Volume 11, Issue 8, August– 2026 International Journal of Innovative Science and Research Technology
ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug1168

Security Architecture for Agentic AI in Enterprise Cloud Environments: A Zero-Trust Framework for Secure Autonomous Systems

Sachin Suryawanshi[1]

[1]Even & Odd Minds LLC Cloud Architecture, Cybersecurity and Artificial Intelligence United States

Publication Date: 2026/09/02

Abstract: Agentic artificial intelligence expands the enterprise security boundary because autonomous agents can plan tasks, retain memory, invoke tools, call APIs, and initiate business actions. Authentication at session start is therefore insufficient when later actions may be influenced by untrusted content, poisoned memory, compromised tools, or excessive delegated privilege. This paper proposes the Zero Trust Agentic AI Security Framework (ZT-AASF), a vendor-neutral architecture that applies continuous verification to consequential agent actions. The framework separates six control planes: identity and delegation, context and data trust, policy and risk decision, tool and action enforcement, runtime observability, and containment and recovery. A contextual authorization model evaluates delegated scope, source provenance, data sensitivity, tool risk, behavioral deviation, and action impact before execution. A design-level evaluation against ten OWASP agentic risk classes produces 27 of 30 control-coverage points for ZT-AASF versus 5 of 30 for a conventional integration baseline. These values represent architectural coverage, not measured attack-prevention rates. The results indicate that moving enforcement from the session boundary to the action boundary can reduce implicit trust, constrain privilege propagation, and improve auditability while preserving useful autonomy.

Keywords: *Agentic AI; Cloud Security; Enterprise Architecture; Identity and Access Management; Runtime Governance; Secure Autonomous Systems; Tool Security; Zero Trust.*

How to Cite: Sachin Suryawanshi (2026) Security Architecture for Agentic AI in Enterprise Cloud Environments: A Zero-Trust Framework for Secure Autonomous Systems. *International Journal of Innovative Science and Research Technology*, 11(8), 2451-2458. https://doi.org/10.38124/ijisrt/26aug1168

I. INTRODUCTION

Large language models are increasingly embedded in systems that do more than generate text. Agentic systems combine reasoning with planning, memory, external tools, software interfaces, and iterative execution. ReAct demonstrated how reasoning and action can be interleaved, while Toolformer showed how models can learn to invoke external tools [1], [2]. In enterprise environments, these capabilities enable agents to query databases, update records, call cloud APIs, and coordinate workflows, but they also create new trust boundaries around every external source and action.

The security problem is broader than conventional prompt safety. Untrusted documents can alter reasoning, memory can preserve malicious context, a legitimate tool can be misused, and a valid service identity can exercise more authority than the current task requires. Agent-focused benchmarks and surveys identify indirect prompt injection, memory poisoning, tool abuse, privilege misuse, and multi-agent failure as recurring risks [4]-[7]. These risks are amplified in cloud environments where agents may operate near business data, identity systems, source repositories, and infrastructure APIs.

NIST zero-trust architecture rejects implicit trust based on location or prior authentication and emphasizes explicit policy decisions before access to protected resources [8], [9]. Agentic AI requires an additional refinement: the unit of authorization should be the proposed action, not merely the authenticated user, agent process, or tool session. This paper therefore introduces the Zero Trust Agentic AI Security Framework (ZT-AASF), which separates model reasoning from enterprise authorization and evaluates each consequential action against identity, delegated scope, context, data sensitivity, tool state, and operational impact.

ZT-AASF contributes four elements: six security control planes, a per-action verification flow, a contextual risk model that routes actions to permit, human approval, denial, or quarantine, and a repeatable design-coverage evaluation mapped to current agentic security risks. The objective is not to claim that architecture eliminates model attacks, but to reduce the consequences of manipulated or incorrect reasoning before it reaches enterprise resources.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug1168

II. RELATED WORK AND RESEARCH GAP

➢ *Agentic AI and Tool Use*

Agentic AI extends language models with planning, memory, tool use, and action. ReAct and Toolformer established influential patterns for iterative reasoning and external tool invocation [1], [2], while broader surveys describe these capabilities as core components of autonomous-agent architectures [3]. Each component increases utility but also adds a trust boundary between the model and systems that can influence or execute its plan.

