ISSN :2584-2706 DOI: https://doi.org/10.5281/zenodo.23767601

Zero-Trust AI: A Security Architecture for Protecting Autonomous AI Agents Against Adversarial and Manipulative Inputs

Naseef Talukder
Faculty of ITHM Department, Primeasia University, Dhaka

Abstract: Autonomous AI agents combine language-model reasoning with external data, persistent context, and privileged tools, creating attack paths in which adversarial content can redirect actions and misuse delegated authority. This paper proposes a zero-trust security architecture that treats inputs, agent identities, and tool requests as untrusted until independently verified, and enforces least-privilege, context-bound authorization at execution time. The architecture separates retrieved content from governing instructions and places deterministic policy checks, sandboxed execution, and auditable approval boundaries between agent plans and consequential actions. By making authority explicit and validating each action against policy and current context, the design aims to limit the impact of prompt injection and other manipulative inputs without relying solely on model-level safeguards.

Keywords: *Zero-Trust Architecture; Autonomous AI Agents; Indirect Prompt Injection; Least-Privilege Authorization; Runtime Governance; Adversarial Inputs.*

I. INTRODUCTION

Autonomous AI agents extend large language models beyond text generation by enabling them to plan, access tools, and act on external systems, including environments containing untrusted content such as emails and websites. This shift makes security depend not only on what an agent says, but also on the authority it exercises and the effects of its tool calls.

Indirect prompt injection illustrates this risk: instructions embedded in retrieved content can manipulate an agent's decisions and induce unauthorized actions, while adaptive attacks have shown that existing defenses can be bypassed. Because prompt-level guardrails and semantic filters remain insufficient as sole enforcement mechanisms, this document develops a zero-trust architecture that treats inputs as untrusted and places identity, least-privilege authorization, and deterministic checks at the point of execution. Its scope is to connect these controls into a security model for governing agent actions and limiting the consequences of adversarial or manipulative inputs.

From Probabilistic Assistants to Tool-Using Systems with Real-World Authority

Autonomous AI agents differ from conversational assistants because they transform natural-language instructions into plans, tool calls, state changes, and delegated requests. Their security boundary therefore includes the model, controller, memory, retrieved content, external tools, credentials, and the environment in which actions execute. The agent may read email, modify files, query databases, invoke cloud services, or issue transactions while processing content controlled by other parties [1], [2]. A prompt injection that would previously produce an inaccurate answer can consequently become data disclosure, code execution, privilege misuse, or an irreversible transaction.

The central architectural problem is the **determinism gap**: the agent may be technically capable of an action without being authorized to perform that particular action in its present context [3]. Model alignment and prompt-level filtering influence proposed behavior, but they do not establish that an executed operation satisfies identity, scope, approval, target, or provenance constraints. This paper therefore treats the agent as an untrusted runtime actor whose proposed actions require independent authorization before state transition.

Research Scope, Evidence Corpus, and Zero-Trust Architectural Thesis

The analysis covers indirect prompt injection, tool misuse, delegated authority, context contamination, memory persistence, retrieval authorization, multi-agent communication, execution isolation, and lifecycle controls. The evidence corpus combines benchmark studies, architectural specifications, formal authorization proposals, deployment reports, red-team systems, and threat surveys. Experimental claims are kept distinct from conceptual designs and simulation results, since reported attack rates and latency measurements arise from different environments and endpoints.

ISSN :2584-2706 DOI: https://doi.org/10.5281/zenodo.23767601

The thesis is **zero-trust execution**: authenticate every workload and delegation, classify every input by trust, minimize each capability, authorize each canonical action at execution time, isolate the action environment, and preserve evidence sufficient for replay and review. This position does not reject probabilistic defenses. It assigns them to inspection, prioritization, and anomaly detection while reserving the final allow or deny decision for controls outside the model's influence [4], [5].

Planned Evidence Map of Agent Capabilities, Attack Surfaces, and Controls

The evidence map follows the agent's operational path: skills and configuration, input acquisition, memory and retrieval, planning, tool invocation, delegation, execution, and recovery. Each stage is linked to an attacker capability, a protected asset, an enforcement point, and an auditable outcome. This structure follows lifecycle accounts that identify initialization, input processing, memory, decision-making, and execution as connected stages rather than isolated interfaces [6].

The Expanding Threat Surface: Indirect Prompt Injection, Authority Misuse, and Context Manipulation

Indirect Prompt Injection Across Email, Web Pages, Repositories, Tool Results, and RAG Contexts

Indirect prompt injection occurs when instructions embedded in externally retrieved material alter an agent's subsequent reasoning or actions. The attacker need not control the user's initial prompt. Instead, malicious text can enter through email, web pages, repositories, downloaded files, group chats, tool responses, or retrieved documents. InjecAgent evaluated 1,054 cases across 17 user tools and 62 attacker tools, and found that ReAct-prompted GPT-4 agents were vulnerable in 24% of cases; reinforcing the malicious instructions with a hacking prompt nearly doubled that rate [7]. The benchmark separates direct harm from private-data exfiltration, showing why textual correctness is an inadequate security endpoint.[8]

The attack surface expands because agents often treat content and instructions as tokens in one context. A webpage can therefore influence browsing, a repository comment can influence shell execution, and an email can redirect an otherwise legitimate search. LivePI evaluates seven input surfaces, twelve attack or rendering families, and five malicious goals in a test-controlled environment. Across five evaluated model families, total attack success rates ranged from 10.7% to 29.6%; group-chat injections were successful across the evaluated backbones, while repository-link attacks produced high-severity failures despite a small denominator [9]. These results indicate that domain allowlists do not establish content trust: an approved site can still deliver attacker-controlled instructions.[10]

Retrieval introduces a second problem, because unauthorized information can influence generation before any response filter operates. In a corpus of 43 chunks from 28 documents, retrieve-then-filter exposed unauthorized context in 96.8% of structured queries and violated authorization correctness in every query that should have been denied. Authorization-first retrieval produced zero structural leaks and zero answer leaks across 93 queries [11]. The implication is architectural: content must be classified and access-checked before it enters model context, not after the model has already incorporated it.[12]

Permission Escalation at the Agent–Tool and Agent–Resource Boundaries

Authority misuse is distinct from prompt compromise. Prompt injection changes what the agent believes it should do; permission escalation concerns what authority the system permits it to exercise. A legitimate agent can still exceed task scope through broad tool permissions, inherited credentials, ambiguous delegation, or a tool that performs more operations than its interface suggests. The distinction is material because a secure response to injected instructions cannot compensate for an execution layer that grants unrestricted shell, filesystem, database, or cloud access [13].

The agent-tool boundary is especially exposed when protocols convey capability without independently constraining use. MCP servers may have host-level access, while coding agents can read repositories, invoke terminals, inspect user files, and contact external services. In a study of 296 popular MCP servers, automatically generated access-control policies reached 80.9% accuracy, and the enforcement engine blocked most tested threats in malicious servers with negligible overhead [14]. The result supports mediation rather than direct trust between model and tool, while also showing that generated policies require validation.

Least privilege must be task-specific and state-sensitive. Progent represents allowed and unnecessary tool calls as symbolic rules, checks each call deterministically, and permits automatic policy narrowing while requiring explicit approval for privilege expansion [15]. MiniScope similarly reconstructs relationships among tool permissions and reports 1% to 6% latency overhead relative to unrestricted tool calling [16]. These approaches address a limitation of static roles: the same identity may need read access for one task, write access for another, and no access to a sensitive resource during either task's untrusted retrieval phase.

