IJSSP
المجلة الدولية لنشر الدراسات العلمية
المجلد الواحد والثلاثون – العدد الأول – البحث الرابع (ص: 90 - 106)
15 – تشرين الأول – 2026م
www.ijssp.com ISSN: 2707-5184 International Journal of Scientific Studies Publishing

[Security and Privacy of AI Agent Systems: A Systematic Evidence Map, Taxonomy, and Future Research Agenda]

(Evidence-focused review of tool-using, memory-enabled, and multi-agent artificial intelligence systems)

Researchers:
[Hamad Hamed Mohamed Almalki]
[Jeddah Technical College – Computer Science Department - Jeddah, Kingdom of Saudi Arabia]
[Jamaan bin Sultan Jamaan Al-Nathli Al-Malki]
[Bachelor's Student in Computer Science]

Abstract:
Artificial intelligence (AI) agents combine probabilistic planning with delegated tool authority, persistent state, and communication across trust boundaries. Consequently, failures may produce unauthorized effects and contextual disclosure rather than merely unsafe text. We conducted a bounded, source-verified systematic evidence mapping of 34 scholarly and normative records identified through public scholarly search, publisher proceedings, and standards repositories up to 16 October 2026. The selected evidence comprises 21 primary empirical or technical studies, five related reviews, six technical standards or guidance documents, and two conceptual or methodological sources. Search scope, eligibility rules, source status, and reporting limitations are specified; exhaustive database coverage and a conventional PRISMA screening flow are not asserted. We derive a five-axis taxonomy of entry surfaces, propagation routes, targeted assets, outcomes, and temporal persistence. Across the included work, four interacting failure classes recur: indirect prompt injection, over-broad agent/tool authority, memory extraction or poisoning, and contextual privacy violations in inter-agent exchanges. Comparative analysis of InjecAgent, AgentDojo, PrivacyLens, Agent Security Bench, WASP, and ConVerse show why rates of instruction capture, attempted tool misuse, and verified unauthorized effects must not be treated as equivalent or pooled across incompatible settings. We propose a layered enforcement architecture and a research agenda for task-scoped delegation, privacy-aware information flow, state provenance, adaptive outcome-based evaluation, and auditable oversight. The principal contribution is a unified security–privacy analysis at the level of completed agent workflows, together with explicit evidence-quality boundaries.
Keywords: AI agents; Agentic systems; Prompt injection; Privacy; Autonomous tools; Memory poisoning; Authorization; Security benchmarks; Systematic evidence synthesis.

1. Introduction

Large language models have become components of operational systems that retrieve documents, browse interfaces, invoke APIs, write code, update records, and coordinate multi-step tasks. Such capabilities are usually presented as increased autonomy, yet autonomy is also a change in the placement of trust. A chatbot can generate an incorrect recommendation; a tool-using agent may transform the same error into a database mutation, financial transfer, disclosure, or message to a third party. The security boundary has consequently moved from a single model response to a sequence of decisions distributed across software services [3]–[5], [24].

Unlike a conventional deterministic application, an agent frequently interprets data and instructions in a shared linguistic representation. Attacker-controlled web pages, search snippets, repository files, and tool responses can therefore be mistaken for higher-priority requests. Indirect prompt injection has been demonstrated against integrated applications and operational agents, and it can alter both language outputs and downstream tool use [1], [2], [4], [5]. The existence of a tool interface matters because a successful redirection can create a real-world effect, not merely an unwanted sentence.

Privacy introduces a separate but interacting concern. Personal assistants may see information about users, colleagues, customers, and bystanders, even when the immediate task concerns only a subset of those people. Memory stores can outlive the activity that justified collection. A benign request may retrieve private past interactions, while an apparently legitimate agent-to-agent negotiation may disclose details beyond the user's authorization [6], [11], [15], [31]. Privacy failures thus cannot be reduced to deliberate malicious prompts or to the presence of personally identifiable information alone.

This review asks four research questions: RQ1, which architectural boundaries produce agent-specific security and privacy risks? RQ2, how should existing attacks, assets, failure outcomes, and time horizons be classified without conflating them? RQ3, which control families have empirical support, and how well do existing benchmarks measure meaningful harm? RQ4, what near- and longer-term research designs would produce more generalizable assurance?

Our intended contributions are an integrated taxonomy, a structured synthesis with source-status distinctions, a defense and evaluation framework, and a research agenda with falsifiable hypotheses. We make no claim that this is a meta-analysis: agent tasks, attacker access, base models, defense configurations, and outcome definitions are too heterogeneous to justify a common pooled effect. Equally, we distinguish demonstrated exploitation within benchmark conditions from estimates of real-world incident prevalence.

2. Scope and Conceptual Foundations

2.1. Defining an AI agent

For this review, an AI agent is a software system in which a learned model participates in selecting or sequencing actions toward a delegated goal, using some combination of tool interfaces, environmental observations, stored state, and human or machine communication. This operational definition deliberately includes single-agent applications and multi-agent systems, but excludes ordinary prompt-only chat interactions unless the study directly tests an agent boundary. It also distinguishes the model, the agent orchestration layer, and the external execution environment; weaknesses in one need not imply equivalent weaknesses in the others [3], [8], [29].

Five operational features expand risk. Delegation assigns authority to an uncertain planner. Tool use couples model outputs to privileged interfaces. Retrieval imports material that was not authored by the user or developer. Persistent memory transforms one compromised step into possible cross-session influence. Multi-agent communication creates channels in which another agent's assertions can masquerade as trustworthy context. A complete threat model must track these features as independently testable mechanisms rather than label every failure as hallucination.

2.2. Security, safety, and privacy are related but not interchangeable

Security concerns adversarial influence or unauthorized access, modification, or disruption. Safety concerns harmful outcomes, including harmful but non-adversarial actions. Privacy concerns the legitimacy of collecting, retaining, inferring, or communicating information within a context. For example, disclosing an address to a properly authenticated vendor can still violate privacy expectations when the vendor had no task-related need for it. Contextual integrity offers a useful account of why recipient, purpose, data type, and transmission principle jointly matter [31].