➢ *Emerging Security Risks*

Recent security work shows that agent risks are system-level rather than purely model-level. InjecAgent demonstrates indirect prompt injection against tool-integrated agents, and Agent Security Bench evaluates attacks across prompts, tools, planning, and memory [4], [5]. Surveys by Chhabra et al. and Su et al. further identify autonomy, persistence, recursive planning, and inter-agent interaction as distinct sources of risk [6], [7]. OWASP summarizes these concerns into ten agentic risk classes, including goal hijacking, tool misuse, privilege abuse, memory poisoning, insecure inter-agent communication, cascading failure, and rogue-agent behavior [10].

➢ *Zero Trust and Cloud-Native Access Control*

Zero-trust architecture provides a mature basis for limiting implicit trust. NIST SP 800-207 separates policy decision and enforcement functions, and SP 800-207A applies identity-aware access principles to cloud-native and multi-cloud applications [8], [9]. Agentic systems, however, can reinterpret a task repeatedly within one authenticated session, so authorization context must follow each action rather than remain fixed at login.

Complementary defenses such as Spotlighting, Task Shield, and contextual agent policy aim to distinguish trusted instructions from untrusted content and preserve task alignment [11]-[13]. These techniques improve model behavior, but they do not independently guarantee that a manipulated agent lacks permission to execute a harmful action. The research gap is therefore architectural: enterprises need a control layer that combines user intent, provenance, privilege, tool semantics, and action impact before execution.

III. MATERIALS AND METHODS

➢ *Research Design*

This study uses a design-science approach in which the research artifact is a security architecture rather than a new machine-learning model. The design process identifies agent-specific trust boundaries, maps them to zero-trust principles, defines enforcement components, and evaluates the resulting framework against recognized agentic risk categories. Three research objectives guide the design: determine where trust decisions should occur in an autonomous workflow; identify the minimum context needed to authorize a consequential action; and define controls that limit blast radius when model reasoning, memory, credentials, or tools are compromised. The framework is intentionally vendor-neutral so that its claims depend on security properties rather than a particular cloud or model platform.

➢ *Source and Standards Selection*

The framework is grounded in peer-reviewed research on autonomous agents, prompt injection, agent security, and contextual policy [1]-[7], [11]-[13], together with NIST zero-trust and AI risk-management guidance [8], [9], [14]-[16] and the OWASP agentic risk taxonomy [10]. These sources were selected because they cover both model behavior and enterprise security controls. The literature was examined for recurring control requirements involving identity, delegation, provenance, tool trust, data handling, monitoring, and recovery. Requirements that appeared across multiple sources were converted into architectural capabilities, then organized into control planes to reduce overlap between model safeguards and enterprise enforcement responsibilities.

➢ *Threat Model and Assumptions*

The threat model assumes that users, models, retrieved content, memory stores, tool providers, and downstream cloud resources may have different trust levels. Attackers may control content consumed by an agent, steal credentials, poison memory, alter tool metadata, or influence another agent. The enterprise identity and policy services are treated as trusted control components, while model output and proposed actions are treated as untrusted until verified. Consequential actions are assumed to be expressible in a normalized form containing actor, delegated principal, task identifier, operation, target resource, data class, and relevant context. The model excludes compromise of the enterprise root of trust itself, such as a fully compromised identity provider or policy engine, because such failures require separate platform-security controls. It does include failures of the agent runtime, model output, retrieved data, memory, and external tools because these are realistic boundaries for enterprise deployments.

➢ *Evaluation Method*

Evaluation uses a design-control coverage rubric rather than an attack-success metric. Each of the ten OWASP agentic risk categories receives a score from 0 to 3 for a conventional baseline and for ZT-AASF: 0 indicates no explicit control, 1 a primarily reactive control, 2 a preventive control with limited context, and 3 a preventive control with contextual verification and containment. The baseline represents a common integration in which a user authenticates, the agent receives a broadly permitted service identity, tools are configured statically, and normal application logging is enabled. Scoring is performed by mapping documented framework controls to each risk class and checking whether prevention, contextual decision making, and containment are explicitly represented. The comparison is intentionally architectural and does not claim empirical attack-prevention percentages. Scenario analysis is then used as a second validation step to verify that the individual control planes cooperate coherently when an attack crosses more than one trust boundary.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug1168

IV. THREAT MODEL FOR ENTERPRISE AGENTIC AI

➢ *Assets and Trust Boundaries*

Critical assets include business data, credentials, approved user intent, persistent memory, tool definitions, cloud resources, audit evidence, and transaction authority. Trust boundaries exist between the user and agent, agent and model, model and retrieved context, agent and memory, agent and tools, tools and enterprise resources, and one agent and another. The framework focuses on boundaries where model-generated intent becomes externally observable action.