Adaptive and Automated Attacks Against Browsing, Coding, and Multi-Agent Workflows

Static attack strings provide an incomplete robustness test because an attacker can observe outcomes and optimize payloads. An adaptive-attack study bypassed eight defenses and achieved attack success rates above 50% against each evaluated defense [17]. AGENTVIGIL used black-box optimization with Monte Carlo Tree Search and achieved 71% success on AgentDojo against o3-mini and 70% against GPT-4o, approximately doubling handcrafted baselines; it also transferred to unseen tasks and redirected browsing agents to arbitrary URLs [18]. These findings weaken claims based only on performance against fixed prompts. [19]

ISSN :2584-2706 DOI: https://doi.org/10.5281/zenodo.23767601

Automated attacks are model-dependent rather than uniformly portable. In an evaluation of 80 task pairs across four domains, black-box TAP attacks outperformed gradient-based GCG attacks, while task-universal attacks transferred to unseen tasks and out-of-distribution domains. Attacks optimized on smaller open models did not transfer reliably to frontier models such as GPT-5 [20]. This mixed result argues for adaptive evaluation against the deployed model and tool configuration, not for a single universal attack score.[21]

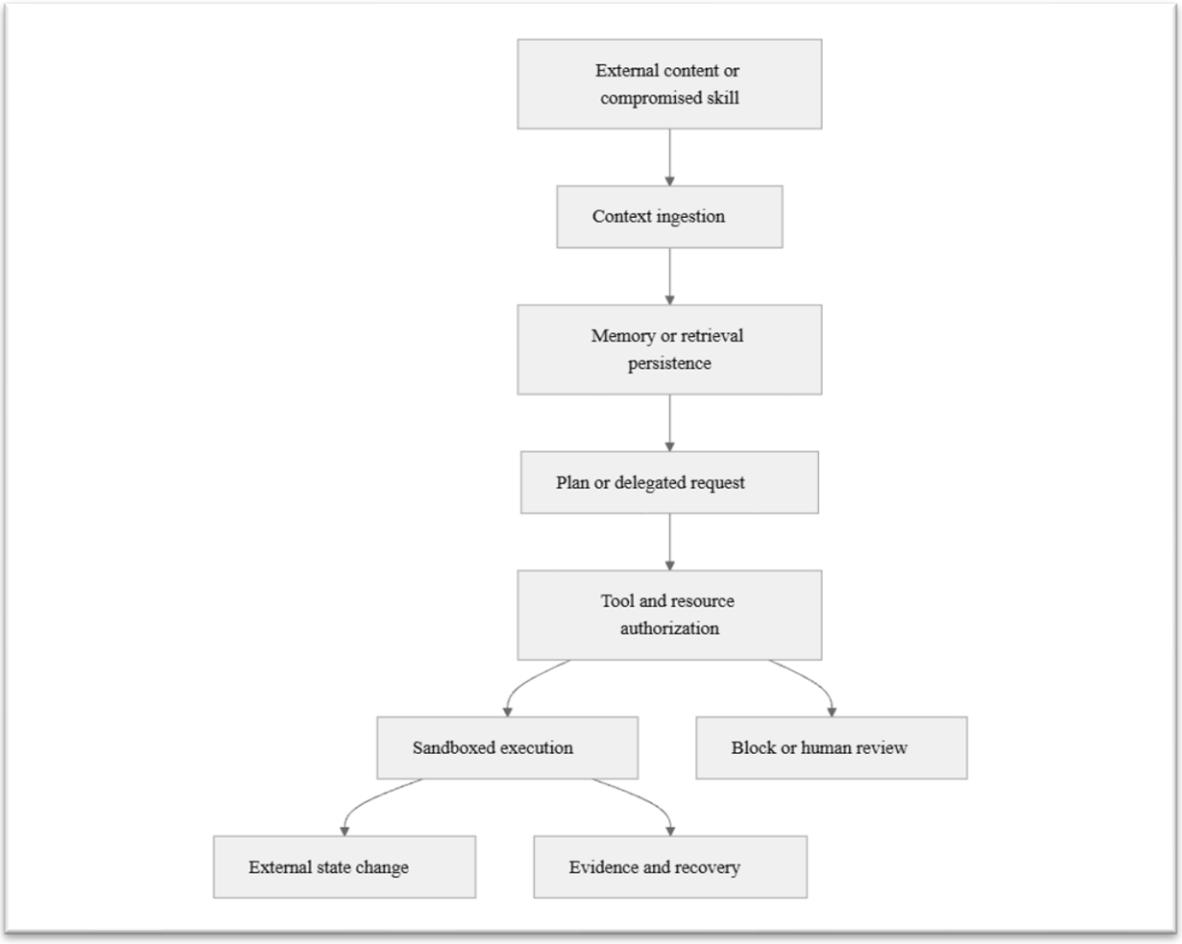
Coding agents expose a particularly direct pathway from content to system effect. AIShellJack evaluated 314 payloads covering 70 MITRE ATT&CK techniques against GitHub Copilot and Cursor, with malicious-command execution success reaching 84% [22]. The pathway may begin in a poisoned repository or issue, pass through a coding plan, and end in shell execution or credential theft. Multi-agent systems add propagation: one compromised agent can transmit altered instructions, tool results, or unsafe practices to peers, making identity and delegation controls necessary even when each local agent appears benign [3], [23].

Threat Distribution Across the Agent Lifecycle: Skills, Memory, Planning, Execution, and Collaboration

Lifecycle analysis prevents security teams from treating injection as a single input-filtering problem. At initialization, an untrusted skill, plugin, tool description, or configuration can establish a persistent capability. During input processing, hostile content enters through external channels. Memory and retrieval can preserve contamination beyond the original encounter, while planning converts manipulated context into a sequence of intended operations. Execution then exposes credentials and external state, and collaboration can transmit the compromise across agents or domains [6].

The resulting threat distribution is temporal as well as spatial. A benign-looking action may create a memory entry that changes later decisions; a delegated request may carry insufficient provenance; and a low-risk read may prepare a high-risk export. AgentCanary therefore evaluates full trajectories in executable environments with persistent state, scoring outcome safety, security awareness, and task utility rather than relying on final text or one tool call [24]. Controls must consequently bind authority to the current task and execution state, expire credentials, inspect cross-stage transitions, and retain evidence of how a decision was reached.[25]

The following pathway separates attacker influence from system effect. It also identifies where a zero-trust control should interrupt propagation.



ISSN :2584-2706 DOI: https://doi.org/10.5281/zenodo.23767601

Comparative Attack-Surface Table and Threat-Pathway Diagram

The comparative evidence shows that attack success is not a single property of the model. It varies with task framing, input surface, adaptive optimization, available authority, and whether the environment executes real actions. A controlled study of 3,274 trials found 89% to 91% attack success for instruction-following tasks versus 15% to 26% for constrained tasks; MCP did not add a measurable attack surface relative to a direct API baseline on the tested model, with p greater than 0.6 [26]. This result cautions against attributing all vulnerability to a framework when task design and authority determine the consequence.

Attack pathway	Entry surface	Immediate agent failure	Potential consequence	Evidence
Retrieval poisoning	Documents and RAG chunks	Untrusted content enters context	Unauthorized answer or disclosure	Authorization-first retrieval eliminated 93 of 93 tested leaks [11]
Tool-result injection	Browsing, email, and APIs	Result is interpreted as instruction	Unapproved follow-on call	InjecAgent reported 24% vulnerability for ReAct GPT-4 [7]
Repository poisoning	Code, issues, and links	Agent adopts malicious coding step	Shell execution or exfiltration	Coding-agent success reached 84% for malicious-command execution [22]
Delegation misuse	Agent-to-agent messages	Authority or provenance is inherited broadly	Lateral movement or policy bypass	Multi-agent trust failures span user, agent, device, and domain boundaries [3]
Adaptive injection	Repeated attack interaction	Defense-specific payload bypasses filter	Persistent or high-impact action	Eight evaluated defenses were bypassed above 50% ASR [17]

The table separates where influence enters from what authority makes influence harmful. That separation motivates the next section's identity, context, and capability controls.