The distinction matters for measurement. Refusal of a harmful request evaluates one safety boundary; resistance to hostile content embedded in a tool response evaluates another; prevention of unauthorized downstream disclosure measures an additional property. AgentHarm explicitly examines the harmful capability of tool-using agents, whereas PrivacyLens examines privacy-norm violations during action and ConVerse evaluates contextual sharing in multi-agent conversations [6], [7], [15]. None is a substitute for the others.

2.3. Position relative to existing surveys

Related reviews are essential context rather than evidence of an unstudied field. Chu's layered security analysis emphasizes architectural layers and time horizons [22], Xu et al. discuss agent self-security alongside agent-enabled cybersecurity [23], and Dantas et al. synthesize specification, verification, and enforcement [32]. Gan et al. examine cross-module and software-lifecycle security and privacy threats [33], while Sammartino systematizes entry points, propagation, target assets, and defenses for autonomous and multi-agent execution [34]. These works overlap substantially with the present domain. Our narrower methodological contribution is the joint classification of security and contextual privacy outcomes, explicit temporal state and recipient/authorization boundaries, and an outcome hierarchy that separates following malicious text from completing an unauthorized action. This is an analytical synthesis, not a claim of the first agent-security taxonomy.

3. Review Methodology

3.1. Search strategy and eligibility

Searches and source verification were conducted through 16 October 2026 using openly accessible scholarly web discovery, publisher proceedings, archival preprint pages, and primary technical standards. Search expressions combined ("LLM agent" OR "AI agent" OR "agentic AI") with ("security" OR "privacy" OR "prompt injection" OR "memory poisoning" OR "tool poisoning" OR "authorization" OR "benchmark"), supplemented with focused queries for multi-agent disclosure, agent memory extraction, and named studies. Authoritative identity and publication status were reconciled, where available, against ACM records, ACL Anthology, NeurIPS/ICLR proceedings, arXiv, NIST, OWASP, and the MCP specification. Search terms and verification surfaces appear in Appendix A. Neither licensed Scopus/Web of Science result exports nor an exhaustive search-result audit were available; therefore, the review is a bounded systematic evidence map, not an exhaustive census or a PRISMA-complete systematic review [30].

An empirical item was eligible for primary synthesis when it (i) evaluated a language-model agent with real or emulated tool execution, persistent memory, or agent–agent interaction; (ii) specified an attacker capability, privacy challenge, or security-relevant test; and (iii) stated a verifiable outcome or evaluation design. Conceptual surveys were used to situate novelty, and technical standards were used as normative—not experimental—sources. Exclusions comprised prompt-only jailbreak studies without agent implications, unsourced marketing claims, incident reporting without independently described methods, and duplicate preprint/conference versions. When a peer-reviewed version was identified, that version took precedence.

Eligibility was assessed at the title, abstract, and accessible study-description level for English-language work available by the cutoff. The primary stratum required a security or privacy threat, evaluation, or defense tied to tool-using, memory-enabled, or interacting AI agents; closely related LLM-integrated applications were retained only where they established an attack mechanism subsequently exercised by agents. Secondary surveys, public technical standards, and conceptual/reporting sources were coded separately. A peer-reviewed version replaced an earlier preprint when both described the same study. The final INCLUDED corpus contains 34 distinct records: 21 empirical/technical studies, five reviews, six standards or guidance documents, and two methodological/conceptual sources. These numbers are counts of cited records, not numbers returned, screened, or excluded in any database. No unrecorded search-flow statistics or risk-of-bias scores have been invented; the evidence register is supplied in Appendix B.

Table 1. Review decisions and evidentiary safeguards

Dimension	Protocol applied	Reporting safeguard
Time and language	Works available through 16 Oct 2026; English-language scholarly records	No post-cutoff findings inferred
Source hierarchy	Peer-reviewed proceedings → publisher records → arXiv preprints → standards	Preprints and guidance clearly labeled
Unit of evidence	Agent task, threat capability, system boundary, outcome	No mixing model-level and workflow-level results
Exclusion	Pure chatbot jailbreaking, unverifiable blogs, duplicate versions	Avoids artificial inflation of support
Synthesis	Thematic analysis plus cross-axis taxonomy and benchmark comparison	No statistical pooling of incomparable success rates
Reproducibility	Queries and inclusion logic documented; cited primary landing pages	Full corpus register; no unlogged screening totals or unsupported quality scores

3.2. Extraction, quality appraisal, and synthesis

The coding matrix recorded publication status; operating environment; agent actions and privileges; attacker-controlled surface; target asset; memory and cross-agent persistence; success criteria; proposed controls; and limitations of the reported task model. Coding proceeded by mechanism rather than by attack name alone. Primary findings were checked against official publication or archival records for quantitative assertions and identified as author-reported evidence. The analysis distinguished intermediate instruction capture, prohibited tool attempt, observed unauthorized effect, and contextual privacy violation. We synthesized qualitatively because base models, tools, defender settings, attack budgets, and outcome denominators varied materially across studies.

Four evidence-quality dimensions were assessed descriptively: explicit attacker control, task/environment realism, a verifiable final-effect endpoint, and availability of implementation or test details. No numerical risk-of-bias score was assigned and no quantitative meta-analysis was attempted. A single reviewer performed record selection and synthesis; an independent duplicate-screening or consensus process was not performed. These choices limit exhaustiveness and reproducibility and are stated to prevent the appearance of greater methodological certainty than the available audit trail supports.

The synthesis combined mechanism-based thematic coding with negative-case analysis. In particular, we compared studies reporting strong model susceptibility with studies finding that attackers struggle to complete end-to-end goals, and studies documenting successful task-alignment defenses with studies showing bypass under adaptive adversaries [9], [10], [13]. This approach helps explain apparently conflicting evidence without assuming that all experiments sampled identical security phenomena.