➢ *Adversary Capabilities*

An adversary may embed instructions in documents, email, websites, tickets, or retrieval corpora; steal or replay agent credentials; poison persistent memory; modify a tool or its description; or exploit an overly trusted peer agent. The model itself may also fail without an adversary through hallucination, goal drift, incorrect planning, or repeated retries. Security controls must therefore address both malicious influence and ordinary model error.

➢ *Enterprise Failure Modes*

Five failure modes are especially important: authority inheritance, where broad privileges persist across steps; context contamination, where untrusted content becomes instruction; tool substitution, where a changed tool is treated as equivalent to an approved one; delegation expansion, where downstream agents gain more authority than the parent task permits; and cascading execution, where repeated autonomous actions amplify an initial error. These failure modes are chosen because each can convert a local model mistake into an enterprise consequence. They also cut across traditional security domains: identity controls address authority, data controls address context, supply-chain controls address tools, and runtime limits address cascading behavior. A useful architecture must connect these domains at the moment an agent attempts action rather than relying on any one defensive layer. These modes motivate the action-level enforcement boundary shown in Fig. 1.

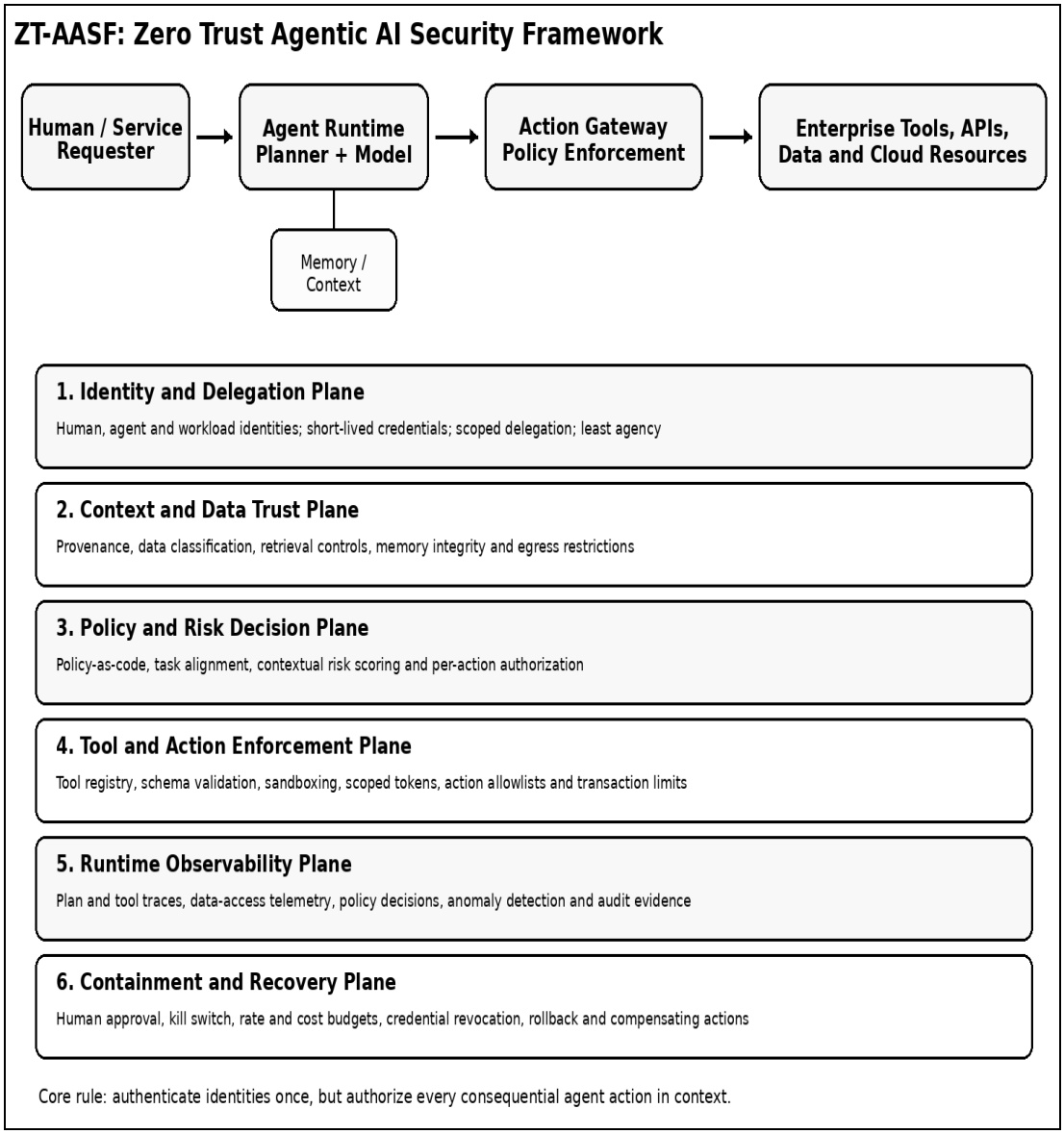

Fig 1 ZT-AASF Architecture and Six Security Control Planes.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug1168

V. PROPOSED ZERO TRUST AGENTIC AI SECURITY FRAMEWORK

➢ *Design Principles*

ZT-AASF follows five principles: authenticate identities but authorize actions; minimize standing privilege; treat context and memory as data with provenance rather than trusted instruction; mediate consequential tool calls through an independent enforcement point; and contain failures with revocation, limits, approval, and recovery controls. The model may propose an action, but it does not control the policy that authorizes that action.

➢ *Six Security Control Planes*

