Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Beyond Prompt Injection: Trust-Boundary Security Assurance for LLM-Integrated and Agentic Applications

Nazar Waheed

nwaheed@hct.ac.ae

Higher Colleges of Technology

Research Article

Keywords: LLM security, agentic AI, prompt injection, security assurance, trust boundaries, Model Context Protocol

Posted Date: September 3rd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10798245/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Beyond Prompt Injection: Trust-Boundary Security Assurance for LLM-Integrated and Agentic Applications

Nazar Waheed

Abstract Large language model (LLM) applications increasingly combine probabilistic language interpretation with retrieval, persistent memory, external tools, and delegated enterprise credentials. This creates a system-security problem in which untrusted semantic content can cross trust boundaries and acquire operational authority. Existing work provides prompt-injection attacks, agent benchmarks, risk taxonomies, and governance guidance, but less directly addresses how those threats should be translated into verifiable controls for a deployed application. This paper presents a trust-boundary security assurance framework for LLM-integrated and agentic systems. The framework models the application as a mediated-authority system, identifies six architectural threat surfaces and seven recurring trust boundaries, and represents attack paths through source, mechanism, boundary, privilege, asset, and outcome. It then operationalizes the model through a six-stage assurance process covering system decomposition, authority mapping, adversarial scenario construction, boundary-control verification, containment and detection analysis, and evidence-based reporting. A dedicated Model Context Protocol (MCP) profile distinguishes protocol security, semantic trust, and enterprise authorization. The framework further connects defensive controls to observable evidence and proposes a decomposed measurement profile covering attack success, boundary interception, benign-task preservation, blast-radius characteristics, detection latency, and undetected events. A worked enterprise RAG–MCP scenario illustrates how the framework traces an indirect injection from retrieved content through delegated authority to an authorization decision and auditable evidence. The contribution is methodological rather than an empirical claim of defense superiority: it provides a traceable structure for converting evolving LLM and agent threats into testable security claims and application-level assurance evidence.

Keywords LLM security · agentic AI · prompt injection · security assurance · trust boundaries · Model Context Protocol

N. Waheed
Higher Colleges of Technology, Abu Dhabi, United Arab Emirates
E-mail: nwaheed@hct.ac.ae
ORCID: 0000-0001-5679-5365

1 Introduction

Large language models (LLMs) are moving from isolated text generation into applications that retrieve information, maintain state, invoke external tools, and act on behalf of users. Retrieval-augmented generation (RAG), reasoning-and-acting agents, learned tool use, and general agent benchmarks illustrate this transition from language generation toward software systems that retrieve, decide, and act [1,2,3,4]. The security consequence is significant: natural-language content can influence not only model output, but also resource selection, tool arguments, authorization-sensitive operations, and persistent state.

Prompt injection makes this change visible. Early work demonstrated direct instruction hijacking and prompt leakage, while subsequent studies showed that safety training can fail under adversarial prompting and that practical attacks exhibit substantial diversity [5,6,7]. Indirect prompt injection broadens the threat because malicious instructions may arrive through webpages, retrieved documents, emails, tool responses, or other content later consumed by the application [8, 9,10]. Once an LLM is connected to tools, the same instruction-following failure can become an application-security event. InjecAgent, AgentDojo, ToolEmu, and

Agent Security Bench demonstrate this shift in increasingly realistic environments [11,12,13,14].

The impact of an identical model-level failure therefore depends on the surrounding architecture. A chatbot with no privileged integration may produce an undesirable response but little direct system impact; an enterprise agent operating through broad service credentials may expose records, modify files, invoke external services, or create durable state. Recent agent-security work accordingly emphasizes information flow, delegated authority, persistent state, and composition rather than model behavior alone [15,16,17,18]. Secure-by-design proposals such as structured query separation, capability-oriented execution, prompt-flow integrity, and privilege-preserving design patterns likewise move critical controls outside the model [19,20,21,22].

Security assurance research is also beginning to address LLM-specific risk. Karlsen et al. propose a quantitative assurance framework that translates LLM security requirements and vulnerabilities into measurable assurance levels and demonstrates the approach through a case study [23]. That work is an important assurance-oriented predecessor, but it addresses a different problem from the one considered here. Our focus is not a scalar assurance score for LLM vulnerabilities; it is the end-to-end security of agentic applications in which untrusted information traverses retrieval, orchestration, identity, tool, and persistence boundaries. The central assurance question is therefore where semantic influence becomes operational authority, which independent control should interrupt that transition, and what evidence demonstrates that the control operated correctly.

Standards and practitioner guidance provide complementary foundations. NIST's AI RMF and Generative AI Profile frame risk as a lifecycle concern [24, 25]; MITRE ATLAS catalogs adversarial techniques against AI-enabled systems [26]; and OWASP guidance now spans both LLM-specific and agentic risks [27,28]. These resources identify threats and control objectives, but they do not by themselves provide a deployment-specific chain from attacker influence to trust boundary, authority, control, verification, and evidence.

This paper addresses that operational assurance gap. We treat the LLM-enabled application as a composed software system in which probabilistic language interpretation interacts with conventional security mechanisms such as access control, credential management, segmentation, logging, output handling, and recovery. The contribution is deliberately bounded and consists of four elements:

C1: a mediated-authority system model that identifies six architectural threat surfaces, seven recurring trust boundaries, protected assets, and a traceable attack-path representation;

C2: a six-stage assessment methodology that translates threat knowledge into boundary-oriented adversarial scenarios, control verification, containment analysis, and evidence-based findings, together with a worked enterprise RAG–MCP instantiation;

C3: a defense and MCP assurance profile that separates protocol correctness, semantic trust, enterprise authorization, and recovery while mapping controls to observable evidence; and

C4: a decomposed measurement profile that reports attack success, boundary interception, legitimate-task preservation, blast-radius characteristics, detection latency, and undetected events without collapsing heterogeneous properties into an unvalidated universal score.

Three research questions guide the work:

RQ1: How should trust, authority, and attack paths be represented in LLM-integrated and agentic applications that combine retrieval, tools, identity, and persistent state?

RQ2: How can known LLM and agent threats be converted into a repeatable security-assurance process with explicit control objectives and verification evidence?

RQ3: Which decomposed measurements are most useful for reporting attack feasibility, containment, legitimate functionality, and detection without implying unsupported precision?

The paper is methodological rather than experimental. Section 2 explains how the framework was constructed and bounded. Sections 3–4 define the system and threat model; Section 5 presents the six-stage assessment process; Section 6 provides a worked enterprise RAG–MCP assurance case; Section 7 specializes the analysis for MCP and agentic tool ecosystems; Sections 8–9 define control and measurement guidance; and Sections 10–13 position, discuss, and delimit the framework.

2 Methodological Approach and Framework Construction

2.1 Research Design and Source Selection

This study uses a structured conceptual-synthesis design. Its objective is framework construction rather than prevalence estimation, bibliometric mapping, or comparative model evaluation. The evidence base combines

four source classes: (i) primary research on prompt injection and indirect instruction attacks; (ii) studies and benchmarks addressing tools, agents, retrieval, memory, and persistent state; (iii) established security-testing, control, and risk-management guidance; and (iv) current practitioner frameworks and protocol specifications relevant to deployed LLM applications [29,12,30,31,32,33].

Source selection was purposive rather than exhaustive. A source was retained when it contributed directly to at least one analytical need: an attack mechanism, an architectural trust boundary, a privilege or persistence failure, an assessment procedure, a control objective, an evaluation measure, or an integration interface. For LLM- and agent-specific research, emphasis was placed on work published from 2022 through August 2026, supplemented by earlier foundational security standards and measurement literature where required. Peer-reviewed publications were preferred when an archival version existed; authoritative NIST, OWASP, MITRE, and MCP materials were used for operational guidance and current specifications. Broader surveys were used to check coverage and terminology rather than as substitutes for primary evidence [34,35,15].

This procedure is intentionally not presented as a systematic literature review. No claim is made that the corpus is exhaustive, and publication counts are not used to infer prevalence or importance. To improve transparency, targeted searches used combinations of terms including "prompt injection", "indirect prompt injection", "LLM agent security", "RAG poisoning", "memory poisoning", "tool poisoning", "MCP security", "AI red teaming", and "LLM security assurance", followed by citation chaining from relevant primary papers, surveys, and standards. The literature cut-off for LLM- and agent-specific material was August 2026. The methodological aim is to establish a traceable synthesis of concepts that recur across research, benchmarks, standards, and deployed integration patterns.

2.2 Framework Construction Procedure

Framework construction proceeded through four analytical steps. First, recurring attack and failure mechanisms were extracted from the evidence base, including direct and indirect injection, retrieval poisoning, memory manipulation, tool steering, metadata poisoning, authorization misuse, unsafe downstream interpretation, and persistent influence. Second, these mechanisms were abstracted into system-security concepts: source of influence, architectural surface, trust boundary, privilege-bearing component, protected asset, security outcome, and observable evidence. Third, the resulting concepts were mapped to established security-assessment principles such as system scoping, least privilege, independent authorization, control verification, logging, and recovery [36,37,38,39]. Fourth, the framework was checked for consistency against representative agent benchmarks, OWASP and MITRE threat knowledge, NIST assessment guidance, and the current MCP architecture and authorization specification [11,12,28,26,40].

