakkola, E. Designing Conceptual Articles: Four Approaches. *AMS Rev.* **2020**, *10*, 18–26. [CrossRef]
21. Derouiche, H.; Brahmi, Z.; Mazeni, H. Agentic AI Frameworks: Architectures, Protocols, and Design Challenges. Available online: https://arxiv.org/abs/2508.10146 (accessed on 3 June 2026).
22. Zhao, W.; Li, Z.; Zhang, P.; Sun, J. ClawGuard: A Runtime Security Framework for Tool-Augmented LLM Agents Against Indirect Prompt Injection. Available online: https://arxiv.org/abs/2604.11790 (accessed on 1 July 2026).
23. Wang, H.; Poskitt, C.M.; Sun, J. AgentSpec: Customizable Runtime Enforcement for Safe and Reliable LLM Agents. In Proceedings of the International Conference on Software Engineering (ICSE 2026), Rio de Janeiro, Brazil, 12–18 April 2026.
24. Rose, S.; Borchert, O.; Mitchell, S.; Connelly, S. *NIST SP 800-207-Zero Trust Architecture*; NIST Special Publication: Gaithersburg, MD, USA, 2020; Volume 1. [CrossRef]
25. Tabassi, E. *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*; National Institute of Standards and Technology (NIST): Gaithersburg, MD, USA, 2023.

26. OWASP GenAI Security Project. *OWASP Top 10 for Agentic Applications for 2026*; OWASP Foundation: Wilmington, DE, USA, 2025.
27. MITRE. *MITRE ATLAS (Adversarial Threat Landscape for Artificial-Intelligence Systems)*; MITRE: Bedford, MA, USA, 2025.
28. *ISO/IEC 27001:2022*; Information Security, Cybersecurity and Privacy Protection—Information Security Management Systems—Requirements. International Organization for Standardization: Geneva, Switzerland, 2022.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.