The identity and delegation plane separates human, agent, workload, tool, and service identities. A user grants a bounded mandate rather than a reusable impersonation token. Credentials are short-lived and scoped to the approved resource, operation, and task.

The context and data trust plane records provenance and sensitivity for user instructions, retrieved content, memory entries, and tool responses. Low-trust context can inform reasoning without automatically becoming policy, and sensitive data can be filtered or redacted before it reaches external services.

The policy and risk decision plane evaluates normalized actions against explicit enterprise rules. It checks delegated authority, task alignment, resource sensitivity, tool trust, behavioral deviation, and expected impact. Hard policy remains deterministic, while contextual risk can trigger step-up approval or quarantine.

The tool and action enforcement plane is the primary policy enforcement point. Agents do not connect directly to arbitrary resources using broad credentials. An action gateway validates tool identity, parameters, destination, schema, and network policy, then brokers a scoped credential only after authorization.

The runtime observability plane records the evidence needed to reconstruct an autonomous workflow, including task, requester, agent, model or policy version, tool, target, risk score, decision, credential issuance, and outcome. Structured events support anomaly detection and incident investigation without requiring full prompt logging.

The containment and recovery plane limits blast radius when prevention fails through human approval for irreversible operations, rate and transaction limits, circuit breakers, credential revocation, rollback where feasible, and isolation of suspicious agents or tools.

➢ *Per-Action Continuous Verification*

The key architectural change is the location of the security decision. Authorization is performed immediately before a consequential action, not only when the agent receives a tool connection. The agent submits a normalized action request to the policy layer; identity, delegation, provenance, data sensitivity, tool state, and behavioral context are evaluated; and the enforcement point either blocks the action, requests approval, or issues narrowly scoped execution authority. Fig. 2 summarizes this flow.