4. Agent Architecture and the Threat Boundary

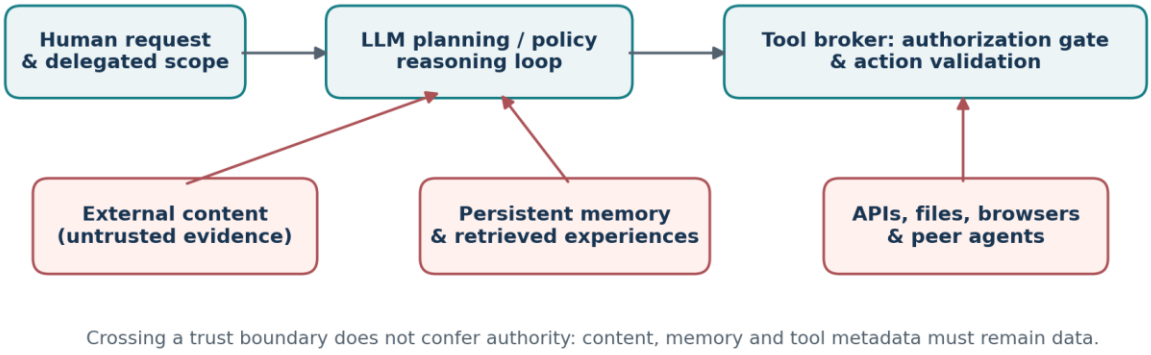


Figure 1. Reference architecture and principal trust-boundary crossings (original conceptual diagram).

Figure 1 represents an agent as a controller situated between user intent and an execution environment. The user's instruction carries an authorization context, but the model may observe lower-trust material that describes a page, file, search result, or tool. A key failure occurs when the planner promotes that descriptive material into operative authority. In security terms, the source possesses influence over input bytes but should not gain the privilege to choose the agent's goal or widen the agent's permissions [1], [4].

The execution broker is therefore a distinct enforcement component. It should validate that each proposed action is permissible for the current user, task, resource, and time. A tool name or natural-language explanation is not sufficient authorization; the server receiving an operation must independently enforce identity and scope. The broker also needs checks that are insensitive to the model's internal rationale, including deny-by-default write access, capability expiry, and confirmation of material side effects [27], [29].

Persistent state introduces a second temporal trust boundary. Memories may be authored by earlier users, derived from documents, or summarized by the agent itself. Their apparent location inside the agent must not confer epistemic or operational trust. Evidence from memory-extraction studies and poisoning attacks shows that information retained for future usefulness can later become a disclosure or control channel [11], [12], [18].

5. An Integrated Five-Axis Threat Taxonomy



Figure 2. Five-axis coding taxonomy for agent security and privacy incidents (original conceptual diagram).

Rather than classify attacks by a single name, the proposed taxonomy codes each reported case on five orthogonal axes. The entry axis records the attacker-controlled surface; propagation records where an injected instruction or sensitive datum travels; the asset axis identifies what can be affected; the outcome axis distinguishes confidentiality, integrity, availability, privacy, and unwanted autonomy; and temporality records whether effects are immediate, session-bounded, persistent, or cascading. An incident can have multiple coded outcomes or paths while retaining a clear primary event.

Table 2. Operational taxonomy and representative mechanisms

Axis	Values and decision rule	Representative evidence
A. Entry	Direct instruction; retrieved content; webpage pixels; tool description; memory ingestion; peer message	Indirect injections [1], [4]; visual WebInject [14]
B. Propagation	Prompt context; retrieval chain; tool output; tool-call argument; memory recall; agent conversation	AgentDojo [5]; MINJA [12]; ConVerse [15]
C. Target asset	Secret; private record; workflow decision; identity token; external service; durable memory	PrivacyLens [6]; MEXTRA [11]; MCP study [17]
D. Outcome	Disclosure; unauthorized action; integrity loss; resource abuse; normative privacy breach	ASB [8]; AgentHarm [7]; CoSec [19]
E. Temporality	Single step; within task; cross-session; transitive multi-agent effect	MemoryGraft [18]; ConVerse [15]

5.1. Untrusted-content instruction laundering

Indirect prompt injection is the most developed line of evidence. In an attack, hostile instructions are placed in material the agent must read for a legitimate objective. Greshake et al. established the mechanism across integrated applications; Liu et al. examined practical black-box attacks; InjecAgent created tool-embedded scenarios; and AgentDojo placed attacker-controlled inputs in dynamic work tasks [1], [2], [4], [5]. The causal failure is not exposure to adversarial text by itself, but the agent taking that text as authority for another action.

One reason prompt-based filters underperform is that the payload can mix harmless facts with persuasive process instructions or impersonate developer-level communication. The attack may request hidden actions, substitute a contact endpoint, or reorder a workflow while continuing to appear relevant to the

user's task. Mitigation thus requires preservation of source provenance and an explicit boundary between the user's legitimate instructions and retrieved observations; simple admonitions to ignore malicious content are insufficient as a sole control [9], [10].

5.2. Tool execution and excessive agency

Agent systems frequently expose high-privilege functions as natural-language descriptions. An agent permitted to read a calendar may also be able to modify an event; a browser tool can transmit query parameters; a shell or repository tool can change executable artifacts. Excessive agency describes this mismatch between necessary task capability and granted operational scope. The OWASP guidance identifies excessive functionality, excessive permissions, and excessive autonomy as root conditions [27].

For agent security, tool output is an observation and a tool invocation is an authorized operation: neither should be interchangeable with a model prediction. The decisive control is not merely whether a tool call looks reasonable, but whether the operation is consistent with verified user intent, the user's permissions, contextual purpose, and downstream impact. The same reasoning applies to Model Context Protocol integrations where tool metadata can carry attacker-controlled instructions and clients differ in validation behavior [17], [21], [29].

5.3. Long-term memory as both privacy store and attack carrier

Memory can contain private interaction transcripts, compact summaries, embeddings, feedback signals, and demonstrations. The design aims to preserve relevance across tasks, but retention lengthens the exposure window and complicates deletion. MEXTRA shows that private memory can be probed through ordinary agent interaction; PrivacyLens shows that agents can breach contextual privacy norms even without an adversarial attacker [6], [11]. Storage-layer encryption does not address inappropriate semantic release by an otherwise authorized model.

The opposite attack direction is memory poisoning. MINJA studies query-only insertion of malicious records into an agent's memory, while MemoryGraft investigates poisoning retrieved successful experiences so that unsafe procedures are imitated later [12], [18]. Such attacks make "memory" different from passive logs: recalled content can be used as guidance for future decisions. Defensive memory design should label source and owner, constrain retrieval by current authorization context, distinguish facts from instructions, and support correction, expiry, and demonstrable erasure.

5.4. Multi-agent trust and privacy by interaction

Inter-agent communication distributes context and delegated authority across separate models, services, and organizations. An untrusted peer can make an otherwise plausible request for a user attribute, attempt to induce tool execution, or present false policy claims. ConVerse demonstrates this setting using autonomous conversations in travel, real-estate, and insurance scenarios; its reported privacy and security outcomes depend on contextual dialogue, not only keyword leakage [15].

