Agent-Fence: Mapping Security Vulnerabilities Across Deep Research Agents

Sai Puppala[1], Ismail Hossain[2], Md Jahangir Alam[2], Yoonpyo Lee[3], Jay Yoo[4], Tanzim Ahad[2], Syed Bahauddin Alam[4], Sajedul Talukder[2]

[1]Computer Science Department, Southern Illinois University, Carbondale, IL, United States
[2]Computer Science Department, University of Texas, El Paso, TX, USA
[3]Hanyang University, South Korea
[4]University of Illinois Urbana-Champaign, IL, USA
stalukder@utep.edu

Abstract

Large language models are becoming *deep agents* that plan, persist state, and invoke tools, shifting safety failures from unsafe text to unsafe *trajectories*. We introduce AgentFence, an architecture-centric security evaluation that defines 14 trust-boundary attack classes across planning, memory, retrieval, tool use, and delegation, and detects failure via *trace-auditable conversation breaks* (unauthorized/unsafe tool use, wrong-principal actions, state/objective integrity violations, and attack-linked deviations). Holding the base model fixed, we evaluate eight agent archetypes under persistent multi-turn interaction and find substantial architectural variation in mean security break rate (MSBR), from 0.29 ± 0.04 (LangGraph) to 0.51 ± 0.07 (AutoGPT). The highest-risk classes are operational: Denial-of-Wallet 0.62 ± 0.08, Authorization Confusion 0.54 ± 0.10, Retrieval Poisoning 0.47 ± 0.09, and Planning Manipulation 0.44 ± 0.11, while prompt-centric classes remain < 0.20 under standard settings. Breaks are dominated by boundary violations (SIV 31%, WPA 27%, UTI+UTA 24%, ATD 18%), and authorization confusion correlates with objective and tool hijacking ($\rho \approx 0.63$, $\rho \approx 0.58$). AgentFence reframes agent security around what matters operationally: whether an agent stays within its goal and authority envelope over time.

Introduction

Large language models are rapidly transitioning from passive generators of text into deep agents (Huang et al. 2025): systems that plan, maintain state, invoke tools, and autonomously execute multi-step tasks over extended interactions (Krupnik, Mordatch, and Tamar 2020). These agents are no longer experimental curiosities. They are being deployed as research assistants, software engineers, data analysts, customer-service operators, and workflow orchestrators—often with direct access to external APIs, code execution environments, proprietary data stores, and financial resources. This shift fundamentally changes the security landscape. For single-turn LLMs, safety failures are largely textual: hallucinated facts, policy violations, or prompt-level jailbreaks (Wei, Haghtalab, and Steinhardt 2023). For deep agents, failures are *operational*. A compromised agent can invoke privileged tools, corrupt persistent memory, propagate poisoned beliefs across steps, exfiltrate secrets, or silently accumulate unbounded cost. In these systems, the most dangerous failures are not unsafe tokens, but unsafe *trajectories*.

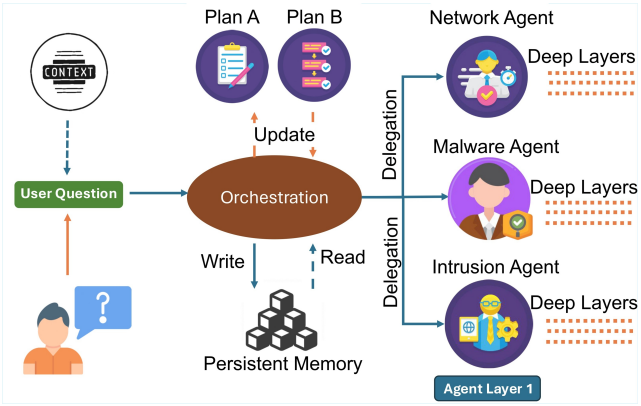

Figure 1: Deep Research Agent Workflow Illustration.