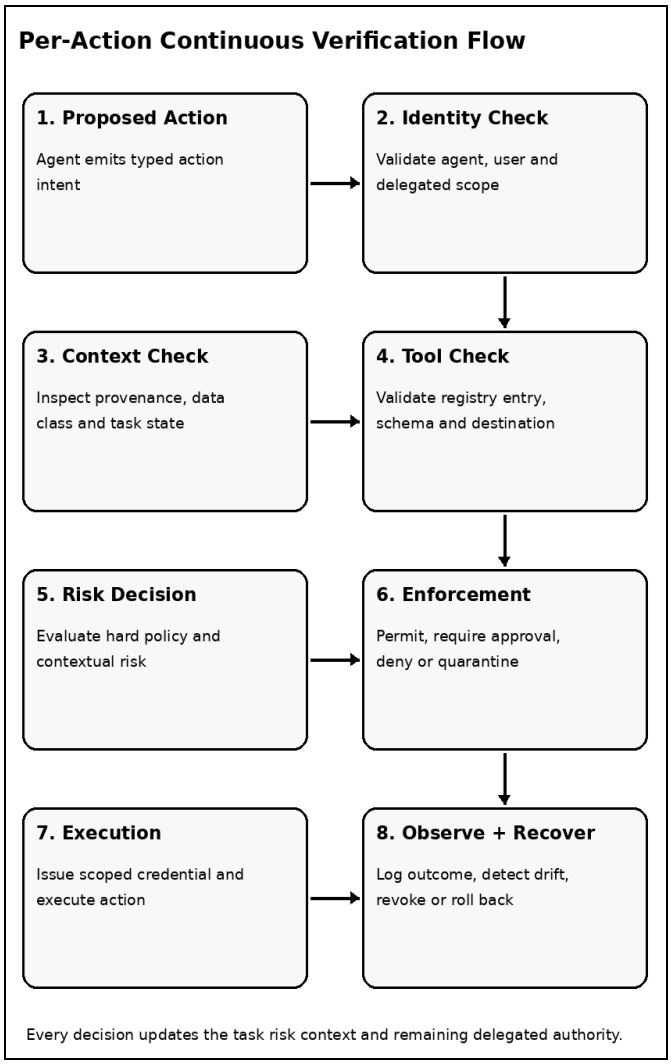

Fig 2 Per-Action Continuous Verification Flow.

➢ *Dynamic Risk and Authorization Model*

Deterministic policy should govern non-negotiable constraints, while a contextual risk score can route otherwise permissible actions. Let normalized risk at time t be:

$$R_t = w1S_t + w2P_t + w3U_t + w4B_t + w5D_t + w6I_t \quad (1)$$

In (1), S(t) is source-trust deficit, P(t) privilege exposure, U(t) tool or action sensitivity, B(t) behavioral deviation from the approved task, D(t) data sensitivity, and I(t) impact or irreversibility. Weights w1...w6 are organization-specific and sum to 1. The score is not a machine-learning prediction; it is a transparent policy abstraction that combines evidence already available to the control plane.

For actions that pass hard policy, two thresholds are sufficient for a practical decision model. Low-risk actions may execute automatically, intermediate-risk actions require independent approval or additional verification, and high-risk actions are denied or quarantined. Thresholds should be

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug1168

calibrated using incident data, business criticality, and acceptable operational friction.

➢ *Identity, Delegation and Least Agency*

Identity should distinguish the requester from the autonomous workload. A delegated mandate can carry the task ID, permitted resource classes, operation types, expiry, and maximum impact. The agent exchanges this mandate for short-lived, audience-bound credentials only after the current action is authorized. This prevents a valid human session from becoming an unrestricted agent credential.

Least agency complements least privilege. An invoice-summary agent may require read access to invoice data but should not possess a payment tool. An infrastructure-planning agent may read configuration and produce a proposed change without holding production write authority. Reducing the available action set decreases both accidental and adversarial blast radius.

➢ *Memory, Context and Data Protection*

Context should retain distinctions between policy, user intent, retrieved evidence, tool output, and remembered state. Provenance, integrity metadata, and trust labels should travel with context so persistent memory does not become trusted merely because it survived from a previous session. High-impact decisions that depend on low-trust or anomalous context can be revalidated or escalated.

Data protection must also control outbound disclosure. The action gateway can apply classification, destination policy, redaction, and data-loss-prevention checks before information is sent to a model, tool, or external service. This keeps data-governance policy outside model control and allows the same rule set to apply across agent implementations.

➢ *Tool, MCP and Action Security*

Tools convert model intent into side effects and therefore require independent trust. Each approved tool should have a registry identity, owner, version, allowed operations, input schema, expected destinations, and risk classification. The enforcement point rejects unknown or changed capabilities, validates parameters, and prevents the model from silently expanding the tool contract.

Model Context Protocol and similar interoperability mechanisms make tool discovery easier but can also enlarge the supply-chain boundary [17]. Enterprise deployments should authenticate tool servers, pin or attest approved versions, constrain outbound destinations, and separate tool description from authorization. Discovery may tell an agent what a tool can do; enterprise policy decides whether the agent may do it.