A permission to collaborate is not a blanket right to share the source user's full profile. The correct disclosure may be a derived answer (e.g., whether a criterion is satisfied) rather than the underlying record. This is consistent with contextual integrity and with data minimization, but requires explicit machine-enforceable policies over senders, recipients, purposes, and data categories [15], [19], [31]. Multi-agent scale also increases the challenge of tracing who caused an unauthorized action when requests are rephrased or handed off.

5.5. Multimodal and supply-chain surfaces

Web-based agents interpret both DOM-like text and screenshots. Text-only threat models may miss adversarial content encoded in an image, layout, or rendered pixel perturbation. WebInject demonstrates that altered page pixels can redirect vision-based agents, while WAInjectBench documents uneven performance of text and image detectors against more subtle perturbations [14], [20]. The observation

modality must therefore be reported in benchmark design, since a text guard may not defend a screenshot pathway.

Ecosystem dependencies create an additional class of failure. Tools, server manifests, registry metadata, templates, and shared memory components may change after initial review. MCP-focused studies discuss descriptor poisoning and the need for manifest integrity, behavioral monitoring, and user-visible authorization [17], [21]. This problem overlaps software supply-chain security yet differs because a syntactically valid descriptor can still contain a persuasive instruction addressed to the language model.

6. Privacy as a First-Class Agent Property

6.1. From data confidentiality to appropriate information flow

An agent can reveal information without hacking a database: it may simply compose a contextually inappropriate message from data it was legitimately shown. Shao et al.'s PrivacyLens evaluates behavior across realistic trajectories and finds a gap between responses to privacy questions and actual agent actions [6]. This distinction motivates a privacy model centered on information flows, not exclusively secret-string matching. The relevant variables include the subject of the data, the current task, the information recipient, the declared purpose, and the user's permission to disclose.

Privacy must also encompass inference. Summarization can transform records into sensitive conclusions; combination of innocuous clues can re-identify someone; and seemingly harmless tool logs can expose an authentication token or user preference. The boundary between authorized use and onward sharing needs to be evaluated after transformations, including aggregation or paraphrase. Normative privacy rules are context-dependent rather than reducible to one global blacklist [31].

6.2. A privacy threat model across the agent lifecycle

Table 3. Privacy-preserving controls across the agent data lifecycle

Lifecycle point	Privacy failure	Proposed control and measurement
Collect	Over-broad access to emails, contacts, browsing history	Purpose-scoped collection; measured excess-data rate
Represent	Private attributes included in prompts or embeddings	Field minimization, sensitive-label propagation
Retain	Stale summaries and memories exceed task duration	Retention budgets, deletion proof, owner-scoped access
Retrieve	Unrelated historical content influences new requests	Authorization-aware retrieval; cross-context leak tests
Share	Agent sends private context to external agent or tool	Recipient/purpose gating; minimal-disclosure outputs
Audit	Traces preserve raw personal details indefinitely	Selective logging, access controls, redaction efficacy

PrivacyLens and ConVerse suggest that privacy evaluation must ask not just whether an agent knows a policy, but whether it behaves accordingly during a legitimate task [6], [15]. An agent should ideally satisfy the user's task while minimizing dissemination of raw source details. Reporting task success alone can therefore hide privacy violations; conversely, a solution that simply refuses all tasks will fail the legitimate utility objective.

Memory and audit logs create a governance trade-off. Detailed traces are necessary for investigating unauthorized behavior, yet unconstrained traces can replicate the same personal-data concentration they are meant to detect. Privacy-preserving observability should favor action metadata, scoped identifiers,

protected logging channels, retention schedules, and incident-specific access over copying entire raw prompts into permanent dashboards. The desired balance is measurable accountability without default bulk retention [24], [25].

7. Evidence Synthesis Across Major Empirical Studies

The peer-reviewed benchmark literature reveals a common mechanism but divergent observation points. InjecAgent tests indirect attacks in tool-integrated scenarios and reports vulnerability under particular prompting conditions; AgentDojo exercises realistic tasks with dynamically situated attacks; ASB spans several attack families including memory poisoning; and WASP emphasizes completed web-agent compromise in constrained environments [4], [5], [8], [13]. Their numerical results do not estimate the same quantity. Careful synthesis therefore treats each benchmark as a conditional measurement.

Table 4. Selected primary evidence: attack surfaces, outcomes, and key interpretive caveats

Study	Test focus / reported finding	Interpretive boundary
Greshake et al. 2023 [1]	Indirect injection can alter integrated application behavior and tool use	Foundational cases; not population prevalence
InjecAgent 2024 [4]	1,054 attack tests spanning tool-integrated agent tasks	Success depends on prompt/tool setting
AgentDojo 2024 [5]	97 user tasks; 629 security tests in a dynamic sandbox	Task failure without attack affects interpretation
PrivacyLens 2024 [6]	Contextually inappropriate disclosure during agent actions	Privacy norms require case-specific judgments
AgentHarm 2025 [7]	110 malicious multi-step tasks across 11 harm categories	Misuse capability is not an indirect injection test
ASB 2025 [8]	Ten domains, diverse injection, backdoor and defense methods	Cross-scenario percentages require model matching
Adaptive attacks 2025 [9]	Eight tested indirect-injection defenses bypassed adaptively	Defender strength depends on attacker iteration
Task Shield 2025 [10]	Action-to-goal checking lowers benchmark attack success	Single defense and setting; test transfer needed
MEXTRA 2025 [11]	Black-box extraction from agent memory	Memory access and prompts define exposure
MINJA 2025 [12]	Query-only insertion of malicious memory records	Poisoning conditions vary across agent memories
WASP 2025 [13]	Intermediate hijack distinguished from final attacker goal	Partial compliance is not completed compromise
WebInject 2025 [14]	Visual perturbations redirect screenshot-using agents	Requires visual observation pathway
ConVerse 2026 [15]	864 contextual attacks in agent-to-agent dialogues	Context and disclosure norms are domain-dependent
MCP client analysis 2026 [17]	Seven client implementations assessed for tool poisoning	Preprint; implementation versions may change
CoSec 2026 [19]	208 scenarios with evolving privacy/authorization boundaries	Recent preprint; independent validation pending

7.1. Apparent contradictions and their resolution

Reported attack-success rates can differ for principled reasons. An attack may cause an agent to begin obeying a malicious instruction yet fail because the agent cannot navigate a site, because a tool rejects the operation, or because the attacker lacks a feasible end-to-end path. WASP is especially useful in reporting intermediate and final outcomes separately [13]. Any synthesis that treats partial hijack as completed harm overstates practical risk; any synthesis that ignores partial hijack overlooks a genuine boundary failure that stronger future agents could exploit.