The fourth step is a form of analytical triangulation, not empirical validation. It tests whether the framework can represent threat and control concepts already documented across independent source families; it does not demonstrate that use of the framework improves security outcomes. During consolidation, a category was retained only when it represented a distinct security-relevant enforcement locus or state transition; concepts that differed mainly in terminology but mapped to the same enforcement point were merged. Where feasible, retained categories were checked against more than one source family (for example, primary attacks together with a benchmark, standard, or practitioner taxonomy). The synthesis and abstraction were performed by a single author, so no inter-rater agreement is claimed. The result should therefore be understood as an *evidence-informed analytical model* intended for subsequent application and independent empirical evaluation.

2.3 Unit of Analysis, Outcomes, and Traceability

The unit of analysis is the AI-enabled application rather than the foundation model in isolation. For assurance purposes, an attack path is represented as

$$\mathcal{P} = \langle E, V, B, \Pi, A, O \rangle, \tag{1}$$

where E is the entry source of attacker-controlled or otherwise untrusted influence, V the mechanism by which that influence affects the application, B the trust boundary crossed, Π the privilege or authority available after the transition, A the protected asset or external capability exposed, and O the resulting security outcome. The tuple is a traceability device rather than a probabilistic risk equation. Using E and Π avoids conflating the entry source with the architectural-surface symbols S_I–S_P introduced later.

Three primary unauthorized outcome classes are used throughout the paper: **disclosure**, **modification**, and **execution or action**. **Persistence or propagation** is treated as a cross-cutting temporal consequence that can accompany any of the three primary outcomes, rather than as a mutually exclusive category. A content-safety violation is not automatically classified as a cy-

bersecurity compromise unless it contributes to one of these system-security consequences.

Each threat is then connected to an assurance chain:

$$\text{Threat} \rightarrow \text{Trust Boundary} \rightarrow \text{Control Objective} \rightarrow \text{Verification} \rightarrow \text{Evidence}. \quad (2)$$

This structure follows established control-assessment practice, in which the presence of a stated control is insufficient without evidence that it operates as intended [30,31]. It also provides the organizing principle for the later assurance process: threat knowledge identifies what should be exercised; the architecture determines where the transition occurs; the control objective specifies what should prevent or constrain it; and the evidence supports the resulting assurance claim.

2.4 Use of Generative AI in Manuscript Preparation

OpenAI ChatGPT was used during manuscript development as an AI-assisted writing and editorial tool to support restructuring, drafting and refinement of prose, LaTeX formatting, and literature discovery. AI assistance was not used to generate experimental data or empirical results. All AI-assisted material was reviewed and revised by the manuscript's author, who retains responsibility for the arguments, citations, and final content. No AI system is listed as an author.

2.5 Scope and Validity Boundaries

The framework addresses post-deployment security risks in LLM-integrated and agentic applications, including direct and indirect prompt injection, retrieval-mediated influence, memory and context manipulation, tool misuse, authorization failures, insecure integration, unsafe output handling, and persistent state manipulation. Training-time compromise of the foundation model, proprietary weight extraction, hardware attacks, and general alignment are outside the core scope. Multi-agent propagation is considered where it affects trust or authority, but a complete multi-agent security model is left for future work.

The framework also assumes that model-level instruction-following failures cannot be eliminated reliably. Its security objective is therefore twofold: reduce susceptibility where possible and ensure that a model-level failure does not automatically become unrestricted enterprise impact. Because this paper does not report an empirical deployment, the framework's effectiveness should not be inferred from its internal structure alone; Section 12 identifies the validation work required before stronger effectiveness claims can be made.

3 System Model, Assets, and Trust Boundaries

3.1 Mediated-Authority System Model

We model an LLM application as a *mediated-authority system*. The model typically does not hold enterprise authority directly; the surrounding application supplies credentials, service identities, tool connectors, workflow permissions, and approval mechanisms. The model influences how that delegated authority is exercised by interpreting input and proposing retrievals, plans, tool calls, or outputs.

A representative architecture contains eight logical components: interaction/ingestion, orchestration, model inference, retrieval/context, memory/persistent state, tools/external actions, identity/authorization, and downstream output handling. The implementation may collapse or distribute these components, but the security analysis focuses on where trust, privilege, or persistence changes.

3.2 Trust Boundaries

Seven recurring boundaries capture the main transitions:

1. **B1: External input → application context.** User or external content enters the AI pipeline.
2. **B2: Retrieved content → decision context.** External data is selected and introduced into model context; indirect injection and RAG poisoning are relevant here [8,10,41].
3. **B3: Context → orchestration/planning.** Information that may be valid as data begins to influence routing or control decisions.
4. **B4: Model proposal → privileged tool execution.** A model-generated call becomes an external action; tool-agent benchmarks show the importance of this transition [12,11].
5. **B5: Application identity → enterprise resource.** The application presents delegated or service credentials to protected systems. Least privilege and explicit authorization remain conventional requirements [38].
6. **B6: Transient interaction → persistent state.** Content is written into memory or reusable state, enabling delayed influence [42,43].
7. **B7: Generated output → downstream interpretation.** LLM output is rendered, stored, executed, or consumed by another component and must be handled according to its destination context [44].

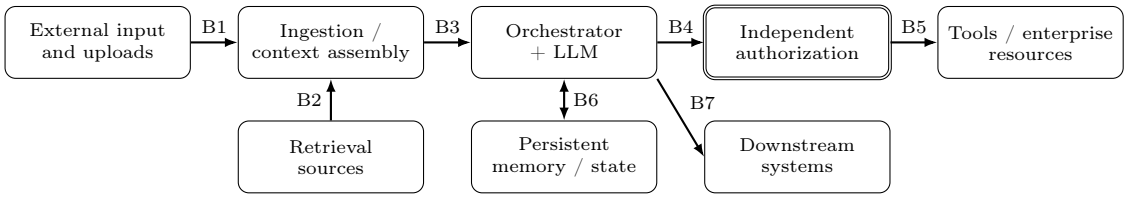


Fig. 1 Reference trust-boundary model. The boundaries are logical enforcement transitions rather than mandatory deployment tiers; implementations may combine components while preserving the same security questions about provenance, authority, persistence, and downstream interpretation.

Table 1 Protected assets and principal security objectives.

Asset	Examples	Security objective
Business / user data	PII, financial records, documents, source code, RAG corpora	Prevent unauthorized disclosure and cross-user access
Credentials / authority	API keys, OAuth tokens, service identities, delegated scopes	Bind authority to identity, purpose, resource, and minimum scope
Control artifacts	System prompts, policies, tool schemas, workflow configuration	Preserve integrity and governed change
Persistent state	Memory, RAG indices, profiles, prompt stores, cached plans	Preserve provenance, integrity, revocability, and traceability
External capabilities	Email, CRM, databases, code execution, cloud APIs	Prevent unauthorized read/write/execute actions
Downstream systems	Browsers, dashboards, databases, other agents	Treat generated output as untrusted for its destination context

The critical distinction is between *information relevance* and *instruction authority*. An authorized document can be relevant without being authorized to direct an agent; a trusted connector can return attacker-controlled content; and a valid model proposal is not itself an authorization decision.

3.3 Security Invariants and Adversary Model

The framework uses five invariants: (i) provenance is preserved across context assembly; (ii) data does not gain instruction authority solely by entering context; (iii) consequential actions are independently authorized; (iv) delegated privileges are bounded to the legitimate workflow; and (v) persistent state is attributable and recoverable. Privacy-oriented architectures such as AirGapAgent illustrate the broader value of reducing unnecessary information exposure across agent boundaries [45].

The adversary may be a direct external user, an indirect-content attacker, a constrained authenticated user attempting to exceed their scope, or a compromised/malicious integration. Depending on the scenario, the adversary may supply arbitrary text, influence retrievable content or tool responses, attempt to shape tool arguments, manipulate persistent state, or adapt attacks based on observed outputs. The baseline model does not assume compromise of the model provider, operating-system administrator, or enterprise identity infrastructure.

4 Architecture-Oriented Threat Surface Classification

4.1 Classification Principle

Mechanism-oriented labels such as "prompt injection" are useful but do not by themselves identify where an enterprise control should be placed. The same mechanism can enter through a user message, retrieved document, tool descriptor, or stored memory. We therefore classify threats by the architectural surface through which attacker influence is converted into authority or persistent effect:

$$\mathcal{S} = \{S_I, S_R, S_O, S_T, S_A, S_P\}, \tag{3}$$

representing input/ingestion, retrieval/context, orchestration, tools/actions, identity/authorization, and output/persistent state.