➢ *Multi-Agent Trust*

Multi-agent systems should not propagate authority transitively. When Agent A delegates to Agent B, the receiving agent authenticates independently and receives a new bounded mandate derived from the parent task. Depth, duration, operation class, and transaction limits constrain cascading behavior, while correlated high-risk actions can trigger a circuit breaker.

VI. ENTERPRISE CLOUD IMPLEMENTATION

➢ *Vendor-Neutral Cloud Mapping*

ZT-AASF can be mapped to standard cloud components: an identity provider for human and workload identity, an API or action gateway for enforcement, a policy engine for deterministic and contextual decisions, a secrets or token broker for short-lived credentials, data-classification services for outbound controls, and centralized telemetry for audit and response. The design is vendor-neutral and can operate across single-cloud or multi-cloud estates.

Not every inference requires the full control path. Enforcement should focus on consequential boundaries such as protected-data access, external transmission, code execution, state-changing APIs, privileged cloud operations, financial actions, and cross-agent delegation. Low-risk internal reasoning can remain within the agent runtime.

➢ *Deployment Patterns*

A centralized action gateway simplifies policy, credential brokerage, tool governance, and audit for organizations with many agents. A distributed pattern may be used where latency or data residency requires local enforcement, provided policy semantics and telemetry remain consistent. Adoption can be incremental: inventory agents and privileges, route tool calls through enforcement, replace standing credentials with scoped tokens, add context-aware risk decisions, and finally introduce automated containment.

➢ *Operational Telemetry and Response*

Telemetry should record structured security facts rather than relying only on natural-language traces. Each event should capture task ID, actor, delegated identity, action, target, data classification, tool version, policy decision, risk score, credential issuance, and outcome. Sensitive prompt or document content should be logged only when necessary and protected by normal data-governance controls.

Response should match action impact. Suspicious reads may trigger narrower retrieval or step-up approval; suspicious writes should be denied or quarantined; credential compromise should revoke task tokens; and repeated correlated failures should suspend the agent or tool. The objective is to stop propagation even when the initial model error is not detected.

VII. RESULTS

➢ *Design-Level Coverage Matrix*

Table I summarizes the design-level coverage produced by the rubric in Section III. The conventional baseline scores 5 of 30 because it relies mainly on authentication, static tool configuration, and reactive logging. ZT-AASF scores 27 of 30 by adding action-level authorization, provenance-aware context controls, scoped credentials, tool verification, delegation boundaries, and containment. The largest gains

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug1168

occur in tool misuse, identity and privilege abuse, memory poisoning, insecure inter-agent communication, cascading failure, and rogue-agent behavior. The comparison measures explicit architectural controls, not empirical attack resistance.

Table 1 Design-Control Coverage Against Agentic AI Risk Classes

Risk Class	Baseline	ZT-AASF	Primary ZT-AASF Controls
ASI01 Agent Goal Hijack	1	3	Context provenance; task alignment; action authorization; egress control
ASI02 Tool Misuse and Exploitation	1	3	Tool registry; parameter schema; scoped tokens; sandboxing
ASI03 Identity and Privilege Abuse	1	3	Separate workload identity; delegated scope; short-lived credentials
ASI04 Agentic Supply Chain Vulnerabilities	0	2	Approved tool versions; integrity metadata; lifecycle re-evaluation
ASI05 Unexpected Code Execution	1	3	Restricted execution tools; sandbox; network and file policy
ASI06 Memory and Context Poisoning	0	3	Provenance; memory integrity; retention limits; quarantine
ASI07 Insecure Inter-Agent Communication	0	3	Independent authentication; non-transitive delegation; message provenance
ASI08 Cascading Failures	1	3	Budgets; rate limits; circuit breakers; kill switch; rollback
ASI09 Human-Agent Trust Exploitation	0	2	Independent approval UI; structured action summary; risk-based step-up
ASI10 Rogue Agents	0	2	Least agency; continuous observation; revocation; containment
Total	**5 / 30**	**27 / 30**	**16.7% baseline versus 90.0% design coverage**

➢ *Scenario-Based Validation*

Scenario analysis further illustrates how the controls interact. For indirect prompt injection, a malicious supplier document may influence reasoning, but the resulting action is still checked against the approved task, source trust, destination, and data policy before execution. For credential theft, short-lived audience-bound tokens reduce replay value compared with a broad service-account credential. For poisoned memory, provenance and integrity metadata prevent persistence alone from converting an untrusted statement into policy. For a compromised tool server, registry identity, version, schema, parameter, and destination checks move trust from model-visible descriptions to enterprise-controlled metadata. In multi-agent delegation, each receiving agent obtains a new bounded mandate rather than inheriting authority transitively, and transaction limits or circuit breakers constrain cascading failure.