Defense evidence also appears inconsistent until the threat model is aligned. On AgentDojo with GPT-4o, Task Shield reports a 2.07% attack success rate and 69.79% task utility under its evaluation [10]; by contrast, adaptive attacks against eight indirect-injection defenses report success exceeding 50% under a different, defense-aware threat model [9]. These percentages are not competing estimates of one underlying quantity. A credible defense comparison must match the model, environment, attacker query budget, tool privileges, legitimate-task utility, and exact success criterion.

Recent 2026 studies expand the analytic scope. ConVerse operationalizes contextual privacy in multi-agent exchange; CoSec examines changing community boundaries; MCP work focuses on poisoned tool descriptions; and surveys separate layer-specific compromise from runtime assurance [15], [17], [19], [22], [32]. The direction of the literature is clear: agentic security cannot be evaluated adequately as a static prompt classification problem, and it requires task traces, durable state, and policy-aware outcomes.

8. Defense-in-Depth and Secure Reference Design

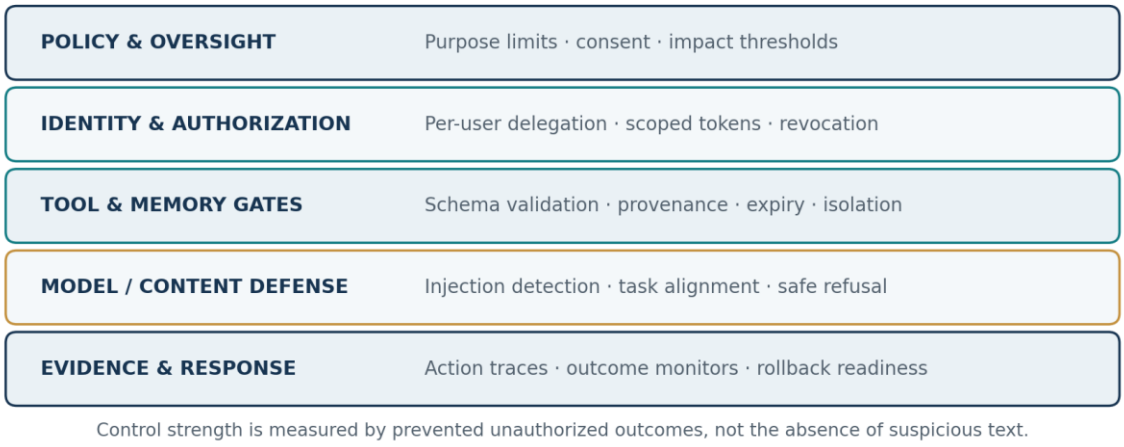


Figure 3. Layered controls for secure and privacy-preserving AI agents (original conceptual diagram).

A robust agent architecture needs preventive, detective, and recovery controls outside the language model. At the outermost boundary, a user identity and task-specific delegation are established. Before each tool call, a deterministic authorization service checks resource ownership, operation scope, policy, and impact. Lower-trust material is labeled and isolated from privileged policy instructions. Memory updates and reads are subjected to ownership, provenance, and lifetime constraints. Runtime monitors record the attempted action, the final external effect, and any intervention [24], [26], [27], [29].

This design deliberately does not require perfect detection of malicious language. A model can occasionally be manipulated yet fail to generate harm because the execution broker refuses an unauthorized operation. Conversely, a text filter can appear successful while the application retains unnecessary write privileges that turn unrelated model mistakes into harmful actions. The goal is the

conjunction of trustworthy reasoning and independently enforced authority; model-level guardrails are useful but not the sole security perimeter.

Table 5. Mechanism-to-control map and the evidence still needed

Failure mode	Defense-in-depth control	Critical evaluation
Indirect prompt injection	Source taint labels; task-intent checks; action authorization	Adaptive attacks and completed unauthorized effect
Over-broad tool authority	Least-privilege capabilities; per-user scopes; JIT access	Scope bypass, abuse through tool composition
MCP descriptor poisoning	Signed manifests; allowlisted metadata; server-side checks	Descriptor swaps and semantic instruction variants
Memory extraction	Owner-isolated storage and retrieval; output purpose checks	Query-only exfiltration across sessions
Memory poisoning	Ingestion provenance; quarantine; memory expiry	Delayed activation after benign interactions
Agent-to-agent leakage	Recipient- and purpose-aware disclosure policies	Leakage across context and role changes
Inadequate accountability	Protected action traces; event correlation; rollback	Causal reconstruction without broad raw-data retention

8.1. Authorization as a continuous process

Traditional permission systems often authorize a user when a session begins. Agents introduce a finer-grained problem: each newly proposed action may differ in purpose, sensitivity, and effect from the original request. A read-only task can spontaneously lead to write requests through planning drift or untrusted content. Authorization should therefore be rechecked at the tool boundary, with capability tokens scoped to the target server and resource, short lifetimes, and clear revocation behavior. The MCP specification addresses authorization flows and audience-bound tokens, but integration correctness remains the system implementer's responsibility [29].

Human approval should be risk-proportionate. Requiring consent for every low-impact read destroys utility and encourages habituation; never requiring consent for a payment or mass deletion is indefensible. Research should test approval design as a policy instrument with measurable false-approval burden, user understanding, and recovery efficacy. A human-in-the-loop prompt is not meaningful oversight if the system obscures the specific action, affected resource, recipient, or expected side effect [26], [28].

8.2. Provenance, isolation, and recovery

Every retrieved passage, tool message, and memory item should preserve machine-readable provenance: creator, time, owner, integrity status, and allowed use. This does not make the content true, but enables policy engines to distinguish untrusted observations from privileged instructions. Sandboxed tool execution, network egress limits, and transactional APIs reduce the blast radius of compromised plans. For actions that cannot be rolled back, such as sending confidential email, preventing execution matters more than subsequent alerting.