Zero-Trust Foundations for Agentic Systems: Identity, Context Separation, and Least Privilege

Cryptographically Bound Agent Identity with SPIFFE, OIDC, OAuth 2.1, and Delegated Tokens

Workload identity must distinguish the human principal, the agent process, the tool service, and each delegated hop. A user token alone cannot establish which agent executed an action, while a service identity alone cannot establish on whose behalf the action occurred. The reference identity model therefore carries both a user authorization context and a cryptographically verifiable workload identity through each boundary. A proposed implementation combines OIDC and OAuth 2.1 with PKCE for human authentication, SPIFFE JWT-SVIDs for workload identity, and OPA for joint policy evaluation; its reported overhead was below 12 ms per request [27].

SPIFFE/SPIRE addresses short-lived workload credentials and automated rotation. A Kubernetes multi-agent implementation issued one-hour X.509 SVIDs and rotated them at 50% of their lifetime, with no authentication failures during continuous rotation [28]. JWT-SVIDs or mTLS certificates should be audience-scoped, bound to the workload instance, and rejected when expired or presented to an unintended service. OAuth token exchange can narrow a user's authority into a task-specific token, while sender-constrained mechanisms such as DPoP can reduce replay risk [29].

Delegation requires explicit provenance. An agent should receive a token that identifies the originating subject, the delegating workload, the intended recipient, the permitted capability, the resource scope, the expiry, and the workflow step. Agentic JWT proposes chained delegation assertions and proof-of-possession keys to bind actions to user intent and workflow steps, with a proof-of-concept reporting sub-millisecond overhead and blocking scope violations, replay, and impersonation pathways [30]. These mechanisms do not make the model trustworthy; they make the request attributable and constrain what a compromised model can present to downstream services.[31]

Separating Instructions, Untrusted Data, Tool Results, Memory, and Privileged Context

Trust-context separation treats every context component according to origin and authority. System policy, user intent, retrieved content, tool output, memory, and inter-agent messages should have distinct typed envelopes rather than being concatenated as undifferentiated text. Each envelope should identify its producer, integrity status, sensitivity, expiry, and permitted interpretation. A tool result may be data for summarization without becoming an instruction to call another tool.[32]

Dual-path processing can preserve this distinction: a privileged policy channel contains governing instructions, while an untrusted data channel carries external material for analysis. Cryptographic metadata can attest to channel origin and prevent ordinary content from impersonating policy. TrustGuard describes this pattern together with behavioural attestation and dynamic privilege containment, reporting 4.2% attack success against direct injection compared with 26.2% for prompt sanitisation in its deployment evaluation [33]. The result is architecture-specific and should not be generalized to all agents, but it illustrates why sanitisation alone cannot establish a trust boundary.[34]

ISSN :2584-2706 DOI: https://doi.org/10.5281/zenodo.23767601

Memory requires the same discipline. Persistent memories should be labelled as observations, preferences, instructions, credentials, or derived facts, with provenance and expiry attached. An agent should not promote a retrieved instruction into durable memory without a policy decision. The lifecycle model from AgentWard places memory between input processing and decision-making, which makes it a propagation point rather than a passive cache [6]. Context separation therefore limits both immediate instruction confusion and delayed contamination.[35]

Capability-Based Access, Least-Privilege Tooling, and Task-Bounded Authority

Capability-based access grants a concrete authority over a named operation and resource instead of relying on a broad role. A capability should specify action, target, arguments, data class, rate or quantity limit, validity interval, and delegation constraints. A read-only capability for one repository is materially different from generic filesystem access, even when both are assigned to the same agent identity.

Task-bounded authority should be monotonic unless an independent approver expands it. Progent formalizes this as automatic policy narrowing and approval-gated expansion [15]. This arrangement addresses the common failure in which an agent encounters a new instruction and silently requests broader access to complete the task. Coding-agent evidence supports the need for external policy generation: frontier models often omit permissions required by an execution chain while granting unused or sensitive permissions, and additional reasoning makes the model more consistent without resolving the mismatch [36].

Tool contracts should expose narrow, typed operations rather than general interpreters. A contract can constrain argument schemas, target namespaces, output sensitivity, and side effects before invocation. MiniScope reports that hierarchical permission reconstruction reduced permissions and operational costs relative to an LLM-based baseline while adding 1% to 6% latency [16]. The control objective is not zero autonomy. It is to make the agent's effective action space no broader than the smallest set required by the commissioned task.[37]

Execution Boundaries for MCP, Cloud Services, Coding Environments, and Multi-Agent Delegation

The **execution boundary** must sit between the model's proposed call and the external effect. For MCP, a mediation gateway should validate the caller identity, tool contract, arguments, resource scope, and policy decision before forwarding the request. AgentBound applies declarative policies to MCP servers without requiring server modification and reports that generated policies from source code reached 80.9% accuracy across 296 servers [14]. Capability containers extend this approach by ensuring that agents never access tools directly; the gateway provisions an isolated container with only the required capabilities [38].

Cloud execution requires per-task credentials, workload federation, intent-aware policy, behavioural monitoring, and immutable audit logs. ZT-AgentCloud reports less than 8% latency overhead while implementing these controls across AWS, Google Cloud Platform, and Microsoft Azure in a prototype assessed against the NIST SP 800-207 maturity model [39]. Coding environments require filesystem namespaces, restricted shells, network egress controls, secret scanning, and approval for destructive commands. Multi-agent delegation should create a new authorization decision at every hop rather than treating an upstream approval as transferable without scope checking.[40]

The reference trust boundary can be represented as follows. Identity establishes who requests; context labels establish what information is being interpreted; policy decides whether the action is permitted; isolation limits residual harm.

ISSN :2584-2706 DOI: https://doi.org/10.5281/zenodo.23767601

Zero-Trust Identity and Trust-Boundary Reference Diagram
The diagram distinguishes authentication from authorization and authorization from containment. A valid SPIFFE or OAuth credential does not itself authorize a database export; it supplies attributes to the policy decision. Likewise, an ALLOW verdict does not remove the need for sandboxing, egress filtering, or recovery. This separation reflects the claim that identity, policy, and execution control are complementary layers rather than substitutes [5], [38].

A multi-agent request should therefore carry a verifiable chain: subject identity, initiating agent, delegated agent, requested capability, context classification, policy version, and nonce or proof-of-possession material. The receiving service validates the chain independently and rejects missing, expired, audience-mismatched, or scope-expanding assertions. Such a design limits confused-deputy behavior because a downstream service can distinguish the original user from the immediate caller and evaluate both identities.

Deterministic Runtime Enforcement: Pre-Action Authorization and Evidence-Preserving Governance
Why Prompt-Level Guardrails and Semantic Legitimacy Cannot Guarantee Execution Legitimacy
Prompt-level guardrails ask the model to recognize unsafe content or refuse unsafe actions. They are useful for reducing risky proposals, but they remain probabilistic and can be bypassed by semantic obfuscation, indirect injection, task framing, or adaptive optimization. A comparative study found that hardened system prompts reduced attack success by more than 17 times in one setting, yet among the trials in which the model engaged, exploitation exceeded 55%, indicating a brittle threshold rather than continuous assurance [26]. Adaptive attacks bypassed eight evaluated defenses with success above 50% [17].