➢ *Interpretation*

Three observations follow. First, the strongest control is the separation of model reasoning from enterprise authorization rather than any single prompt defense. Second, independent enforcement limits consequences even when model-level defenses fail. Third, contextual authorization can preserve useful autonomy by allowing low-risk actions automatically while escalating irreversible or anomalous actions.

VIII. DISCUSSION

➢ *Contribution Versus Existing Approaches*

ZT-AASF complements model-level defenses such as input separation, spotlighting, and task alignment [11], [12]. Those methods aim to improve the action selected by the model; ZT-AASF assumes the model can still be wrong and requires independent enterprise controls before protected state changes or sensitive disclosures occur. This follows a familiar security-engineering principle: an untrusted component may participate in decision making without receiving unilateral authority over protected resources.

The framework also extends traditional zero trust by adding purpose, provenance, tool semantics, and action impact to the authorization context [8], [9]. In agentic systems, a valid workload can reinterpret its task repeatedly within one session. The practical shift is from identity-aware access toward intent-aware, action-level access while retaining deterministic policy for non-negotiable controls.

The design aligns with NIST AI risk-management guidance by converting governance goals into runtime controls for delegated authority, traceability, risk decisions, and containment [14]-[16]. Secure development remains essential because vulnerabilities in the agent runtime, tool code, or enforcement layer can bypass these protections.

➢ *Security-Utility Tradeoffs*

Per-action authorization adds latency and policy complexity. The framework therefore distinguishes consequential operations from low-risk reasoning. Local policy caches, pre-authorized read classes, and scoped capability tokens can reduce overhead, while human approval is reserved for actions whose impact justifies interruption. Approval interfaces should show the exact operation, destination, data involved, and reason for escalation.

Detailed observability also creates privacy risk. Logs should capture structured security metadata by default, restrict sensitive content, enforce retention limits, and protect high-

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug1168

value evidence from tampering. Agent telemetry should be governed like any other enterprise data set.

➢ *Limitations*

The evaluation is architectural and scenario-based, so the 27 of 30 score is not an empirical attack-prevention rate. Organizations must calibrate the risk factors and thresholds for their own environment. The framework also assumes that consequential actions can be normalized and mediated; agents with unrestricted operating-system or network access require sandboxing to preserve the enforcement boundary. Human approval can itself fail through overload or social engineering and therefore needs separate usability validation.

Agent platforms and interoperability protocols are evolving quickly, and future browser, multimodal, or computer-use agents may introduce actions that are difficult to express in static schemas. Empirical work should therefore test the framework across multiple agent stacks, model families, cloud tools, and attack datasets.

IX. CONCLUSION AND FUTURE SCOPE

Agentic AI moves generative models from content generation into enterprise execution, making session-level authentication insufficient. This paper proposed ZT-AASF, a zero-trust architecture that treats each consequential agent action as an untrusted proposal until identity, delegation, context, data sensitivity, tool state, task alignment, and operational risk have been evaluated. Six control planes place an independent enforcement boundary between model reasoning and protected enterprise resources.

The design-level evaluation shows broader explicit coverage of current agentic risk categories than a conventional integration baseline, especially for tool misuse, privilege abuse, memory poisoning, insecure inter-agent communication, and cascading failure. The result demonstrates architectural coverage under the stated rubric, not a measured prevention percentage. The central contribution is a practical method for limiting what a manipulated or mistaken agent can access, disclose, invoke, or change.

Future work should implement the framework in a controlled cloud testbed and measure attack success, task completion, policy latency, false escalation, and recovery time. Additional study is needed on risk-score calibration, machine-verifiable user intent, secure cross-agent delegation, tool and MCP attestation, approval usability, and privacy-preserving telemetry.

REFERENCES

[1]. S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing Reasoning and Acting in Language Models," in Proc. International Conference on Learning Representations (ICLR), 2023.