A recovery design must specify incident boundaries. If a poisoned memory influences multiple sessions, revoking a single tool credential may not erase the attack. Operators may need to isolate memories, replay trusted events, examine downstream outputs, and notify affected data subjects where legally required. Trace integrity and privacy-preserving forensic storage are therefore part of security

engineering, not optional operations work. The evidence base for such cross-session recovery remains substantially weaker than that for detecting isolated prompt injections.

9. Toward Outcome-Oriented Agent Security Evaluation

9.1. A hierarchy of evaluation outcomes

We recommend measuring five nested outcomes in an evaluation trace: (O1) the malicious content is encountered; (O2) the model adopts or repeats the attacker's instruction; (O3) an unauthorized operation is attempted; (O4) a protected external action or data release actually occurs; and (O5) the unauthorized effect persists or propagates after the initial interaction. The hierarchy separates model confusion from realized harm and makes it possible to identify which boundary stopped the attack. These levels should be reported separately rather than collapsed into one ambiguous "attack success" percentage [5], [8], [13].

The denominator is as important as the numerator. Attack success can be measured per payload, agent episode, unique user task, attacker budget, or eligible opportunity; changing the unit can change the interpretation. Authors should publish the number of legitimate tasks completed without attack, the attacked task set, the number and type of tool permissions, the attack budget, and all pass/fail conditions. For privacy, the outcome must include the recipient and the contextual rule violated, not merely the presence of a sensitive string [6], [15], [19].

Table 6. Suggested minimum reporting metrics for agent security studies

Metric	Operational definition	Why it matters
Legitimate task utility	Completed authorized tasks / attempted tasks	Avoids defenses that simply disable the agent
Instruction capture	Episodes with observed following of hostile content / exposed episodes	Diagnoses a trust-boundary failure
Unauthorized action rate	Episodes with prohibited tool attempts / attacked episodes	Measures externalization of harmful plans
Realized harm rate	Episodes with verified forbidden effects / attacked episodes	Separates intention from consequential compromise
Contextual disclosure	Disclosures failing recipient/purpose/data rules / assessed opportunities	Captures privacy beyond secret matching
Cross-session persistence	Clean follow-up tasks influenced by prior poison / follow-up tasks	Captures delayed memory attacks
Control cost	Latency, tokens, approvals, false blocks and intervention overhead	Exposes deployment feasibility

9.2. Evaluation design and external validity

A scientifically useful comparison should hold the application, model version, tool set, authorization policy, and attacker goal constant while varying one defense factor. Test sets should separate benign ambiguous data from malicious content; otherwise a detector can score well by labeling ordinary instructions in a retrieved document as malicious. Multi-turn trials are necessary to expose delayed manipulation, and cross-model replication is necessary to avoid tuning a control to one interface or prompt template [9], [20].

Benchmarks should disclose whether an external action was executed, simulated, or merely predicted by a judge model. Isolated environments improve safety and reproducibility, but attack difficulty depends on what the simulated user, bank, repository, or browser allows. Agreement among human

raters or independent outcome verifiers is especially important for privacy norms, which are often contextual. The evaluation should include negative controls, confidence intervals when repeated sampling is used, and clear accounting of failures attributable to ordinary agent incompetence.

A further challenge is compositional risk. Testing memory, tools, peers, and authorization separately may miss failures where a compromised memory changes which tool is selected, then a peer agent reuses the result under broader privileges. Evaluation environments should include multi-stage tasks with shared or segregated state, dynamic identities, authorization revocation, and post-task checks for latent compromise [12], [15], [19].

10. Research Gaps and a Future Research Agenda

10.1. Gap 1: Machine-checkable intent and delegated authority

Natural-language user instructions are incomplete policy specifications. A request to "organize my travel" does not define whether the agent may share passport details, contact a new vendor, spend money, or alter existing reservations. Future work should translate user intent into a constrained set of operations, recipients, data categories, budgets, and temporal scope, with explicit defaults for unspecified sensitive actions. The key hypothesis is that typed, revocable task delegation lowers completed unauthorized effects without substantially lowering authorized task completion compared with prompt-only policy text [27], [29], [32].

Research designs should implement the same agent across three conditions: conventional natural-language guardrails, static role-based permissions, and dynamically scoped action capabilities. Tests should include attack-induced goal drift and legitimate changes of plan. Both privacy-preserving disclosures and action outcomes must be measured, so an architecture cannot score highly merely because it rejects useful workflow changes.

10.2. Gap 2: Persistent-state integrity and selective forgetting

Memory research is moving beyond immediate retrieval corruption toward attacks that implant long-lived behavior [11], [12], [18]. Yet robust definitions of memory legitimacy, update authority, and deletion verification remain uncommon. A promising direction is provenance-preserving memory with typed fields, owner and purpose labels, evidence expiry, and reversible consolidation. Evaluations should insert malicious records, perform apparently unrelated clean tasks, and then measure delayed activation, data leakage, and successful targeted removal.

Selective forgetting is not simply removing a text row. Summaries, embeddings, caches, and agent-produced artifacts can preserve derivatives of the original record. Audits should verify that access revocation and deletion propagate through stores and re-indexing jobs. The research question is whether practical deletion guarantees can be specified at the level of accessible agent behavior rather than only at the level of storage APIs.

10.3. Gap 3: Confidential information flow between agents

Inter-agent ecosystems need explicit policies for which data may be shared to accomplish which task. Conventional API access controls can authorize the sending service yet fail to evaluate the semantics of what is sent. ConVerse and CoSec motivate recipient-aware, purpose-aware tests involving multiple participants and changing memberships [15], [19]. Methods such as attribute-level labels, context-aware consent receipts, privacy budgets for repeated interaction, and structured minimal-disclosure interfaces deserve comparative study.

The most informative benchmark would include lawful information sharing, prohibited disclosure, derived conclusions, and repeated inference requests in the same environment. Model behavior would be evaluated against a policy engine and a task utility oracle. A system that leaks no raw values yet reveals a sensitive attribute through repeated yes/no answers must still count as a privacy failure.