Semantic legitimacy is also narrower than execution legitimacy. An action may appear consistent with the user's goal while violating the required approval channel, target scope, data classification, delegation provenance, or temporal constraint. PCL identifies these as separate constraint classes and reports F1 of 1.0000, zero false-positive rate, and mean latency of 7.09 microseconds on 107 scenarios spanning 19 attack classes [41]. Because this is a benchmark result from a proposed engine, it demonstrates the measured behavior of that implementation rather than universal performance.[42]

ISSN :2584-2706 DOI: https://doi.org/10.5281/zenodo.23767601

The authorization decision must therefore occur synchronously before the irreversible transition. Execution-time authorization evaluates a canonical action instance against versioned policy and governed state, producing a replayable verdict and immutable decision artifact [4]. The model may propose, explain, or revise an action, but it cannot directly determine whether the external system accepts it.

Canonical Action Instances, Context-Bound Policy, and Execution-Time Authorization Decisions

A **canonical action instance** converts a variable tool request into a stable policy object. It should include the authenticated subject, agent and tool identities, action type, normalized arguments, target resource, data sensitivity, workflow or task identifier, delegation chain, current state, requested time, risk signals, and policy version. Canonicalization prevents semantically equivalent requests from receiving inconsistent decisions because of formatting, aliases, omitted defaults, or argument-order changes.

The policy engine then evaluates context rather than tool name alone. A request to read a database may be permitted for one tenant, denied for another, allowed during observation, and gated when it exports records. PCL's identity binding, channel trust, approval constraints, target validation, and coupled policy rules illustrate this context-bound design [41]. Deontic governance extends permit and prohibit rules with obligations, dispensations, conflict resolution, and post-action duties such as notification [43].

A conforming decision should be deterministic for the same canonical action, policy version, and governed state. The engine returns ALLOW, DENY, or REVIEW, together with reasons, required conditions, and a signed artifact. OAP frames the corresponding objects as Passport, Decision, and Proof, connecting cryptographic identity, capabilities, authorization outcome, and audit evidence [44]. The agent receives only the decision and permitted result, not authority to alter the policy evaluation.[45]

Risk-Adaptive Policies, Human Approval, and Allow–Deny–Review Control Paths

Static least privilege is necessary but insufficient when transaction risk changes with amount, target, data class, irreversibility, or abnormal context. **Risk-adaptive authorization** can assign low-risk operations to automatic approval, intermediate operations to supervisor-agent review, and high-risk operations to human verification. A retail-exchange prototype reported cold-path latency of 32.7 ms and warm-path latency of 4.7 ms across 120 authorization requests, with delegation narrowing blocking all invalid escalation attempts in 20 test cases [29].

The decision path should be explicit. ALLOW permits the exact canonical action and records conditions. DENY prevents execution and returns a non-sensitive reason. REVIEW suspends the action, presents the human or supervisor with target, scope, evidence, risk, and consequence, and binds the eventual approval to the same action hash. An approval should expire when arguments, target, policy, or relevant state changes. Otherwise, a model can obtain approval for one operation and reuse it for another.

Human review reduces high-impact actions and exports in simulation, but its effectiveness can degrade through user error and habituation [2]. Review therefore needs workload controls, meaningful summaries, escalation rules, and default denial for unattended expiry. AEGIS uses graduated hard-gate policies, canary runs, rollback drills, and human promotion acts before granting enforcement authority [46]. These practices connect approval to operational evidence rather than treating a click as independent assurance.

Authorization-First Retrieval and Protection Against Data-Access Bypass in RAG Pipelines

Authorization-first retrieval places access control before candidate generation reaches any learned component. The retrieval service first authenticates the requester, resolves tenant and resource scope, filters the candidate index, and only then ranks or summarizes permitted chunks. This ordering matters because once unauthorized content enters context, downstream filtering cannot guarantee that its concepts have not influenced generation [11].

The retrieval decision must be bound to the same identity and task context as subsequent tool calls. A delegated agent cannot use a broad service account to retrieve content that the originating user could not access. Chunk metadata should carry owner, classification, policy labels, source integrity, expiry, and access decision. The system should also distinguish structural leakage, answer leakage, and parametric knowledge leakage. AFR eliminated structural and answer leaks in its controlled corpus, but it explicitly identifies parametric knowledge as a boundary on what retrieval architecture can guarantee [11].

Retrieval authorization does not solve malicious content within an authorized source. Content remains untrusted data after access is granted and must retain its type label. Result parsing, source allowlists, tool dependency graphs, and pre-action authorization can reduce the chance that retrieved text becomes an instruction. IPIGuard's separation of action planning from external-data interaction illustrates this strategy, while tool-result parsing targets injected code within returned results [47], [48].

ISSN :2584-2706 DOI: https://doi.org/10.5281/zenodo.23767601

Tamper-Evident Logs, Immutable Decision Artifacts, and Counterfactual Auditing
Auditability requires more than storing the final answer. The system should record the request identity, context hashes, retrieved-source identifiers, model proposal, canonical action, policy version, decision inputs, approval evidence, tool response, resulting state, and recovery actions. A signed or hash-chained journal can reveal deletion, reordering, and post hoc alteration. OATS specifies tamper-evident cryptographic audit journals as one of five layers of model-independent agent execution [5].

Counterfactual auditing tests whether the decision depended on legitimate task information or on attacker-controlled context. DCA artifacts support comparison of direct execution, DCA, CaMeL, policy-enhanced CaMeL, and DRIFT across 949 attacked runs and 97 clean benign tasks [49]. The relevant audit question is not only "what action occurred?" but also "would the action have occurred if the suspicious content had been masked, removed, or replaced?" A changed decision indicates contextual dependence that merits review, even when the final action was blocked.[50]

Evidence must remain useful without exposing protected data. Logs should hash or tokenize secrets, separate operational evidence from payload content, and enforce access control on audit readers. Replay requires deterministic policy inputs and stable canonicalization, whereas model reasoning may remain stochastic. The architecture should therefore replay authorization and execution metadata, not pretend that a language model's internal deliberation can be reproduced exactly.

Layered Enforcement Architecture: From Input Inspection to Post-Action Evidence
A practical architecture places controls along the full action path. Input inspection labels and scans content but does not grant trust. Context separation prevents data from impersonating policy. Identity verification authenticates each workload and delegation. Capability checks restrict available operations. The policy gate canonicalizes and evaluates the proposed action. The execution boundary supplies isolation, egress controls, and scoped credentials. Post-action evidence records outcome and supports recovery.

ISSN :2584-2706 DOI: https://doi.org/10.5281/zenodo.23767601

This layered arrangement avoids two category errors. Sandboxing contains an action but does not determine whether it was authorized, while model screening can reject suspicious proposals but cannot provide deterministic enforcement. OAP reports a median authorization latency of 53 ms across 1,000 decisions, whereas capability-container mediation reports approximately 0.025 ms gateway overhead in its reference implementation [38], [51]. These figures are not directly comparable because the systems measure different paths, but they illustrate that enforcement cost should be measured at the policy and execution boundary rather than inferred from model latency.[52]

Containment and Resilience in Deployment: Sandboxing, Capability Containers, and Lifecycle Security
WebAssembly, Kubernetes, Local-Agent Sandboxes, and Infrastructure-Level Isolation
Containment limits damage when an agent, tool, or policy adapter is compromised. WebAssembly can restrict generated code to an explicit host interface, while containers and Kubernetes provide workload scheduling, resource quotas, network policy, identity attachment, and restart controls. A proposed Wasm and Kubernetes architecture uses layered isolation, policy enforcement, monitoring, and auditable execution, arguing that Wasm offers stronger isolation and lower overhead than traditional virtualized environments in its design analysis [53].