4.2 Six Security Surfaces

S_I – **Input and ingestion.** Direct injection, malicious uploads, field concatenation, encoding differences, and parser/normalization inconsistencies can blur data and

instruction boundaries. Practical prompt-injection and jailbreak research shows both the diversity of attacks and the limits of static filtering [5,7,29]. Defenses such as spotlighting and structured query separation aim to preserve this distinction [46,19].

S_R – **Retrieval and contextual data.** Indirect injection and RAG poisoning exploit the fact that retrieved content can be both relevant and attacker-controlled [8,10,41]. Security therefore requires source provenance, user-appropriate retrieval authorization, and separation of retrieved data from control instructions.

S_O – **Orchestration and instruction handling.** Prompt construction, planning, routing, delegation, and approval logic determine whether model interpretation becomes application control. Prompt-flow integrity and capability-oriented designs show why the orchestration layer should enforce explicit trust and privilege boundaries [21,20].

S_T – **Tools and external actions.** Threats include tool steering, semantically malicious but syntactically valid arguments, unsafe tool composition, and poisoned descriptors. Tool-agent benchmarks demonstrate how untrusted observations can propagate into external actions [13,12,14]. Dynamic tool ecosystems add metadata poisoning, discussed further in Section 7.

S_A – **Identity, delegation, and authorization.** Excessive privilege, confused-deputy behavior, credential exposure, approval bypass, and transitive privilege convert model errors into enterprise compromise. The control objective is to bind actions to authenticated principals, resources, operation types, and explicit policy rather than to model confidence.

S_P – **Output and persistent state.** Unsafe rendering or execution can turn generated output into a conventional injection path, while memory or RAG writes can preserve adversarial influence across sessions [42,43]. Persistent state therefore requires write authorization, provenance, versioning, and rollback.

4.3 Cross-Surface Attack Chaining

High-impact attacks are often compositional. A malicious document can cross S_R, influence planning in S_O, trigger a tool in S_T, exploit over-scoped authority in S_A, and create exfiltration or persistent state in S_P. The attack is not adequately described by the initial prompt alone. Three cross-cutting amplifiers recur: *delegated authority*, *persistence*, and *transitive trust*.

The classification complements rather than replaces OWASP, MITRE ATLAS, and benchmark taxonomies [27,28,26]. Its purpose is operational: to project known threats onto the deployed architecture so that a control and verification point can be identified.

5 Six-Stage Security Assurance Assessment Framework

5.1 Design Principles

The assurance process adapts conventional penetration testing and control assessment to LLM-mediated systems [30,31]. It follows four principles: assess the complete application rather than the model alone; separate attacker influence from authorization; define explicit security outcomes; and prefer observable evidence over model text alone. The working assumption is that model-level failures remain possible, so the key question is what independent mechanisms prevent them from becoming unauthorized enterprise actions. Contemporary AI red-teaming guidance similarly emphasizes evaluation across model, implementation, infrastructure, and runtime layers [47,32,48].

5.2 Stage 1: Scope and Decompose

Identify the system under assessment, including interfaces, model endpoints, orchestration, prompts, retrieval sources, memory, tools/MCP servers, identities, credentials, downstream destinations, and monitoring. Direct and indirect input paths should be recorded separately. The output is a component/integration inventory and data-flow view sufficient to locate the boundaries in Section 3.

5.3 Stage 2: Map Trust and Authority

For every tool or protected resource, record the executing principal, operation class, scope, credential lifetime, and approval requirement. Particular attention should be given to *delegation mismatch*, where the application has more authority over a resource than the initiating user. This condition is not automatically a vulnerability, but it requires a separate authorization mechanism. Least privilege and zero-trust principles provide the conventional basis for this analysis [37,38].

5.4 Stage 3: Construct Threat-Aligned Scenarios

Each scenario should represent a plausible path from attacker influence to a prohibited outcome rather than an arbitrary jailbreak prompt. We represent a scenario as

Table 2 Threat surfaces, representative failures, and assurance evidence.

Surface	Representative failures	Primary boundary	Representative evidence
S_I	Direct injection; parser/field confusion; malicious uploads	B1	Input inventory, normalization traces, provenance labels
S_R	Indirect injection; RAG poisoning; unauthorized retrieval	B2	Source/chunk IDs, retrieval policy, context provenance
S_O	Planning/routing hijack; instruction hierarchy failure	B3	Context construction, policy and routing decisions
S_T	Tool steering; argument abuse; descriptor poisoning	B4	Tool inventory, proposed/executed calls, validation outcomes
S_A	Excessive privilege; confused deputy; token/scope misuse	B5	Identity mapping, scopes, resource policy, approval logs
S_P	Unsafe rendering; memory poisoning; delayed activation	B6–B7	State diffs, version history, renderer and downstream logs

$$Q_i = \langle E, S, B, \Pi, A, O, C, \mathcal{E} \rangle, \tag{4}$$

where E is the entry/influence source, S the affected surface, B the boundary, Π the available authority, A the asset, O the prohibited outcome, C the expected control, and $\mathcal{E}$ the required evidence. Benchmarks such as JailbreakBench, InjecAgent, AgentDojo, Agent Security Bench, and ToolEmu can supply scenario patterns, but they should be adapted to the actual deployment [49, 11, 12, 14, 13]. Adversarial cases should be paired with benign analogues where feasible so that security and legitimate-task preservation can be assessed together.

5.5 Stage 4: Exercise Boundary Controls

Under explicit authorization, execute the scenarios and record where the attack path is interrupted. Testing should distinguish model-level compliance from system-level compromise. For example, if a malicious document causes the model to propose an unauthorized CRM write but a deterministic policy denies execution, the model is susceptible but the protected outcome is contained.

Controls should be tested at the relevant surface: input/context separation at S_I; provenance and retrieval authorization at S_R; policy and routing at S_O; tool and argument restrictions at S_T; principal/resource authorization at S_A; and output/state governance at S_P. Evidence should include inputs, context or identifiers, retrieval provenance, policy decisions, proposed and executed tool calls, authorization outcomes, state changes, and relevant alerts.

5.6 Stage 5: Analyze Containment, Persistence, and Detection

For any successful or near-successful path, determine which data and operations are reachable, whether permissions extend across systems, whether external egress is possible, whether the influence persists, and whether useful detection signals are generated. Persistent-memory and knowledge-base attacks make delayed activation particularly important [42, 43]. The goal is to characterize impact without escalating beyond the authorized test scope.

5.7 Stage 6: Report Assurance Evidence and Remediation

Each finding should state the security outcome, source, affected surface and boundary, authority used, protected asset, expected control, observed behavior, containment, evidence, and remediation. Root-cause categories include provenance failure, data/instruction confusion, over-scoped credentials, missing resource authorization, unsafe tool policy, persistent-state governance failure, and insufficient telemetry.

Remediation should prioritize removing unnecessary authority, enforcing authorization, preserving trust boundaries, reducing model susceptibility, and improving detection. Findings and scenarios should be retained as regression assets because model, prompt, tool, retrieval, and authorization changes can alter established assumptions.

The process complements conventional application/API/infrastructure testing rather than replacing it. LLM-specific analysis is needed because malicious intent may be embedded in semantically valid data, distributed across functioning components, or retained in natural-language state rather than executable code.

6 Worked Illustrative Assurance Case

This section instantiates the framework in a deliberately hypothetical enterprise scenario. It is not an experiment, benchmark result, or claim that the described controls achieve a particular failure rate. Its purpose

Table 3 Six-stage assurance process.

Stage	Objective	Primary question	Core evidence
1	Scope/decompose	What receives untrusted information or exercises authority?	Component, input, tool, retrieval, and memory inventory
2	Map trust/authority	Where do trust or privilege change?	Boundary map, principals, permission matrix, scopes
3	Construct scenarios	How can influence plausibly reach a protected outcome?	Scenario register, success predicate, expected control
4	Exercise controls	Which boundary blocks or permits the path?	Context, policy, tool, authorization, and state traces
5	Analyze containment	What can a failure reach, persist in, or evade detection from?	Reachability, persistence, alerting, rollback evidence
6	Report/remediate	What failed and what change most directly reduces risk?	Finding, root cause, remediation, regression case

is to demonstrate how the abstractions in Sections 3–5 can be converted into a concrete assurance argument.

6.1 Scenario and Attack Path

Consider an internal service-desk assistant used by employees to answer policy questions and perform limited customer-support actions. The application authenticates users through enterprise single sign-on, retrieves internal knowledge-base documents through RAG, and uses an MCP host to connect three logical services: a knowledge server, a CRM server, and a messaging server. The CRM server exposes customer lookup and record-update capabilities; the messaging server can send approved case communications. Retrieved documents may include material contributed by business teams and are therefore authorized for retrieval but not automatically trusted as instructions.

Assume that one retrieved document has been maliciously modified to contain an indirect instruction that attempts to redirect the assistant from the employee's benign policy question toward retrieving customer information and sending it to an external recipient. The security-relevant path is not merely that the model sees or follows the instruction. The path becomes consequential only if the retrieved content can influence orchestration, obtain access to CRM data, and cause an externally visible messaging action.