10.4. Gap 4: Adaptive and compositional attack evaluation

Static prompt corpora are insufficient once an adversary can observe the defense and iteratively adapt. Evidence from adaptive indirect-injection work demonstrates the importance of attacker budgets and knowledge assumptions [9]. More realistic evaluations should allow cross-modal payloads, tool metadata changes, poisoned memories, and peer relaying within constrained resources; they should track the entire sequence rather than only the first response.

Compositional attack benchmarks also require defenses to be combined. Provenance filtering, task alignment, scoped tokens, and monitoring may have non-additive costs or weaknesses. Factorial ablations can determine whether one layer meaningfully reduces harm after others are present, rather than merely improving a detector score on a narrow benchmark.

10.5. Gap 5: Formal assurance and human-governable autonomy

Current agent testing is predominantly empirical. Formal methods could add value by specifying invariants: a personal record may be sent only to an approved recipient for an approved purpose; a payment may be issued only below a user-approved ceiling; a read-only role can never mutate a file. The difficult part is connecting high-level language intent to concrete policy semantics without changing the task. Recent work on specification and enforcement emphasizes this bottleneck and warns that local blocking may still leave unsafe alternative paths [32].

Guarantees should be stated at clear abstraction levels. One can verify that a tool broker never accepts a write token for a read-only task, but cannot thereby guarantee every fact in a model's plan is true. Similarly, formal information-flow labels offer partial confidentiality assurances but may not resolve all contextual privacy judgments. The research priority is compositional proof of enforceable boundaries combined with empirical evaluation of semantic decisions at the model layer.



Figure 4. Proposed sequence of research priorities; horizons indicate relative maturity, not observed forecasts.

Table 7. Actionable research agenda with falsifiable research questions

Priority	Research question	Proposed study design / measurable outcome
P1: Delegation	Do typed task capabilities outperform natural-language guardrails?	Randomized agent trials; completed unauthorized action vs utility
P2: Memory	Can provenance and expiry block delayed-memory compromise?	Longitudinal poisoned-memory benchmark; persistence and deletion

P3: Privacy	Does recipient-aware disclosure reduce contextual leakage?	Agent-to-agent benchmark with policy-labeled flows
P4: MCP	Are signed and validated manifests sufficient against tool poisoning?	Descriptor-swap red teaming with client/version coverage
P5: Benchmarks	How much risk is hidden by intermediate-only metrics?	Paired O2–O5 scoring in identical agent episodes
P6: Human oversight	Which approval format improves genuinely informed decisions?	User studies: approval accuracy, burden, and delay
P7: Formal methods	Which invariants transfer across multi-step workflows?	Verified broker + adversarial task generator
P8: Recovery	Can poisoned state be isolated and safely replayed?	Cross-session incident simulations with recovery metrics

11. Implications for Practice and Governance

Developers should begin with an inventory of tools, data classes, and the identities on whose behalf actions occur. They should specify read/write access separately, prohibit arbitrary network egress by default, and make high-impact operations independently confirmable. Retrieval and external metadata require labeling as untrusted information. Memory should be partitioned by user and task with retention periods. Tool brokers should log verified outcomes and enforce authorization at the recipient service rather than trusting the agent's explanation [24]–[29].

Organizational risk assessment should evaluate specific workflows rather than assign one risk value to an entire "AI agent." An internal document summarizer with no outgoing write capability differs materially from a purchasing agent with payment credentials. Privacy impact assessment should identify not only the direct user, but also third parties whose records may enter the context. Explicit escalation and incident response playbooks matter for cross-service effects and durable memory poisoning.

Governance frameworks provide a broad lifecycle view but do not replace system-specific penetration testing. NIST AI RMF and its generative-AI profile support risk governance, while OWASP guidance describes relevant agentic threats and control families [24]–[28]. Evidence of conformance to a checklist should be paired with attack-trace evidence under realistic task privileges. The useful question is whether an unauthorized data flow or external action can still complete after the control is installed.

12. Limitations

This review has important limits. First, discovery relied on public scholarly search and primary source records, without preserved multi-database exports or independently adjudicated title/full-text exclusions; the source register cannot establish complete PRISMA 2020 identification and selection counts [30]. Second, the reviewed examples are concentrated in tool-using language-model agents rather than robotics, conventional autonomous systems, or proprietary production deployments. Third, heterogeneous attacker budgets, model versions, agent scaffolds, success metrics, and access permissions prohibit a defensible pooled effect size. Fourth, selected recent results remain preprints; their conclusions and publication statuses may change. Finally, source coding was performed by one reviewer rather than independently by multiple reviewers.

The proposed five-axis taxonomy, reference architecture, and prioritization scheme are analytical contributions, not experimentally validated countermeasures. A benchmark's attack frequency is not a population estimate of deployed-system incidence. No defense is represented as provably preventing all prompt injections, privacy violations, or cross-session compromise. We separate these original proposals from independently reported empirical measurements and identify close antecedents [22], [23], [32]–[34].

13. Conclusion

Security and privacy in AI agent systems are properties of complete delegated workflows. The model matters, but the decisive risks emerge where language is converted into authority, where external content enters planning, where persistent memory crosses time, and where agents transmit information to other agents or services. The evidence shows that indirect prompt injection, memory abuse, tool poisoning, and contextual disclosure are distinct mechanisms with overlapping consequences [4], [11], [15], [17].

The principal practical recommendation is to treat the model as a fallible planner rather than a trusted authorization server: enforce least privilege at every tool call, preserve provenance and purpose limits for state and data flows, inspect externally visible outcomes, and test with adaptive, multi-stage attacks. The principal research recommendation is to converge on verifiable outcome metrics and reusable stateful benchmarks while developing scoped delegation and information-flow mechanisms with clearly stated guarantees. Better agent security will not be demonstrated by a single lower attack-success percentage; it will be demonstrated by preserving legitimate usefulness while preventing unauthorized effects under transparent adversarial conditions.

References:

K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz, “Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection,” Proc. 16th ACM Workshop on Artificial Intelligence and Security, 2023. doi: 10.1145/3605764.3623985.

Y. Liu et al., “Prompt injection attack against LLM-integrated applications,” arXiv:2306.05499, 2023. [Preprint].

Y. Ruan et al., “Identifying the risks of LM agents with an LM-emulated sandbox,” International Conference on Learning Representations (ICLR), 2024.