Kubernetes alone is not a complete agent security boundary. A compromised container may still access secrets, metadata services, or permitted network destinations. Healthcare deployment evidence combines gVisor kernel-level isolation, credential-proxy sidecars, allowlisted egress, and structured untrusted-content labels across nine production agents; its 90-day report describes four high-severity findings discovered and remediated by an automated audit agent [54]. The layered design is more informative than any individual technology because it addresses execution, credentials, network, and context separately.[55]

Local agents require equivalent controls despite lacking an orchestration cluster. AgentWall intercepts proposed actions, evaluates declarative policy, requests approval for sensitive operations, and records execution trails across several local-agent environments; it reports 92.9% policy-enforcement accuracy with sub-millisecond overhead across 14 benchmark tests [56]. The endpoint remains responsible for filesystem, shell, process, and network isolation.

Capability Containers and Narrow Execution Environments for Tool-Using Agents
A **capability container** is an execution unit provisioned for one narrow tool or task. The agent sends a typed request to a gateway; the gateway validates identity and policy, creates or selects a container with only required binaries, mounts, credentials, and network routes, and returns a structured result. The agent never receives the tool's unrestricted credential or host namespace. This pattern reduces the blast radius of a poisoned tool description or manipulated argument.

The reference capability-container implementation reports six layers: container isolation, network egress filtering, credential vaults, audit logging, secret scanning, and human gates. In a 1,534-sample synthetic corpus, secret detection was 100% with a confidence interval of 0.996 to 1.000, F1 was 0.991 compared with 0.779 for detect-secrets, and the false-positive rate was 0.19%; the authors also report coverage for six of seven reconstructed MCP breach scenarios [38]. Because the data were synthetic and lacked external red-team evaluation, these results support feasibility rather than production assurance.[57]

Coding environments should expose separate read, write, execute, and network capabilities. A build container may write only to a temporary workspace, execute a restricted compiler, and contact an approved package mirror. A data-analysis container may read a filtered dataset but lack export or shell capabilities. Destructive operations should require a fresh approval bound to the exact command and target, with output scanned for secrets before release.

Lifecycle Controls for Provisioning, Runtime Monitoring, Credential Rotation, Recovery, and Revocation
Security begins before the first model call. Provisioning should attest the agent image, skill set, tool contracts, policy bundle, and configuration, then issue short-lived credentials tied to the workload identity. Runtime monitoring should compare actions with the task, detect unusual destinations or volume, and record policy denials as signals rather than suppressing them. SPIFFE-based deployment demonstrates automatic credential rotation without authentication failures in continuous operation [28].

Revocation must operate at several levels: identity, capability, session, tool, container, and delegated chain. A suspected compromise should disable new tokens, terminate active sessions, quarantine memory writes, block egress, and preserve evidence before teardown. Recovery should restore known-good skills and policies, replay authorized actions where safe, and compensate external state when rollback is impossible. The lifecycle architecture proposed by AgentWard coordinates controls across initialization, input, memory, decision, and execution stages, which supports this propagation-aware response [6].

Operational promotion also requires evidence. A service should pass policy tests, canary execution, rollback drills, and approval review before receiving authority over external state. AEGIS records intent, evidence, and outcome for promotion decisions and defines kill-switch and rollback semantics [46]. These controls make deployment a continuing authorization process rather than a one-time model approval.

ISSN :2584-2706 DOI: https://doi.org/10.5281/zenodo.23767601

Applying the Architecture to Cloud Orchestration, Healthcare, Edge AI, and Coding Agents

Cloud orchestration needs federated workload identity, per-task credentials, policy enforcement across accounts and regions, anomaly monitoring, and immutable logs. ZT-AgentCloud organizes these controls around ephemeral credentials, intent-aware enforcement, identity federation, behavioural detection, and auditability, reporting less than 8% latency overhead in its prototype [39]. Healthcare adds patient-data segmentation, egress denial, database-specific policy, and evidence retention because unauthorized disclosure can become a regulatory event. The healthcare deployment's gVisor, sidecar, egress, and prompt-integrity layers provide a concrete implementation pattern [54].

Edge AI requires a cloud, edge, and device trust model because data and decisions move across resource-constrained locations. The proposed framework combines scalable identity, context-aware policy, secure deployment lifecycle controls, and real-time response across those three layers [58]. Coding agents require repository-content labeling, temporary branches, isolated terminals, secret redaction, package controls, and approval for writes outside the workspace. Across all domains, containment reduces consequence, while pre-action authorization determines whether the action should occur.

Deployment Control Matrix for Resources, Actions, Risk Levels, and Required Approvals

The following matrix translates the architecture into deployment decisions. Risk categories are design classes rather than universal ratings; each deployment should calibrate them against reversibility, sensitivity, financial or physical consequence, and delegation context.

Resource or action	Typical capability	Risk class	Minimum control	Approval path
Public documentation retrieval	Read selected sources	Low	Source and egress allowlist, untrusted-content label	Automatic ALLOW
Private document retrieval	Read authorized chunks	Moderate	Authorization-first retrieval, tenant binding, audit record	Automatic ALLOW if scope matches
Repository modification	Write bounded workspace	Moderate	Capability container, diff validation, secret scan	REVIEW for protected branches
Shell or code execution	Execute restricted commands	High	Isolated runtime, syscall and network policy, ephemeral credentials	REVIEW for destructive or external actions
Database export	Read and transmit sensitive data	High	Field-level scope, destination allowlist, DLP inspection	Human REVIEW
Cloud resource mutation	Create, delete, or alter infrastructure	High	Federated identity, canonical action, state check, rollback plan	Human REVIEW or DENY
Financial or physical transaction	Irreversible external effect	Very high	Pre-action authorization, sender constraint, evidence artifact	Explicit human approval

The matrix makes clear that sandboxing is necessary even after approval, while approval is necessary even inside a sandbox when the operation changes protected external state. This combination supplies the operational boundary required for trustworthy autonomous agency.

Evidence Synthesis and Active Debates in Agent Security Evaluation

Benchmark and Red-Team Methods: InjecAgent, AgentDojo, LivePI, AgentVigil, and AgentCanary

The evaluation literature is moving from static prompt tests toward executable, multi-step environments. InjecAgent contains 1,054 cases spanning 17 user tools, 62 attacker tools, and two harm categories; ReAct-prompted GPT-4 was compromised in 24% of cases, with reinforced attacks nearly doubling that rate [7]. AgentDojo supports controlled comparisons of attacked and benign tasks, while its released artifacts document 949 attacked runs and 97 clean tasks across several defenses [49]. LivePI extends this approach to seven input surfaces, twelve attack families, five malicious goals, and live but test-controlled email, chat, web, file, repository, and wallet interfaces [9].

AgentVigil adds automated black-box search through Monte Carlo Tree Search, obtaining 71% success against o3-mini and 70% against GPT-4o on AgentDojo and VWA-adv, approximately twice the performance of handcrafted attacks [18]. AgentCanary instead evaluates full trajectories in executable environments, separating entry conditions from impact and scoring outcome safety, security awareness, and task utility [24]. These methods measure different properties, so their percentages should not be treated as a common leaderboard.

ISSN :2584-2706 DOI: https://doi.org/10.5281/zenodo.23767601

What Controlled Trials Reveal About Framework Effects, Task Framing, and Attack Transferability

Controlled evidence weakens the assumption that a particular orchestration framework determines vulnerability. Across 3,274 trials using LangChain, MCP, and a direct Anthropic API baseline with Claude Sonnet 4, MCP added no measurable attack surface relative to the raw API baseline, whereas instruction-following task framing produced 89-91% attack success compared with 15-26% for constrained tasks [26]. Retrieval altered exposure according to semantic similarity, amplifying attacks in some LangChain conditions and suppressing them in others [26].