Using Equation 1, the scenario can be represented as

$$\mathcal{P}_{\text{case}} = \langle E_{\text{doc}},\ \text{indirect injection},\ B2 \rightarrow B5,\ \Pi_{\text{CRM+mail}},\ A_D/A_X,\ O_{\text{disclosure/action}} \rangle. \tag{5}$$

where E_{doc} denotes the retrieved document as the entry source on surface S_R; the influence crosses retrieval and orchestration boundaries, reaches tool and authorization boundaries, and attempts to use delegated CRM and messaging authority to produce unauthorized disclosure and action.

6.2 Applying the Six Assurance Stages

Stage 1—scope and decompose. The assessor records the user interface, RAG pipeline, knowledge source, model/orchestrator, MCP host and clients, CRM and messaging servers, user identity, delegated credentials, approval logic, persistent logs, and downstream email destination. Customer data and external messaging capability are identified as protected assets.

Stage 2—map trust and authority. The critical distinction is between the employee's requested objective and the application's available capability. A weak deployment might use a broadly privileged service identity that can read many CRM records and send arbitrary external messages. A stronger deployment binds CRM access to the initiating user or workflow, separates read and write/send capabilities, restricts recipients and resources, and requires additional authorization for high-impact operations. The assessment therefore records not only which tool exists, but the principal, scope, resource, operation, credential lifetime, and approval requirement associated with it.

Stage 3—construct the scenario. The adversarial case pairs the maliciously modified knowledge document with an otherwise legitimate employee request. The prohibited success predicate is defined as unauthorized CRM disclosure to an external destination, not merely model compliance with the embedded instruction. A benign analogue uses the same retrieval and tool path for an authorized customer-support action so that protective controls are not rewarded simply for disabling useful functionality.

Stage 4—exercise boundary controls. The expected defense is independent of whether the model notices the injection. Retrieved content should retain provenance and low instruction authority; CRM and messaging operations should then be evaluated against the authenticated principal, requested resource, action type, recipient, and workflow context. The model may still propose a prohibited tool call. If a deterministic authorization layer rejects the CRM access or external send,

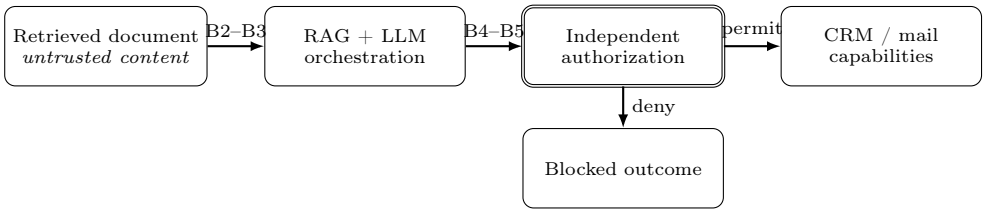


Fig. 2 Worked RAG–MCP attack path. Retrieved content may influence the model, but a separate authorization boundary determines whether the proposed CRM or messaging action can execute.

the test records model susceptibility but system-level containment. Conversely, if a broad service credential permits the operation merely because the tool call is syntactically valid, the failure is attributed to authority and authorization design rather than only to prompt robustness.

Stage 5—analyze containment, persistence, and detection. The assessor characterizes what the path could reach if the authorization control failed: the breadth and sensitivity of CRM data, read/write capability, cross-system reach, external egress, and any persistent state created by the interaction. Detection analysis asks whether the retrieved document identifier, proposed tool call, policy decision, principal, recipient, and resulting resource effect can be correlated through a common trace.

Stage 6—report evidence and remediation. The resulting assurance claim can be stated narrowly: *retrieved documents cannot independently authorize external disclosure of CRM data through the assessed tool path*. Supporting evidence should include the retrieved source and chunk identifier, provenance label, constructed-context or protected trace identifier, proposed tool call, authorization decision, executing principal and scope, messaging policy result, and confirmation that no prohibited downstream side effect occurred. The claim should also state its limitation: it does not cover alternative egress paths that bypass the assessed policy enforcement point.

The example also clarifies how the measurement profile in Section 9 should be used. A single illustrative path does not justify reporting an ASR, BIR, or detection distribution. If the scenario were executed repeatedly in an authorized test environment, ASR would be based on the prohibited system outcome, BIR would identify which boundary intercepted adversarial transitions, BTSR would use the paired benign workflow, BRC would characterize reachable impact, and TTD/UDR would characterize monitoring. The framework therefore separates demonstration of the assessment logic from empirical claims about control effectiveness.

7 Agentic Tool Ecosystems and Model Context Protocol Security

7.1 MCP as an Assurance Boundary

MCP standardizes how AI applications interact with external tools, resources, and prompts. The current architecture uses host–client–server roles, with the host coordinating policy and context and clients mediating individual server relationships [33,50]. Standardization improves interoperability but also concentrates security-sensitive metadata and capabilities behind a common semantic interface.

Three layers must be distinguished:

$$\mathcal{M}_{sec} = \langle M_{\text{protocol}}, M_{\text{semantic}}, M_{\text{enterprise}} \rangle, \tag{6}$$

where protocol security covers communication and protocol authorization, semantic security covers how model-visible tools/resources/prompts influence reasoning, and enterprise security covers business authorization, least privilege, segmentation, and downstream effects. The tuple is descriptive rather than additive: strength at one layer does not compensate automatically for failure at another.

7.2 Protocol Security and Authorization

The 2026 MCP revision introduced a stateless core, explicit routing information, extension mechanisms, and authorization hardening [51]. For HTTP-based deployments, the authorization profile builds on OAuth concepts including resource indicators, audience/issuer validation, client identity, scoped authorization, and incremental authorization [40]. These mechanisms reduce protocol-level credential confusion but cannot determine whether an LLM-generated business action is appropriate for the initiating user.

The core distinction is

$$\begin{aligned}\text{Protocol Access} &\neq \text{Semantic Trust} \\ &\neq \text{Action Authorization}.\end{aligned} \tag{7}$$

Table 4 Worked assurance trace for the hypothetical RAG–MCP scenario.

Assurance element	Illustrative interpretation	Evidence required
Threat/source	Indirect instruction embedded in retrieved knowledge content	Source/chunk ID, provenance and ingestion history
Trust transition	Retrieved data influences orchestration and proposes privileged tool use	Context/routing trace, proposed CRM and messaging calls
Authority	CRM data access and external messaging capability	Principal, token/scope, resource and operation policy
Expected control	User-bound authorization, resource/recipient restrictions, step-up approval where required	Policy decision, approval record, allow/deny reason
Security outcome	Unauthorized disclosure/action is prevented even if model-level influence occurs	Executed-call log and downstream resource/message state
Containment	Reach is limited to authorized data/actions; alternate egress is excluded or separately assessed	Permission matrix, BRC dimensions, network/tool inventory
Detection	Causal path can be reconstructed and compromised content or credentials can be revoked	Trace ID, alert/log correlation, rollback or revocation evidence

A client can be legitimately authorized to contact a server while receiving malicious or misleading semantic content from it; a tool invocation can satisfy the protocol schema while violating enterprise policy.

7.3 Principal MCP Threat Paths

Tool metadata poisoning. Models often see tool names, descriptions, schemas, or annotations before invocation. The specification itself cautions that tool annotations should not be trusted merely because they are structured [52]. MCPTox and MCP-ITP show how poisoned metadata can steer agents, including cases where one poisoned descriptor induces use of another legitimate privileged tool [53,54]. Controls therefore include server trust classification, descriptor review/change detection, provenance labels, and independent authorization after tool selection.

Resource-mediated injection. MCP resources can contain files, records, schemas, or other content [55]. A resource may be authorized to read and still contain attacker-controlled instructions. Assurance must verify provenance, user-specific access, data/instruction separation, and whether resource content alone can drive a consequential action.

Prompt/workflow manipulation. Server-provided prompts are intentionally instruction-bearing. They should therefore be governed as configuration artifacts with source restrictions, versioning, review, integrity monitoring, and controlled change.

Confused deputy and excessive delegation. A correctly authenticated MCP server may operate under credentials broader than the initiating user's authority. Application-level policy must therefore constrain tool operations by principal, resource, and purpose even when OAuth and protocol validation succeed.

Cross-server transitive trust. A host can coordinate several servers. Content supplied by Server A may influence the model to invoke a higher-privilege tool on Server B even though the servers never communicate directly. Connection isolation is therefore not semantic isolation.

Malicious/compromised servers and tool results. A server is both a software and semantic dependency. Tool results from a trusted connector may contain attacker-controlled email, web, repository, or database content. Trust in the connector must not be inherited automatically by the returned data. MCP threat-modeling research documents these host/client/server interactions and tool-poisoning paths [56].

7.4 MCP Assurance Profile

Human approval is useful for consequential operations only when the user can see the security-relevant semantics of the action. It should complement, not replace, least privilege and resource-level authorization. More generally, MCP illustrates why secure agent interoperability requires both protocol correctness and application-level trust/authority enforcement.