Q. Zhan, Z. Liang, Z. Ying, and D. Kang, “InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents,” Findings of ACL, pp. 10471–10506, 2024. doi: 10.18653/v1/2024.findings-acl.624.

E. Debenedetti, J. Zhang, M. Balunović, L. Beurer-Kellner, M. Fischer, and F. Tramèr, “AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents,” Advances in Neural Information Processing Systems, vol. 37, 2024. doi: 10.52202/079017-2636.

Y. Shao, T. Li, W. Shi, Y. Liu, and D. Yang, “PrivacyLens: Evaluating privacy norm awareness of language models in action,” Advances in Neural Information Processing Systems, vol. 37, 2024. doi: 10.52202/079017-2837.

M. Andriushchenko et al., “AgentHarm: A benchmark for measuring harmfulness of LLM agents,” International Conference on Learning Representations (ICLR), 2025.

H. Zhang et al., “Agent Security Bench (ASB): Formalizing and benchmarking attacks and defenses in LLM-based agents,” International Conference on Learning Representations (ICLR), 2025.

Q. Zhan, R. Fang, H. S. Panchal, and D. Kang, “Adaptive attacks break defenses against indirect prompt injection attacks on LLM agents,” Findings of NAACL, pp. 7116–7132, 2025. doi: 10.18653/v1/2025.findings-naacl.395.

F. Jia, T. Wu, X. Qin, and A. Squicciarini, “The Task Shield: Enforcing task alignment to defend against indirect prompt injection in LLM agents,” Proc. 63rd Annual Meeting of the ACL, pp. 29680–29697, 2025. doi: 10.18653/v1/2025.acl-long.1435.

B. Wang, W. He, S. Zeng, Z. Xiang, Y. Xing, J. Tang, and P. He, “Unveiling privacy risks in LLM agent memory,” Proc. 63rd Annual Meeting of the ACL, pp. 25241–25260, 2025. doi: 10.18653/v1/2025.acl-long.1227.

S. Dong et al., “Memory injection attacks on LLM agents via query-only interaction,” Advances in Neural Information Processing Systems, vol. 38, 2025. doi: 10.52202/085713-1554.

I. Evtimov, A. Zharmagambetov, A. Grattafiori, C. Guo, and K. Chaudhuri, "WASP: Benchmarking web agent security against prompt injection attacks," Advances in Neural Information Processing Systems, vol. 38, 2025. doi: 10.52202/085713-0666.
X. Wang, J. Bloch, Z. Shao, Y. Hu, S. Zhou, and N. Z. Gong, "WebInject: Prompt injection attack to web agents," Proc. EMNLP, pp. 2010–2030, 2025. doi: 10.18653/v1/2025.emnlp-main.104.
A. Gomaa, A. Salem, and S. Abdelnabi, "ConVerse: Benchmarking contextual safety in agent-to-agent conversations," Findings of EACL, pp. 3246–3268, 2026. doi: 10.18653/v1/2026.findings-eacl.170.
Y. Zhu et al., "Teams of LLM agents can exploit zero-day vulnerabilities," Proc. EACL, pp. 23–35, 2026. doi: 10.18653/v1/2026.eacl-long.2.
C. Huang, X. Huang, N. P. Tran, and A. M. Fard, "Model Context Protocol threat modeling and analyzing vulnerabilities to prompt injection with tool poisoning," arXiv:2603.22489, 2026. [Preprint].
S. S. Srivastava and H. He, "MemoryGraft: Persistent compromise of LLM agents via poisoned experience retrieval," arXiv:2512.16962, 2025. [Preprint].
H. Chen et al., "CoSec: Benchmarking agent security in communities," arXiv:2609.34790, 2026. [Preprint].
Y. Liu, X. Wang, R. Xu, Y. Jia, and N. Z. Gong, "WAInjectBench: Benchmarking prompt injection detections for web agents," arXiv:2510.01354, 2025. [Preprint].
S. Jamshidi, K. W. Nafi, A. M. Dakhel, N. Shahabi, F. Khomh, and N. Ezzati-Jivan, "Securing the Model Context Protocol: Defending LLMs against tool poisoning and adversarial attacks," arXiv:2512.06556, 2025. [Preprint].
K. Chu, "A systematic survey of security threats and defenses in LLM-based AI agents: A layered attack surface framework," arXiv:2604.23338, 2026. [Preprint].
Y. Xu et al., "LLM agents security duality: A comprehensive survey of self-security and empowered cybersecurity," Artificial Intelligence Review, vol. 59, art. 174, 2026. doi: 10.1007/s10462-026-11563-0.
E. Tabassi, Artificial Intelligence Risk Management Framework (AI RMF 1.0), National Institute of Standards and Technology, NIST AI 100-1, 2023. doi: 10.6028/NIST.AI.100-1.
C. Autio et al., Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile, NIST AI 600-1, 2024. doi: 10.6028/NIST.AI.600-1.
OWASP Gen AI Security Project, "Agentic AI: Threats and mitigations," technical guidance, 2025 (Feb. 17).
Model Context Protocol, "Specification: Authorization," revision 2025-11-25, 2025.
H. Nissenbaum, "Privacy as contextual integrity," Washington Law Review, vol. 79, no. 1, pp. 119–157, 2004.
P. Dantas, L. Cordeiro, E. Nowroozi, and N. Tihanyi, "Toward safe LLM agents: A survey of specification, verification, and enforcement," arXiv:2608.14590, 2026. [Preprint].
Y. Gan et al., "Navigating the risks: A survey of security and privacy threats in LLM-based agents," ACM Transactions on Software Engineering and Methodology, accepted Mar. 2026, published online May 2026. doi: 10.1145/3807666.
V. Sammartino, "Vulnerabilities in autonomous execution: A survey of security threats and defenses in LLM-driven multi-agent systems," SSRN, preprint posted Aug. 1, 2026, SSRN abstract 7218206. [Preprint].
OWASP Gen AI Security Project, "OWASP Top 10 for LLM Applications, 2025 edition," 2025.
OWASP Gen AI Security Project, "OWASP Top 10 for Agentic Applications for 2026," Dec. 9, 2025.
M. J. Page et al., "The PRISMA 2020 statement: An updated guideline for reporting systematic reviews," BMJ, vol. 372, art. n71, 2021. doi: 10.1136/bmj. n71.