Transferability is substantial but conditional. AgentVigil transferred across unseen tasks and internal models [18], and automated task-universal attacks transferred to unseen tasks and out-of-distribution domains in AgentDojo, although attacks optimized on smaller open models did not transfer to GPT-5 [20]. Adaptive attacks bypassed eight defenses with success rates above 50% [17]. Thus, evaluations should vary task framing, attacker capability, model family, retrieval surface, and execution horizon rather than report one static attack rate.

The Debate Between Probabilistic Defense and Deterministic Enforcement

Probabilistic defenses remain useful for detecting suspicious content, comparing trajectories, or inferring task intent. MELON uses masked re-execution and tool comparison to identify action dependence on injected instructions, while hybrid authorization combines deterministic message controls with semantic task-tool matching [59], [60]. Their weakness is residual uncertainty: adaptive attacks can defeat model-mediated defenses, and direct policy generation can omit necessary permissions while granting sensitive ones [17], [36].

Deterministic enforcement addresses a different endpoint: whether a canonical action is permitted before execution. PCL reports F1 of 1.0000, zero false-positive rate, and mean latency of 7.09 microseconds on 107 scenarios, although this is a system-specific benchmark rather than evidence of universal performance [41]. OAP reports zero success across 879 restrictive-policy attack attempts, compared with 74.6% social-engineering success under a permissive policy in its testbed [51]. The defensible synthesis is compositional: probabilistic systems assess meaning and anomaly, while deterministic gates constrain executable authority.

Metrics for Security Architecture: Attack Success, Unauthorized Actions, False Refusals, Latency, and Auditability

Attack success rate measures whether an adversarial objective is achieved, but it should be paired with the exact consequence, such as exfiltration, command execution, or external-state mutation. **Unauthorized-action rate** is more directly architectural because it tests whether a forbidden tool call crossed the enforcement boundary. **False refusal** and benign utility capture the cost of confinement; LivePI reported preserved benign utility for its two-layer defense, while simulation work identifies human approval as protective for high-impact actions but vulnerable to user error and habituation [2], [9].

Latency must be reported by path, including warm and cold authorization, model inference, and human review. OAP measured a 53 ms median over 1,000 decisions, whereas risk-adaptive authorization measured 32.7 ms cold-path and 4.7 ms warm-path latency [29], [51]. Auditability requires replayable action instances, policy versions, identity evidence, decisions, and outcomes, rather than model explanations alone [4], [61].

Comparative Statistics and Graphs for Attack Success, Defense Robustness, and Authorization Outcomes

Reported attack rates differ in task, model, and endpoint, but they illustrate the range that evaluation protocols must explain:

Study and endpoint	Reported result
InjecAgent, GPT-4 attack success	24% [7]
LivePI, cross-model attack success range	10.7-29.6% [9]
AgentVigil, o3-mini attack success	71% [18]
AgentVigil, GPT-4o attack success	70% [18]
AIShellJack, coding-command success	Up to 84% [22]

{"type":"bar","title":"Reported Attack Success Across Agent Evaluations","xLabel":"Evaluation endpoint","yLabel":"Attack success (%)","data":[{"x":"InjecAgent GPT-4","y":24},{"x":"LivePI upper range","y":29.6},{"x":"AgentVigil o3-mini","y":71},{"x":"AgentVigil GPT-4o","y":70},{"x":"AIShellJack coding commands","y":84}]}

The chart is descriptive, not a causal comparison: executable coding attacks, automated black-box attacks, and live workflow attacks impose different objectives and denominators. Authorization outcomes are more comparable when policies and action scopes are explicit. AFR reported zero structural and answer leaks across 93 denied-query evaluations, while the OAP testbed reported zero successful restrictive-policy attacks across 879 attempts [11], [51]. Future tables should therefore separate attack prevention from utility preservation, latency, and evidence completeness.

ISSN :2584-2706 DOI: https://doi.org/10.5281/zenodo.23767601

Research Gaps: Toward Interoperable, Measurable, and Adaptive Zero-Trust Agent Governance

Limited Real-World and Longitudinal Evidence Beyond Benchmarks and Demonstration Systems

Most evidence remains benchmark-based, simulated, or short-lived. LivePI improves ecological validity through test-controlled real interfaces, and healthcare deployment reports 90 days across nine agents, but longitudinal evidence across changing users, tools, policies, and attackers remains sparse [9], [54]. AgentCanary addresses persistent state and long-horizon execution, yet broader production incidence data are still needed [24].

Unresolved Trade-Offs Among Security, Agent Utility, Latency, Autonomy, and Human Oversight

Least privilege can reduce attack impact while omitting permissions needed for task completion, and human approval introduces delay and habituation risk [2], [36]. Existing reports show low enforcement overhead in selected prototypes, but they do not establish acceptable thresholds across high-volume, multi-agent workflows [29], [38].

Missing Standards for Cross-Framework Identity, Policy Portability, and Multi-Agent Accountability

Specifications such as OAP, OATS, SPIFFE-based identity, and authenticated workflows propose portable identity, capabilities, and evidence, but cross-framework conformance remains unverified [5], [28], [44]. Multi-agent accountability also requires permission-inheritance tests and explicit attribution of delegated actions [13].

Future Evaluation Agenda for Adaptive Attacks, Physical-World Actions, and Distributed Agent Economies

Future trials should combine adaptive attackers, persistent memory, physical actuators, compromised tools, and agent-to-agent delegation. They should report unauthorized state transitions, reversibility, human intervention, policy portability, and tamper-evident evidence, extending current trajectory-based evaluation beyond textual or digital outcomes [24], [48].

II. CONCLUSION

Integrating Identity, Context Separation, Least Privilege, Deterministic Authorization, and Containment

The evidence supports treating the agent as an untrusted runtime actor. Cryptographic workload identity, separated trust contexts, task-bounded capabilities, authorization-first retrieval, deterministic pre-action decisions, sandboxing, and immutable evidence address distinct failure points. No single prompt defense can provide an execution guarantee against adaptive manipulation.

Architectural and Research Priorities for Trustworthy Agent Deployment

Deployment should standardize canonical action formats, portable identity claims, policy decision records, and fail-closed allow, deny, and review paths. Research should then test these controls in longitudinal, distributed, and physically consequential environments, measuring security, utility, latency, autonomy, and auditability together rather than optimizing attack success alone.

REFERENCES

[1]. Z. Deng *et al.*, "AI agents under threat: A survey of key security challenges and future pathways," *ACM Computing Surveys*, vol. 57, no. 7, pp. 1–36, Feb. 2025, doi: 10.1145/3716628.

[2]. R. Haider, "From search visibility to answer share: Developing and testing a causal theory of brand discoverability and consumer conversion in generative engine optimization," Jul. 2026, doi: 10.13140/rg.2.2.24339.64807.

[3]. R. Namboothiri, "Authorization-First retrieval: Enforcing least privilege in multi-agent RAG systems." Institute of Electrical and Electronics Engineers (IEEE), Mar. 05, 2026. doi: 10.36227/techrxiv.177273889.98246164/v1.

[4]. R. Haider, "Attention sovereignty in digital markets: Designing and testing consumer-owned preference agents as an alternative to surveillance-based advertising," Jul. 2026, doi: 10.13140/rg.2.2.27695.09125.

[5]. M. Mujeeb, "Permission escalation and prompt injection in autonomous AI agent systems: A layered threat analysis," *Zenodo (CERN European Organization for Nuclear Research)*. Apr. 30, 2026. doi: 10.5281/zenodo.19920653.

[6]. C. Bühler, M. Biagiola, L. Di Grazia, and G. Salvaneschi, "AgentBound: Securing execution boundaries of AI agents," *arXiv (Cornell University)*. Oct. 24, 2025. doi: 10.48550/arxiv.2510.21236.