8 Defense-in-Depth Architecture and Control-to-Evidence Mapping

8.1 Five Defensive Functions

The defensive objective is not to guarantee that every malicious instruction is recognized. Instead,

$$\text{Model Failure} \not\Rightarrow \text{System Compromise}. \tag{8}$$

We organize controls into five functions: **Prevent**, **Constrain**, **Authorize**, **Observe**, and **Recover**. These functions deliberately combine AI-specific defenses with established security engineering [37, 38]. Secure-by-design research likewise emphasizes structural separation and bounded privilege rather than

Table 5 MCP-specific controls and assurance evidence.

Surface	Control objective	Representative evidence
Server connection	Establish server trust and appropriate authorization	Server identity, allowlist/onboarding record, authorization metadata
Tool metadata	Prevent metadata from granting authority	Raw descriptors, provenance, version/change history, tool-selection trace
Tool invocation	Authorize operation, resource, and arguments independently	Proposed/executed calls, principal, policy decision, approval
Resources/prompts	Preserve provenance and govern instruction-bearing artifacts	Resource URI, source server, access scope, prompt version/approval
Authorization	Bind credentials to intended resource, scope, and principal	Audience/issuer, scopes, lifetime, step-up events
Cross-server use	Prevent unauthorized semantic privilege transfer	Server-specific context trace and cross-server policy decision
Persistent artifacts	Support change detection, revocation, and rollback	Version history, timestamps, diff, rollback evidence

relying only on classifiers or system prompts [19, 20, 22, 21].

Prevent. Preserve provenance and separation between trusted instructions and untrusted data. Useful controls include canonicalization, structured prompt construction, retrieval provenance, governed prompt/workflow artifacts, and trust policy for tool metadata. Prompt filtering and model guardrails remain useful susceptibility-reduction mechanisms, but adaptive-defense studies caution against treating them as complete security boundaries [46, 57, 58].

Constrain. Limit the authority available after a model failure through per-tool identities, read/write separation, short-lived scopes, resource restrictions, segmentation, egress controls, isolated execution, and bounded persistent state. Least privilege is therefore also an AI robustness mechanism: it reduces system impact without requiring the model itself to become perfectly robust.

Authorize. A syntactically valid tool call is not automatically permitted. Consequential actions should pass deterministic or independently enforced policy based on the authenticated principal, target resource, operation, and context. Higher-risk write/execute/administrative actions can require step-up authorization or meaningful human approval.

Observe. Security telemetry should reconstruct the causal chain from initiating input through context construction, retrieval, model proposal, authorization, tool execution, and resulting state. Logs may use identifiers or hashes when storing full sensitive context would create additional risk.

Recover. Revocation and rollback are essential when attacks alter credentials, memory, RAG corpora, prompts, tool configuration, or downstream records. Organizations should be able to disable individual tools/servers, rotate credentials, invalidate derived memory, and restore trusted state without rebuilding the complete application.

8.2 Control Independence

Multiple logical checks are not defense in depth if all depend on the same language-model judgment. A stronger path separates responsibilities:

$$\begin{aligned}\text{LLM Proposal} &\rightarrow \text{Deterministic Policy}\\ &\rightarrow \text{Scoped Credential}\\ &\rightarrow \text{Resource Enforcement.}\end{aligned} \tag{9}$$

The central assurance claim is that failure of the model-level control does not imply failure of the downstream authorization and containment controls.

Common anti-patterns follow directly: treating the system prompt as an authorization boundary; relying on prompt-injection detection alone; assuming schema validation implies permission; using broad shared agent accounts; displaying uninformative approval dialogs; trusting content because it arrived through a trusted connector; logging only final tool calls; and maintaining persistent AI state without rollback.

The framework is aligned, rather than competitive, with NIST control families, the AI RMF/GenAI Profile, OWASP LLM and agentic guidance, and MITRE ATLAS [24, 25, 27, 28, 26]. Those sources identify risks and control objectives; the present framework adds a deployment-oriented link between boundary, verification activity, and evidence.

9 Measurement, Reporting, and Assurance Indicators

9.1 Measurement Philosophy

This paper does not report experimental results. It defines indicators that can be applied when the assurance process is executed against a concrete system. Measurements should be tied to explicit security outcomes, report security and legitimate utility separately, and remain decomposable rather than being collapsed into an

Table 6 Consolidated control-to-evidence mapping.

Function	Control objective	Assurance evidence	Indicative alignment
Prevent	Preserve data/instruction and provenance boundaries	Context/retrieval traces, source labels, governed tool/prompt versions	NIST SI/CM; OWASP
Constrain	Minimize reachable privilege and egress	Permission matrix, token scopes, segmentation/egress policy	NIST AC/SC
Authorize	Bind actions to principal, resource, operation, and purpose	Policy allow/deny logs, principal mapping, approval record	NIST AC/IA
Observe	Attribute actions to their causal context and detect anomalies	Correlated input/context/tool/auth/state events, alerts	NIST AU/SI/IR
Recover	Revoke authority and restore trustworthy state	Revocation records, state versions, rollback verification	NIST CM/IR

unvalidated universal score. This approach is consistent with information-security measurement guidance [59].

The proposed profile is

$$\mathcal{M} = \langle \mathrm{ASR}, \mathrm{BIR}, \mathrm{BTSR}, \mathrm{BRC}, \mathrm{TTD}, \mathrm{UDR} \rangle. \tag{10}$$

9.2 Attack Success, Boundary Interception, and Benign Utility

For scenario family g,

$$\mathrm{ASR}_g = \frac{\sum_{i=1}^{N_g} \mathbb{I}[O_i = \text{success}]}{N_g}. \tag{11}$$

Success should mean that the prohibited system outcome occurred, not merely that the model appeared to follow the attack. ASR should be stratified by outcome, surface, data sensitivity, and privilege rather than reported as a single number. When repeated stochastic trials are used, the number of trials and uncertainty method should be stated; Wilson intervals are a defensible choice for binary proportions [60].

Boundary Interception Rate diagnoses where attacks are contained:

$$\mathrm{BIR}_b = \frac{N_{\text{blocked},b}}{N_{\text{presented},b}}. \tag{12}$$

BIR is conditional on an adversarial transition actually reaching boundary b. Reports should therefore state $N_{\text{presented},b}$ and any upstream attrition rather than comparing BIR values across systems as if exposure were identical. It is a control-specific diagnostic rather than an overall security score. Benign-Task Success Rate (BTSR) measures legitimate-task preservation:

$$\mathrm{BTSR} = \frac{\sum_{j=1}^{|B|} \mathbb{I}[\text{task_success}(j)]}{|B|}. \tag{13}$$

AgentDojo and recent defense evaluations illustrate why security and utility must be evaluated together [12, 57]. A defense that blocks all actions may reduce ASR while destroying the intended application.

9.3 Blast-Radius Characterization

Rather than assigning arbitrary universal weights, we represent the consequence of a successful path as

$$\mathrm{BRC}_i = \langle D_i, O_i, X_i, E_i, P_i \rangle, \tag{14}$$

where D_i is data sensitivity/breadth, O_i the strongest operation class (read, write, execute, administrative), X_i cross-system reach, E_i external-egress capability, and P_i persistence. The profile preserves distinctions that a single weighted score would obscure. Recent work similarly argues that binary attack success alone can miss important differences in action severity [61].

9.4 Detection and Coverage

For an observable event,

$$\mathrm{TTD}_i = t_{\text{detect},i} - t_{\text{event},i}. \tag{15}$$

TTD should be paired with an Undetected Event Rate,

$$\mathrm{UDR} = \frac{N_{\text{undetected}}}{N_{\text{security-relevant test events}}}, \tag{16}$$

where the denominator is established by the authorized test harness or other ground truth, not by what the production detector happened to observe. Excluding never-detected cases can make detection performance appear artificially strong. Precision/recall remain useful for a particular guardrail or detector but should not be interpreted as end-to-end application security.

A low measured ASR is also weak evidence if scenario coverage is narrow. Reports should therefore show which

combinations of architectural surface and security outcome were tested. Benchmark analyses have shown that results can change materially with attack strength, implementation, and metric design [58]. The measurement context should record model, prompt/workflow, orchestration, tool metadata, retrieval/state, authorization, guardrail, and scenario-suite versions so that longitudinal comparisons are interpretable.

The framework deliberately omits a composite Resilience Index. Combining heterogeneous measures such as ASR, latency, utility, and impact would require unvalidated normalization and weighting assumptions. Organizations may map the decomposed profile into their own established risk method, but that mapping is deployment-specific.

10 Related Work and Comparative Positioning

10.1 Prompt Injection, Agent Benchmarks, and Defensive Design

Prompt-injection research established direct and indirect instruction attacks, model-level success criteria, and practical attack corpora [5,8,9,10,29,7]. Jailbreak research and automated red teaming broadened understanding of model-safety failure modes [6,47,49]. Defensive work has since moved toward stronger separation and system design, including spotlighting, structured query interfaces, capability-oriented execution, privilege-flow tracking, and agent-specific secure design patterns [46,19,20,21,22].