[2]. T. Schick et al., "Toolformer: Language Models Can Teach Themselves to Use Tools," in Advances in Neural Information Processing Systems, vol. 36, 2023, doi: 10.52202/075280-2997.

[3]. L. Wang et al., "A Survey on Large Language Model Based Autonomous Agents," Frontiers of Computer Science, vol. 18, art. 186345, 2024, doi: 10.1007/s11704-024-40231-1.

[4]. Q. Zhan, Z. Liang, Z. Ying, and D. Kang, "InjecAgent: Benchmarking Indirect Prompt Injections in Tool-Integrated Large Language Model Agents," in Findings of the Association for Computational Linguistics: ACL 2024, pp. 10471-10506, 2024, doi: 10.18653/v1/2024.findings-acl.624.

[5]. H. Zhang, J. Huang, K. Mei, Y. Yao, Z. Wang, C. Zhan, H. Wang, and Y. Zhang, "Agent Security Bench (ASB): Formalizing and Benchmarking Attacks and Defenses in LLM-Based Agents," in Proc. International Conference on Learning Representations (ICLR), 2025.

[6]. A. Chhabra, S. Datta, S. K. Nahin, and P. Mohapatra, "Agentic AI Security: Threats, Defenses, Evaluation, and Open Challenges," IEEE Access, vol. 14, pp. 49455-49482, 2026, doi: 10.1109/ACCESS.2026.3675554.

[7]. H. Su, J. Luo, C. Liu, X. Yang, Y. Zhang, Y. Dong, and J. Zhu, "A Survey on Autonomy-Induced Security Risks in Large Model-Based Agents," IEEE Transactions on Pattern Analysis and Machine Intelligence, early access, Apr. 2026, doi: 10.1109/TPAMI.2026.3688650.

[8]. S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero Trust Architecture," NIST Special Publication 800-207, National Institute of Standards and Technology, 2020, doi: 10.6028/NIST.SP.800-207.

[9]. R. Chandramouli and Z. Butcher, "A Zero Trust Architecture Model for Access Control in Cloud-Native Applications in Multi-Cloud Environments," NIST Special Publication 800-207A, National Institute of Standards and Technology, 2023, doi: 10.6028/NIST.SP.800-207A.

[10]. OWASP GenAI Security Project, "OWASP Top 10 for Agentic Applications for 2026," 2025. [Online]. Available: https://genai.owasp.org/resource/owasp-top-10-for-agentic-applications-for-2026/. Accessed: Aug. 23, 2026.

[11]. K. Hines, G. Lopez, M. Hall, F. Zarfati, Y. Zunger, and E. Kiciman, "Defending Against Indirect Prompt Injection Attacks With Spotlighting," in Proc. CAMLIS 2024, CEUR Workshop Proceedings, vol. 3920, pp. 48-62, 2024, doi: 10.48550/arXiv.2403.14720.

[12]. F. Jia, T. Wu, X. Qin, and A. Squicciarini, "The Task Shield: Enforcing Task Alignment to Defend Against Indirect Prompt Injection in LLM Agents," in Proc. 63rd Annual Meeting of the Association for Computational Linguistics (ACL), pp. 29680-29697, 2025.

[13]. L. Tsai and E. Bagdasarian, "Contextual Agent Security: A Policy for Every Purpose," in Proc. 19th Workshop on Hot Topics in Operating Systems (HotOS), pp. 8-17, 2025, doi: 10.1145/3713082.3730378.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug1168

[14]. E. Tabassi, "Artificial Intelligence Risk Management Framework (AI RMF 1.0)," NIST AI 100-1, National Institute of Standards and Technology, 2023, doi: 10.6028/NIST.AI.100-1.
[15]. C. Autio et al., "Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile," NIST AI 600-1, National Institute of Standards and Technology, 2024, doi: 10.6028/NIST.AI.600-1.
[16]. H. Booth et al., "Secure Software Development Practices for Generative AI and Dual-Use Foundation Models: An SSDF Community Profile," NIST Special Publication 800-218A, National Institute of Standards and Technology, 2024.
[17]. X. Hou, Y. Zhao, S. Wang, and H. Wang, "Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions," ACM Transactions on Software Engineering and Methodology, online publication, 2026, doi: 10.1145/3796519.