[7]. T. Shi *et al.*, "Progent: Securing AI agents with privilege control," *ArXiv.org*. Apr. 16, 2025. doi: 10.48550/arxiv.2504.11703.

[8]. J. Zhu *et al.*, "MiniScope: A least privilege framework for authorizing tool calling agents," *arXiv (Cornell University)*. Dec. 11, 2025. doi: 10.48550/arxiv.2512.11147.

[9]. Q. Zhan, R. Fang, H. S. Panchal, and D. Kang, "Adaptive attacks break defenses against indirect prompt injection attacks on LLM agents," in *Findings of the Association for Computational Linguistics: NAACL 2025*, Stroudsburg, PA, USA: Association for Computational Linguistics, 2025, pp. 7101–7117. doi: 10.18653/v1/2025.findings-naacl.395.

[10]. Z. Wang *et al.*, "AGENTVIGIL: Automatic black-box red-teaming for indirect prompt injection against LLM agents," in *Findings of the Association for Computational Linguistics: EMNLP 2025*, Stroudsburg, PA, USA: Association for Computational Linguistics, 2025, pp. 23159–23172. doi: 10.18653/v1/2025.findings-emnlp.1258.

[11]. "European journal of prosthodontics and restorative dentistry," May 2026, doi: 10.1922/ejprd.v34i.

[12]. H. Kanaker, H. Fakhouri, N. A. Karim, M. Abuhamdeh, N. H. A. Ismail, and S. Fakhouri, "Securing tool-using AI agents against injection and authority misuse," *Computation*, vol. 14, no. 5, p. 98, Apr. 2026, doi: 10.3390/computation14050098.

ISSN :2584-2706 DOI: https://doi.org/10.5281/zenodo.23767601

[13]. D. Hofer, E. Debenedetti, and F. Tramèr, "Assessing automated prompt injection attacks in agentic environments," *arXiv (Cornell University)*. Jun. 09, 2026. doi: 10.48550/arxiv.2606.10525.
[14]. R. Haider *et al.*, "Forensic consumer psychology: An evidence-based framework for detecting deception and digital market manipulation," 2026.
[15]. Y. Liu, Y. Zhao, Y. Lyu, T. Zhang, H. Wang, and D. Lo, "'Your AI, my shell': Demystifying prompt injection attacks on agentic AI coding editors," *arXiv (Cornell University)*. Sep. 26, 2025. doi: 10.48550/arxiv.2509.22040.
[16]. V. S. Narajala and O. Narayan, "Securing agentic AI: A comprehensive threat model and mitigation framework for generative AI agents," in *2025 8th International Conference on Algorithms, Computing and Artificial Intelligence (ACAI)*, IEEE, Dec. 2025, pp. 1–6. doi: 10.1109/acai68217.2025.11406310.
[17]. P. Li *et al.*, "AgentCanary: A security evaluation framework for autonomous AI agents in real executable environments," *arXiv (Cornell University)*. Jun. 09, 2026. doi: 10.48550/arxiv.2606.10484.
[18]. R. Haider, Md. R. Amin, Md. S. Arafat, I. Hossain, and R. Ahmad, "Smart Automation: Revolutionizing Business Operations, Advertising Costs and Customer Satisfaction Through Technology," 2026.
[19]. Y. Gamage, "Indirect prompt injection in LLM agent frameworks: A comparative study," *Zenodo (CERN European Organization for Nuclear Research)*. Apr. 13, 2026. doi: 10.5281/zenodo.19558615.
[20]. V. K. P. Budigi, M. Dhanekula, S. C. Sirigiri, and S. R. Marri, "Zero-Trust identity for autonomous AI agents: Unifying SPIFFE, oauth 2.1 with PKCE, OIDC, and OPA in multi-agent architectures," *International Journal of Scientific Research in Computer Science Engineering and Information Technology*, vol. 12, no. 3, pp. 328–340, May 2026, doi: 10.32628/cseit2612335.
[21]. K. Pappu, B. Bhushan, and A. Mittal, "SPIFFE-Based zero-trust authentication for AI agent ecosystems," in *2025 International Conference on Computer and Applications (ICCA)*, IEEE, Dec. 2025, pp. 1–7. doi: 10.1109/icca66035.2025.11431026.
[22]. B. Bhushan, K. Pappu, V. Jadav, and N. Jaiswal, "Risk-Adaptive authorization for agentic AI systems," in *SoutheastCon 2026*, IEEE, Feb. 2026, pp. 1–6. doi: 10.1109/southeastcon63549.2026.11476129.
[23]. M. Gaikwad, "The agentic trust fabric: A systematization of security architectures for autonomous AI agents." Institute of Electrical and Electronics Engineers (IEEE), Nov. 27, 2025. doi: 10.36227/techrxiv.176422629.96159172/v1.
[24]. A. Goswami, "Agentic JWT: A secure delegation protocol for autonomous AI agents," *arXiv (Cornell University)*. Sep. 16, 2025. doi: 10.48550/arxiv.2509.13597.
[25]. RaiyanHaider, "Generative AI and quantum-inspired optimization: Redefining portfolio risk management and real-time capital allocation in volatile markets," *Journal of Informatics Education and Research*, Mar. 2026, doi: 10.52783/jier.v6i1.4540.
[26]. R. Haider, T. Dwivedi, A. G. G. Girish, N. Verma, B. Kashyap, and V. S. Dubey, "Neuromarketing approaches to shaping healthy consumer choices: An integrative analysis of methods, efficacy, and ethical considerations," *CLOK (University of Central Lancashire)*, Mar. 2026, doi: 10.25258/ijddt.16.4s.62.
[27]. B. P. Singh, "Securing the boundary: Trust context separation in privileged AI agent systems," *Computer Fraud & Security*, pp. 998–1009, Apr. 2026, doi: 10.52710/cfs.1012.
[28]. R. Haider, Md. F. I. Shaif, R. Ahmed, N. H. Nafi, M. R. Sumon, and M. Rahman, "Assessing the impact of influencer marketing on brand value and business revenue: An empirical and thematic analysis," *International Journal of Science and Research Archive*, vol. 16, no. 2, pp. 471–482, Aug. 2025, doi: 10.30574/ijsra.2025.16.2.2355.
[29]. R. Haider, Md. F. I. Shaif, R. Ahmed, N. H. Nafi, M. R. Sumon, and M. Rahman, "The influence of social media branding on consumer purchase behavior: A comprehensive empirical and thematic analysis," *International Journal of Science and Research Archive*, vol. 16, no. 2, pp. 460–470, Aug. 2025, doi: 10.30574/ijsra.2025.16.2.2354.
[30]. Z. Yan *et al.*, "Do Coding Agents Understand Least-Privilege Authorization?," *ArXiv.org*, 2026, [Online]. Available: https://arxiv.org/pdf/2605.14859
[31]. R. Haider, J. T. Juba, and S. Chowdhury, "Exploring the link between suicidal ideation and digital environments: The hidden impact of marketing content," *International Journal of Science and Research Archive*, vol. 16, no. 2, pp. 607–614, Aug. 2025, doi: 10.30574/ijsra.2025.16.2.2353.
[32]. R. Ledan, "The capability-container pattern: Infrastructure-Level security for autonomous AI agents," *Zenodo (CERN European Organization for Nuclear Research)*. Feb. 11, 2026. doi: 10.5281/zenodo.18614503.
[33]. A. K. Kanagala, "Zero-Trust security architecture for AI agent orchestration in cloud environments: A reference design and implementation framework," *Zenodo (CERN European Organization for Nuclear Research)*, Jul. 2025, doi: 10.5281/zenodo.20569496.
[34]. E. Meyman, "Execution-Time authorization for AI agents: A formal framework for deterministic governance boundaries," *Zenodo (CERN European Organization for Nuclear Research)*. Feb. 24, 2026. doi: 10.5281/zenodo.18764561.
[35]. R. Haider, W. A. Megha, J. T. Juba, A. Alamgir, and L. Ahmad, "The conversational revolution in health promotion: Investigating chatbot impact on healthcare marketing, patient engagement, and service reach," *International Journal of Science and Research Archive*, vol. 15, no. 3, pp. 1585–1592, Jun. 2025, doi: 10.30574/ijsra.2025.15.3.1937.
[36]. F.-A. Chitan, "Policy constraint layer (PCL): Deterministic context-bound authorization for agentic AI systems," 2025. doi: 10.5281/zenodo.19222977.