Tool-integrated benchmarks answer a complementary empirical question: how often do agents fail under specified attacks and tasks? InjecAgent focuses on indirect injection in tool-integrated agents; AgentDojo jointly evaluates security and utility; ToolEmu studies tool-use risks in a sandbox; Agent Security Bench broadens coverage across agent stages and defenses [11, 12,13,14]. AgentPoison and PoisonedRAG extend the threat to memory and knowledge-base poisoning [42,41]. These works provide attacks, benchmarks, and empirical evidence that can populate Stages 3–4 of the proposed method, but they do not by themselves specify how an enterprise deployment should map each failure to a trust boundary, control, and assurance artifact.

10.2 System-Level Agent Security and Standards

Recent formalizations and surveys increasingly organize agent security around information flow, delegated authority, persistent state, and layered attack surfaces [16, 15,17,18]. Broader surveys cover LLM security and privacy across attack classes and lifecycle concerns [34,35]. The present paper therefore does not claim to be the first system-oriented view of agent security.

A particularly close assurance-oriented contribution is the quantitative framework of Karlsen et al. [23], which derives measurable assurance levels from LLM security requirements and vulnerability metrics and demonstrates the method through a case study. The distinction is substantive: that work emphasizes quantitative assurance scoring for LLM vulnerabilities, whereas the present framework treats the deployed agentic application as the unit of analysis and keeps assurance decomposed across trust boundaries, delegated authority, independent controls, verification evidence, containment, utility, and detection. This paper therefore complements rather than duplicates metric-oriented LLM assurance.

Instead, its contribution is the operational assurance bridge: threat mechanism → architectural surface → trust boundary → delegated authority → independent control → verification evidence. NIST AI and security publications provide governance, testing, control, zero-trust, and measurement foundations [24,25,30,37,31, 38,59]; OWASP and MITRE provide practitioner risk and adversary knowledge [27,28,26]. The framework uses these resources as inputs rather than competing taxonomies.

10.3 MCP and Agent Interoperability

MCP research has rapidly exposed semantic risks not captured by transport or OAuth correctness. Tool metadata poisoning, implicit cross-tool steering, and host/client/server threat modeling are documented by MCPTox, MCP-ITP, and published MCP threat-modeling work [53,54,56]. The current MCP specification and authorization profile provide the protocol foundation [33,40,52,55]. The present work's three-layer distinction—protocol, semantic, and enterprise security—positions these attacks within a broader assurance model.

The novelty claim is intentionally bounded. We do not claim the first taxonomy, system-oriented agent model, or empirical benchmark, nor do we claim superior defense effectiveness. The contribution is a reusable application-security assessment method that converts rapidly evolving threat knowledge into verifiable control claims.

Table 7 Recommended assurance indicators.

Indicator	Purpose	Interpretation constraint
ASR	Frequency of defined prohibited outcomes	Stratify by outcome/surface; model obedience is not system compromise
BIR	Conditional boundary-level containment effectiveness	Report exposure count and upstream attrition; not overall security
BTSR	Legitimate-task preservation	Interpret with ASR; blocking all functionality is not useful security
BRC	Multidimensional impact/containment	Do not apply universal arbitrary weights
TTD	Detection latency	Report distribution and pair with UDR
UDR	Proportion of ground-truth test events not detected	State observation window and telemetry coverage

Table 8 Comparative positioning of representative approaches.

Work / family	Primary purpose	Main strength	Relationship to this work
Prompt-injection studies [8, 29]	Attack/defense analysis	Mechanisms and success criteria	Supply scenario mechanisms
Agent benchmarks [11, 12, 14]	Empirical evaluation	Realistic tasks, tools, and attack outcomes	Supply test environments and cases
System-level surveys/formalisms [16, 15, 18]	Threat/system synthesis	Trust, authority, state, lifecycle concepts	Closest conceptual foundation
Karlsen et al. [23]	Quantitative LLM assurance	Requirements-to-score assurance levels with case study	Closest assurance-oriented predecessor; differs in unit of analysis and evidence model
OWASP/MITRE	Risk and adversary guidance	Practitioner vocabulary and coverage	Supply threat knowledge and control objectives
NIST	Governance, controls, testing	Mature assurance and assessment practice	Supplies control/evidence discipline
MCP security studies	Protocol/tool ecosystem threats	Concrete metadata/server attack paths	Specialized assurance case
This work	**System-level security assurance**	**Traceability from threat to boundary, control, verification, and evidence**	**Operational synthesis**

11 Discussion and Design Implications

The three research questions can now be answered directly. **RQ1** is addressed by the mediated-authority model, six security surfaces, seven trust boundaries, and the attack-path tuple, which separate semantic influence from delegated privilege and protected outcomes. **RQ2** is addressed by the six-stage assurance process and control-to-evidence mapping, which translate known threats into deployment-specific scenarios, independent control checks, and auditable findings. **RQ3** is addressed by the decomposed measurement profile: ASR characterizes prohibited outcomes, BIR characterizes conditional boundary interception, BTSR captures legitimate utility, BRC preserves multidimensional impact, and TTD/UDR characterize detection without collapsing these properties into a universal score.

Three broader implications follow from the framework. First, **model robustness and application security are distinct**. The same model can have different security postures depending on tool scope, identity, memory, and downstream controls. Benchmarks therefore provide important but incomplete application assurance.

Second, **trust should be represented as authority within context**. A document may be trustworthy as factual input without being authorized to command a tool; a legitimate connector may return malicious content; and an authenticated MCP server may expose data that should remain semantically untrusted. This distinction explains why provenance matters beyond logging.

Third, **least privilege and independent authorization are AI robustness mechanisms**. They reduce the consequence of stochastic model behavior without requiring the model to become perfectly resistant. A useful architecture separates interpretation, authorization, and execution:

$$\begin{aligned}\text{LLM Interpret/Propose} &\rightarrow \text{Policy Authorize}\\ &\rightarrow \text{Scoped Capability}\\ &\rightarrow \text{Execute.}\end{aligned} \tag{17}$$

Agent security is therefore compositional. Individual components may operate according to their local specifications while the overall system violates policy because information from one trust domain acquires authority in another. Persistent memory extends this problem over time: insertion, dormancy, and later activation may span different sessions or users. Consequently, AI assurance should be change-aware, with reassessment triggered by material changes to models, prompts, tools, retrieval sources, memory behavior, credentials, or policy.

For practitioners, this means beginning with architecture and authority rather than a prompt library. For designers, it favors source provenance, narrow privileges, external authorization, reversible state, and causal telemetry. For researchers, it motivates evaluation of boundary failures, delegated authority, persistent-state attacks, and security–utility trade-offs alongside final ASR.

12 Limitations and Future Research

The principal limitation is that the framework is conceptual and has not yet been empirically validated as an assessment methodology. It therefore does not claim to reduce ASR by a particular amount or outperform alternative approaches. Source selection was purposive rather than systematic, and the conceptual coding and consolidation were performed by a single author; consequently, coverage bias and category-design bias remain possible, and no inter-rater reliability is claimed. The six surfaces are also assessment-oriented rather than exhaustive; future multi-agent, multimodal, embodied, or self-modifying systems may require additional boundaries.

Results produced with the framework will depend on scenario quality, instrumentation, and configuration stability. Weak attacks can understate susceptibility, benign suites can overstate utility, undocumented privileges can understate blast radius, and incomplete telemetry can bias BIR, TTD, or UDR. LLM stochasticity further requires repeated trials where quantitative claims are made. Human approval is also difficult to evaluate because its effectiveness depends on interface design, user understanding, and approval fatigue.

Future work should apply the framework across representative RAG, productivity, code, MCP, and memory-enabled agents; compare it with prompt-focused testing; evaluate whether boundary mapping improves discovery and remediation; and investigate machine-readable provenance, multi-agent delegation, causal attribution, and standardized security telemetry. Such validation would allow the framework's categories and indicators to be refined rather than treated as fixed.

13 Conclusion

LLM applications increasingly retrieve, remember, invoke tools, and act through delegated enterprise authority. Prompt injection remains an important attack mechanism, but its security significance depends on whether adversarial influence crosses a trust boundary and reaches protected data, privileged actions, or persistent state.

This paper reframes that problem as system-level security assurance. It contributes an architecture-oriented threat model, a six-stage assessment process, an MCP-specific assurance profile, a defense-in-depth control-to-evidence mapping, and a decomposed measurement profile. A worked enterprise RAG–MCP case illustrates how these elements can be instantiated from attack path to assurance evidence without being presented as empirical validation. The central design objective is simple:

$$\begin{aligned}\text{Threat} &\rightarrow \text{Surface} \rightarrow \text{Trust Boundary} \\ &\rightarrow \text{Authority} \rightarrow \text{Control} \\ &\rightarrow \text{Verification} \rightarrow \text{Evidence}\end{aligned} \tag{18}$$

The framework does not assume that every malicious instruction can be eliminated. Instead, it asks whether untrusted information can silently acquire unrestricted authority and whether independent controls can demonstrate containment when model-level defenses fail. This makes agent security a tractable extension of familiar application-security principles: least privilege, explicit authorization, provenance, auditability, segmentation, and recovery.