ISSN :2584-2706 DOI: https://doi.org/10.5281/zenodo.23767601

[37]. R. Haider, F. A. I. Bari, Osru, N. Afia, and M. A. khan Mugdho, "Leveraging internet of things data for real-time marketing: Opportunities, challenges, and strategic implications," *International Journal of Science and Research Archive*, vol. 15, no. 3, pp. 1657–1663, Jun. 2025, doi: 10.30574/ijsra.2025.15.3.1936.
[38]. A. Joshi, T. Finin, K. P. Joshi, and L. Kagal, "Deontic policies for runtime governance of agentic AI systems," *arXiv (Cornell University)*. Jun. 17, 2026. doi: 10.48550/arxiv.2606.19464.
[39]. U. U. Uchibeke and Ap. T. Inc., "Open Agent Passport (OAP) Specification: Authorization Specification for AI Agent Guardrails," *Open MIND*, 2026.
[40]. R. Haider, M. Bari, Md. F. I. Shaif, M. Rahman, Md. N. H. Ohi, and K. M. M. Rahman, "Quantifying the impact: Leveraging AI-Powered sentiment analysis for strategic digital marketing and enhanced brand reputation management," *International Journal of Science and Research Archive*, vol. 15, no. 2, pp. 1103–1121, May 2025, doi: 10.30574/ijsra.2025.15.2.1524.
[41]. A. K. Sharma, "AEGIS: Evidence-Driven runtime governance for autonomous AI agents," *Zenodo (CERN European Organization for Nuclear Research)*, May 2026, doi: 10.5281/zenodo.19998805.
[42]. H. An *et al.*, "IPIGuard: A novel tool dependency graph-based defense against indirect prompt injection in LLM agents," in *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, Stroudsburg, PA, USA: Association for Computational Linguistics, 2025, pp. 1023–1039. doi: 10.18653/v1/2025.emnlp-main.53.
[43]. Q. Yu, X. Cheng, and C. Liu, "Defense Against Indirect Prompt Injection via Tool Result Parsing," *ArXiv.org*, 2026, [Online]. Available: https://arxiv.org/pdf/2601.04795
[44]. A. Guo, "Decision-Space counterfactual auditing for prompt injection defense in tool-using LLM agents: Code and evaluation artifacts," *Figshare*, Jun. 2026, doi: 10.6084/m9.figshare.32657856.v1.
[45]. J. Wanger, "Open agent trust stack (OATS): A system specification for zero-trust AI agent execution," *Zenodo (CERN European Organization for Nuclear Research)*. Apr. 17, 2026. doi: 10.5281/zenodo.19636534.
[46]. R. Haider, M. Bari, Md. F. I. Shaif, and M. Rahman, "Engineering hyper-personalization: Software challenges and brand performance in AI-driven digital marketing management: An empirical study," *International Journal of Science and Research Archive*, vol. 15, no. 2, pp. 1122–1141, May 2025, doi: 10.30574/ijsra.2025.15.2.1525.
[47]. U. Uchibeke, "Before the tool call: Deterministic pre-action authorization for autonomous AI agents," *arXiv (Cornell University)*. Mar. 21, 2026. doi: 10.48550/arxiv.2603.20953.
[48]. R. Haider, M. Bari, Osru, N. Afia, and T. Al Karim, "Illuminating the black box: Explainable AI for enhanced customer behavior prediction and trust," *International Journal of Science and Research Archive*, vol. 15, no. 3, pp. 247–268, Jun. 2025, doi: 10.30574/ijsra.2025.15.3.1674.
[49]. P. P. Motgi, "SECURING AUTONOMOUS AI AGENTS: DISTRIBUTED, SANDBOXED EXECUTION ENVIRONMENTS VIA WEBASSEMBLY AND KUBERNETES," *Zenodo (CERN European Organization for Nuclear Research)*, Jun. 2026, doi: 10.5281/zenodo.20535686.
[50]. S. Maiti, "Caging the agents: A zero trust security architecture for autonomous AI in healthcare," *arXiv (Cornell University)*. Mar. 18, 2026. doi: 10.48550/arxiv.2603.17419.
[51]. R. Haider and J. Sabatina, "Harnessing the power of micro-influencers: A comprehensive analysis of their effectiveness in promoting climate adaptation solutions," *International Journal of Science and Research Archive*, vol. 15, no. 2, pp. 595–610, May 2025, doi: 10.30574/ijsra.2025.15.2.1448.
[52]. A. Aravind, "AgentWall: A runtime safety layer for local AI agents," *arXiv (Cornell University)*. Mar. 24, 2026. doi: 10.48550/arxiv.2605.16265.
[53]. R. Haider, "Navigating the digital political landscape: How social media marketing shapes voter perceptions and political brand equity in the 21st century," *International Journal of Science and Research Archive*, vol. 15, no. 1, pp. 1736–1744, Apr. 2025, doi: 10.30574/ijsra.2025.15.1.1217.
[54]. S. Xu and Y. Qian, "Zero trust architecture for edge AI security: A framework for distributed intelligence protection," *IEEE Communications Magazine*, pp. 1–10, 2026, doi: 10.1109/mcom.001.2500749.
[55]. K. Zhu, X. Yang, J. Wang, W. Guo, and W. Y. Wang, "MELON: Provable defense against indirect prompt injection attacks in AI agents," *arXiv (Cornell University)*. Feb. 07, 2025. doi: 10.48550/arxiv.2502.05174.
[56]. Y. Zhang, X. Deng, J. Wu, Y. Xiao, K. Xu, and Q. Li, "AgentWard: A lifecycle security architecture for autonomous AI agents," *arXiv (Cornell University)*. Apr. 27, 2026. doi: 10.48550/arxiv.2604.24657.
[57]. M. El Helou, B. Ryder, C. Troiani, J. Diaconu, H. Muyal, and M. Yannuzzi, "Hybrid inspection and task-based access control in zero-trust agentic AI," *arXiv (Cornell University)*. May 04, 2026. doi: 10.48550/arxiv.2605.02682.
[58]. J. Berroa, "Agent authority infrastructure: A framework for runtime authority, approval, and evidence in AI agents," *Zenodo (CERN European Organization for Nuclear Research)*, May 2026, doi: 10.5281/zenodo.20237200.
[59]. 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*, Stroudsburg, PA, USA: Association for Computational Linguistics, 2024, pp. 10471–10506. doi: 10.18653/v1/2024.findings-acl.624.
[60]. R. Haider, "The algorithmic memory of brands: Developing a longitudinal theory of how large language models learn, retain, distort and correct brand knowledge proposal field details," Jul. 2026, doi: 10.13140/rg.2.2.26017.36966.
[61]. L. Zhao, A. Bhaskar, and E. Dobriban, "LivePI: More realistic benchmarking of agents against indirect prompt injection," *arXiv (Cornell University)*. May 18, 2026. doi: 10.48550/arxiv.2605.17986.