Statements and Declarations

Data availability. No new datasets were generated or analysed in this study. The framework is developed from the publicly available research literature, standards, specifications, and practitioner guidance cited in the manuscript.

Funding. The author received no specific funding for this work.

Competing interests. The author declares no competing interests.

References

1. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.t., Rocktäschel, T., Riedel, S., Kiela, D.: Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In: *Advances in Neural Information Processing Systems*, vol. 33 (2020), pp. 9459–9474
2. Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K.R., Cao, Y.: ReAct: Synergizing Reasoning and Acting in Language Models. In: *International Conference on Learning Representations* (2023). URL `https://openreview.net/forum?id=WE_vluYUL-X`

3. Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., Scialom, T.: Toolformer: Language Models Can Teach Themselves to Use Tools. In: *Advances in Neural Information Processing Systems*, vol. 36 (2023), pp. 68,539–68,551. https://doi.org/10.52202/075280-2997
4. Liu, X., Yu, H., Zhang, H., Xu, Y., Lei, X., Lai, H., Gu, Y., Ding, H., Men, K., Yang, K., Zhang, S., Deng, X., Zeng, A., Du, Z., Zhang, C., Shen, S., Zhang, T., Su, Y., Sun, H., Huang, M., Dong, Y., Tang, J.: AgentBench: Evaluating LLMs as Agents. In: *International Conference on Learning Representations* (2024). URL https://openreview.net/forum?id=zAdUB0aCTQ
5. Perez, F., Ribeiro, I.: Ignore Previous Prompt: Attack Techniques For Language Models. arXiv preprint arXiv:2211.09527 (2022). URL https://arxiv.org/abs/2211.09527
6. Wei, A., Haghtalab, N., Steinhardt, J.: Jailbroken: How Does LLM Safety Training Fail? In: *Advances in Neural Information Processing Systems*, vol. 36 (2023). https://doi.org/10.52202/075280-3508
7. Toyer, S., Watkins, O., Mendes, E.A., Svegliato, J., Bailey, L., Wang, T., Ong, I., Elmaaroufi, K., Abbeel, P., Darrell, T., Ritter, A., Russell, S.: Tensor Trust: Interpretable Prompt Injection Attacks from an Online Game. In: *International Conference on Learning Representations* (2024). URL https://openreview.net/forum?id=fsW7wJGLBd
8. Greshake, K., Abdelnabi, S., Mishra, S., Endres, C., Holz, T., Fritz, M.: Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. In: *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security* (Association for Computing Machinery, 2023), pp. 79–90. https://doi.org/10.1145/3605764.3623985
9. Liu, Y., Deng, G., Li, Y., Wang, K., Wang, Z., Wang, X., Zhang, T., Liu, Y., Wang, H., Zheng, Y., Liu, Y.: Prompt Injection Attack Against LLM-Integrated Applications. arXiv preprint arXiv:2306.05499 (2023). URL https://arxiv.org/abs/2306.05499
10. Yi, J., Xie, Y., Zhu, B., Kiciman, E., Sun, G., Xie, X., Wu, F.: Benchmarking and Defending Against Indirect Prompt Injection Attacks on Large Language Models. In: *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '25)*, vol. 1 (Association for Computing Machinery, 2025), pp. 1809–1820. https://doi.org/10.1145/3690624.3709179
11. Zhan, Q., Liang, Z., Ying, Z., Kang, D.: InjecAgent: Benchmarking Indirect Prompt Injections in Tool-Integrated Large Language Model Agents. In: *Findings of the Association for Computational Linguistics: ACL 2024* (Association for Computational Linguistics, Bangkok, Thailand, 2024), pp. 10,471–10,506. https://doi.org/10.18653/v1/2024.findings-acl.624
12. Debenedetti, E., Zhang, J., Balunović, M., Beurer-Kellner, L., Fischer, M., Tramèr, F.: AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents. In: *Advances in Neural Information Processing Systems*, vol. 37 (Neural Information Processing Systems Foundation, 2024), pp. 82,895–82,920. https://doi.org/10.52202/079017-2636
13. Ruan, Y., Dong, H., Wang, A., Pitis, S., Zhou, Y., Ba, J., Dubois, Y., Maddison, C.J., Hashimoto, T.: Identifying the Risks of LM Agents with an LM-Emulated Sandbox. In: *International Conference on Learning Representations* (2024). URL https://openreview.net/forum?id=m1YYAQjO3w
14. Zhang, H., Huang, J., Mei, K., Yao, Y., Wang, Z., Zhan, C., Wang, H., Zhang, Y.: Agent Security Bench (ASB): Formalizing and Benchmarking Attacks and Defenses in LLM-Based Agents. In: *International Conference on Learning Representations* (2025)
15. Ling, Y., Yu, S., Chen, Z., Fang, C.: Toward Secure LLM Agents: Threat Surfaces, Attacks, Defenses, and Evaluation. arXiv preprint arXiv:2606.10749 (2026). URL https://arxiv.org/abs/2606.10749
16. Siu, V., He, J., Montgomery, K., Wang, Z., Gong, N., Wang, C., Song, D.: A Framework for Formalizing LLM Agent Security. arXiv preprint arXiv:2603.19469 (2026). URL https://arxiv.org/abs/2603.19469
17. Chu, K.: A Systematic Survey of Security Threats and Defenses in LLM-Based AI Agents: A Layered Attack Surface Framework. arXiv preprint arXiv:2604.23338 (2026). URL https://arxiv.org/abs/2604.23338
18. Kim, J., Guo, W., Song, D.: SoK: Attack and Defense Landscape of Agentic AI Systems. In: *35th USENIX Security Symposium (USENIX Security 26)* (USENIX Association, Baltimore, MD, 2026). URL https://www.usenix.org/conference/usenixsecurity26/presentation/kim-juhee-agentic
19. Chen, S., Piet, J., Sitawarin, C., Wagner, D.: StruQ: Defending Against Prompt Injection with Structured Queries. In: *34th USENIX Security Symposium (USENIX Security 25)* (USENIX Association, Seattle, WA, 2025), pp. 2383–2400. URL https://www.usenix.org/conference/usenixsecurity25/presentation/chen-sizhe
20. Debenedetti, E., Shumailov, I., Fan, T., Hayes, J., Carlini, N., Fabian, D., Kern, C., Shi, C., Terzis, A., Tramèr, F.: Defeating Prompt Injections by Design. In: *IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)* (2026). URL https://arxiv.org/abs/2503.18813
21. Kim, J., Choi, W., Lee, B.: Prompt Flow Integrity to Prevent Privilege Escalation in LLM Agents. In: *35th USENIX Security Symposium (USENIX Security 26)* (USENIX Association, 2026). URL https://arxiv.org/abs/2503.15547
22. Beurer-Kellner, L., Buesser, B., Creţu, A.M., Debenedetti, E., Dobos, D., Fabian, D., Fischer, M., Froelicher, D., Grosse, K., Naeff, D., Ozoani, E., Paverd, A., Tramèr, F., Volhejn, V.: Design Patterns for Securing LLM Agents against Prompt Injections. arXiv preprint arXiv:2506.08837 (2025). URL https://arxiv.org/abs/2506.08837
23. Karlsen, S.S., Yamin, M.M., Hashmi, E., Katt, B., Ullah, M.: Securing Large Language Models: A Quantitative Assurance Framework Approach. Journal of Information Security and Applications **97**, 104351 (2026). https://doi.org/10.1016/j.jisa.2025.104351
24. Tabassi, E.: Artificial Intelligence Risk Management Framework (AI RMF 1.0). Tech. Rep. NIST AI 100-1, National Institute of Standards and Technology (2023). https://doi.org/10.6028/NIST.AI.100-1
25. Autio, C., Schwartz, R., Dunietz, J., Jain, S., Stanley, M., Tabassi, E., Hall, P., Roberts, K.: Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile. Tech. Rep. NIST AI 600-1, National Institute of Standards and Technology (2024). https://doi.org/10.6028/NIST.AI.600-1
26. MITRE: MITRE ATLAS: Adversarial Threat Landscape for Artificial-Intelligence Systems. https://atlas.mitre.org/ (2026). Living knowledge base, accessed August 2026

27. OWASP GenAI Security Project: OWASP GenAI LLM Top 10 2026. https://genai.owasp.org/resource/owasp-genai-llm-top-10-2026/ (2026). Accessed August 2026
28. OWASP GenAI Security Project: OWASP Top 10 for Agentic Applications for 2026. https://genai.owasp.org/resource/owasp-top-10-for-agentic-applications-for-2026/ (2025). Accessed August 2026
29. Liu, Y., Jia, Y., Geng, R., Jia, J., Gong, N.Z.: Formalizing and Benchmarking Prompt Injection Attacks and Defenses. In: *33rd USENIX Security Symposium (USENIX Security 24)* (USENIX Association, Philadelphia, PA, 2024), pp. 1831–1847. URL https://www.usenix.org/conference/usenixsecurity24/presentation/liu-yupei
30. Scarfone, K., Souppaya, M., Cody, A., Orebaugh, A.: Technical Guide to Information Security Testing and Assessment. Tech. Rep. NIST Special Publication 800-115, National Institute of Standards and Technology (2008). https://doi.org/10.6028/NIST.SP.800-115
31. Joint Task Force: Assessing Security and Privacy Controls in Information Systems and Organizations. Tech. Rep. NIST Special Publication 800-53A Revision 5, National Institute of Standards and Technology (2022). https://doi.org/10.6028/NIST.SP.800-53Ar5
32. OWASP GenAI Security Project: GenAI Red Teaming Guide. https://genai.owasp.org/resource/genai-red-teaming-guide/ (2025). Accessed August 2026
33. Model Context Protocol: Model Context Protocol Specification, Revision 2026-07-28. https://modelcontextprotocol.io/specification/2026-07-28 (2026). Accessed August 2026
34. Yao, Y., Duan, J., Xu, K., Cai, Y., Sun, Z., Zhang, Y.: A Survey on Large Language Model (LLM) Security and Privacy: The Good, The Bad, and The Ugly. High-Confidence Computing **4**(2), 100211 (2024). https://doi.org/10.1016/j.hcc.2024.100211
35. Das, B.C., Amini, M.H., Wu, Y.: Security and Privacy Challenges of Large Language Models: A Survey. ACM Computing Surveys **57**(6), 1 (2025). https://doi.org/10.1145/3712001
36. Souppaya, M., Scarfone, K.: Guide to Data-Centric System Threat Modeling. Tech. Rep. NIST Special Publication 800-154, Initial Public Draft, National Institute of Standards and Technology (2016). URL https://csrc.nist.gov/pubs/sp/800/154/ipd. Initial Public Draft
37. Joint Task Force: Security and Privacy Controls for Information Systems and Organizations. Tech. Rep. NIST Special Publication 800-53 Revision 5, National Institute of Standards and Technology (2020). https://doi.org/10.6028/NIST.SP.800-53r5
38. Rose, S., Borchert, O., Mitchell, S., Connelly, S.: Zero Trust Architecture. Tech. Rep. NIST Special Publication 800-207, National Institute of Standards and Technology (2020). https://doi.org/10.6028/NIST.SP.800-207
39. Vassilev, A., Oprea, A., Fordyce, A., Anderson, H., Davies, X., Hamin, M.: Adversarial Machine Learning: A Taxonomy and Terminology of Attacks and Mitigations. Tech. Rep. NIST AI 100-2 E2025, National Institute of Standards and Technology (2025). URL https://csrc.nist.gov/pubs/ai/100/2/e2025/final
40. Model Context Protocol: Model Context Protocol Specification: Authorization, Revision 2026-07-28. https://modelcontextprotocol.io/specification/2026-07-28/basic/authorization (2026). Accessed August 2026
41. Zou, W., Geng, R., Wang, B., Jia, J.: PoisonedRAG: Knowledge Corruption Attacks to Retrieval-Augmented Generation of Large Language Models. In: *34th USENIX Security Symposium (USENIX Security 25)* (USENIX Association, Seattle, WA, 2025), pp. 3827–3844. URL https://www.usenix.org/conference/usenixsecurity25/presentation/zou-poisonedrag
42. Chen, Z., Xiang, Z., Xiao, C., Song, D., Li, B.: AgentPoison: Red-Teaming LLM Agents via Poisoning Memory or Knowledge Bases. In: *Advances in Neural Information Processing Systems*, vol. 37 (2024), pp. 130,185–130,213. https://doi.org/10.52202/079017-4136
43. Pulipaka, S., Hlebik, S., Raghav, L., Abdelnabi, S., Raina, V., Sheth, I., Fritz, M.: Hidden in Memory: Sleeper Memory Poisoning in LLM Agents. arXiv preprint arXiv:2605.15338 (2026). URL https://arxiv.org/abs/2605.15338
44. OWASP Foundation: OWASP Web Security Testing Guide. https://owasp.org/www-project-web-security-testing-guide/ (2025). Living security-testing guide, accessed August 2026
45. Bagdasarian, E., Yi, R., Ghalebikesabi, S., Kairouz, P., Gruteser, M., Oh, S., Balle, B., Ramage, D.: AirGapAgent: Protecting Privacy-Conscious Conversational Agents. In: *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security* (Association for Computing Machinery, New York, NY, USA, 2024), pp. 3868–3882. https://doi.org/10.1145/3658644.3690350
46. Hines, K., Lopez, G., Hall, M., Zarfati, F., Zunger, Y., Kiciman, E.: Defending Against Indirect Prompt Injection Attacks With Spotlighting. arXiv preprint arXiv:2403.14720 (2024). URL https://arxiv.org/abs/2403.14720
47. Perez, E., Huang, S., Song, F., Cai, T., Ring, R., Aslanides, J., Glaese, A., McAleese, N., Irving, G.: Red Teaming Language Models with Language Models. In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing* (Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, 2022), pp. 3419–3448. https://doi.org/10.18653/v1/2022.emnlp-main.225
48. Amironesei, R., Godil, A., Greenberg, C., Greene, K., Hall, J.P., Jensen, T., Fiscus, J., Schulman, N.: Assessing Risks and Impacts of AI (ARIA): Pilot Evaluation Report. Tech. Rep. NIST AI 700-2, National Institute of Standards and Technology (2025). https://doi.org/10.6028/NIST.AI.700-2
49. Chao, P., Debenedetti, E., Robey, A., Andriushchenko, M., Croce, F., Sehwag, V., Dobriban, E., Flammarion, N., Pappas, G.J., Tramèr, F., Hassani, H., Wong, E.: JailbreakBench: An Open Robustness Benchmark for Jailbreaking Large Language Models. In: *Advances in Neural Information Processing Systems*, vol. 37 (2024). https://doi.org/10.52202/079017-1745
50. Model Context Protocol: Model Context Protocol Specification: Architecture, Revision 2026-07-28. https://modelcontextprotocol.io/specification/2026-07-28/architecture (2026). Accessed August 2026
51. Soria Parra, D., Delimarsky, D.: The 2026-07-28 Model Context Protocol Specification. https://blog.modelcontextprotocol.io/posts/2026-07-28/ (2026). Model Context Protocol Core Maintainers, accessed August 2026
52. Model Context Protocol: Model Context Protocol Specification: Tools, Revision 2026-07-28.

https://modelcontextprotocol.io/specification/2026-07-28/server/tools (2026). Accessed August 2026
53. Wang, Z., Gao, Y., Wang, Y., Liu, S., Sun, H., Cheng, H., Shi, G., Du, H., Li, X.: MCPTox: A Benchmark for Tool Poisoning Attack on Real-World MCP Servers. arXiv preprint arXiv:2508.14925 (2025). URL https://arxiv.org/abs/2508.14925
54. Li, R., Wang, Z., Yao, Y., Li, X.Y.: MCP-ITP: An Automated Framework for Implicit Tool Poisoning in MCP. arXiv preprint arXiv:2601.07395 (2026). URL https://arxiv.org/abs/2601.07395
55. Model Context Protocol: Model Context Protocol Specification: Resources, Revision 2026-07-28. https://modelcontextprotocol.io/specification/2026-07-28/server/resources (2026). Accessed August 2026
56. Huang, C., Huang, X., Tran, N.P., Milani Fard, A.: Model Context Protocol Threat Modeling and Analysis of Vulnerabilities to Prompt Injection with Tool Poisoning. Journal of Cybersecurity and Privacy **6**(3), 84 (2026). https://doi.org/10.3390/jcp6030084
57. Jia, Y., Shao, Z., Liu, Y., Jia, J., Song, D., Gong, N.Z.: A Critical Evaluation of Defenses against Prompt Injection Attacks. arXiv preprint arXiv:2505.18333 (2025). URL https://arxiv.org/abs/2505.18333
58. Bhagwatkar, R., Kasa, K., Puri, A., Huang, G., Rish, I., Taylor, G.W., Dvijotham, K.D., Lacoste, A.: Indirect Prompt Injections: Are Firewalls All You Need, or Stronger Benchmarks? arXiv preprint arXiv:2510.05244 (2025). URL https://arxiv.org/abs/2510.05244
59. Schroeder, K., Trinh, H., Pillitteri, V.: Measurement Guide for Information Security Volume 1: Identifying and Selecting Measures. Tech. Rep. NIST Special Publication 800-55 Volume 1, National Institute of Standards and Technology (2024). https://doi.org/10.6028/NIST.SP.800-55v1
60. Brown, L.D., Cai, T.T., DasGupta, A.: Interval Estimation for a Binomial Proportion. Statistical Science **16**(2), 101 (2001). https://doi.org/10.1214/ss/1009213286
61. Owiredu-Ashley, H.: Beyond Attack-Success Rate: Action-Graded Severity Scale for Tool-Using AI Agents. arXiv preprint arXiv:2607.07474 (2026). URL https://arxiv.org/abs/2607.07474