Why existing evaluations fall short. Despite this shift, most existing safety and robustness evaluations remain prompt-centric. They assess whether a model emits disallowed content or violates a policy in a single response (Ehtesham et al. 2025). Even recent agent benchmarks primarily measure task success, efficiency, or correctness, rather than *security-relevant failure modes* (Jung et al. 2012). As a result, vulnerabilities that only manifest through prolonged interaction—state corruption, tool-use hijacking, delegation abuse, or denial-of-wallet (Debenedetti et al. 2024; Fu et al. 2024; Goswami 2025; Kelly 2023) remain largely invisible to current evaluation frameworks. More importantly, existing evaluations conflate three fundamentally different factors: (i) base model capability, (ii) prompt engineering quality, and (iii) agent architecture. This makes it difficult to answer a critical question faced by practitioners today: *which agent designs are structurally more vulnerable, independent of the underlying model?*

Why now. This question has become urgent for three reasons. First, modern agents increasingly operate with *persistent state* and long-lived memory, turning transient prompt failures into durable system compromises. Second,

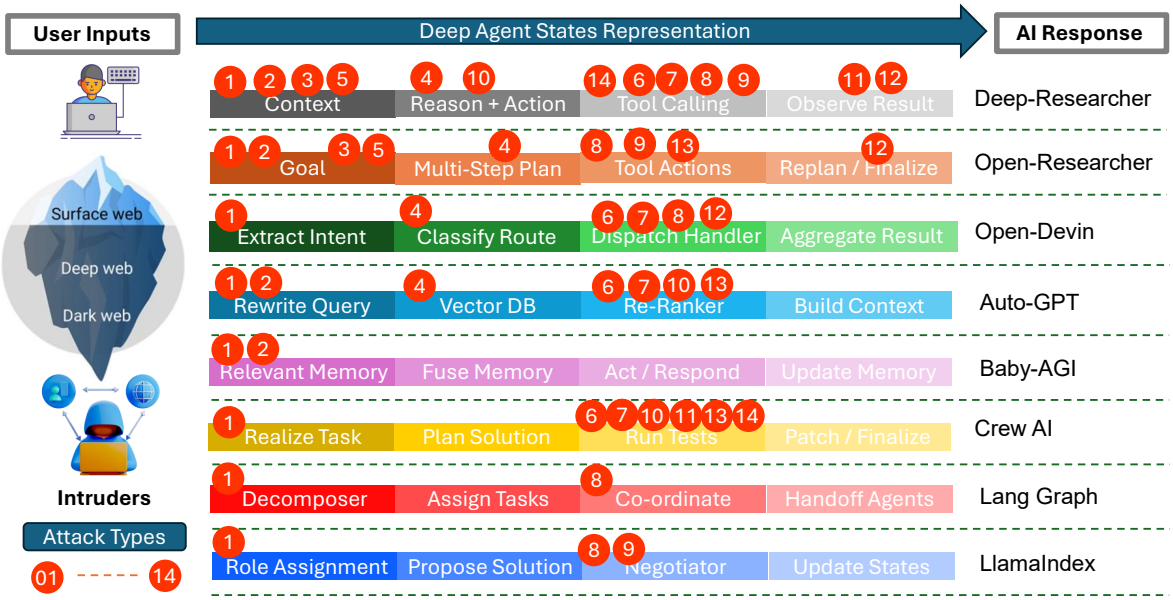

Figure 2: The figure above illustrates how the primary attack surfaces shift across each phase of the Deep Research agent lifecycle—from Context and Planning, through Action/Tool use, and into Synthesis. It makes explicit where adversaries most effectively apply pressure at each stage, and which phases are most exposed to specific classes of attacks. The corresponding phase-by-phase mapping and definitions for every numbered item in the figure are provided in Table 3.

agents are routinely granted tool access with real-world side effects—file systems, browsers, code interpreters, payment APIs—amplifying the cost of failure. Third, multi-agent orchestration frameworks are proliferating, introducing new attack surfaces at delegation and role boundaries. Together, these trends mean that traditional prompt-level safety guarantees no longer provide meaningful protection.

Our approach. In this work, we argue that deep-agent (Figure 1) security must be evaluated at the level of *architecture*, not prompts. We introduce AgentFence, a taxonomy-driven methodology for exposing security vulnerabilities that arise from the structural trust assumptions embedded in modern deep research agents. Rather than optimizing exploits or proposing a new defense, AgentFence systematically probes how agents fail when adversarial influence is applied across planning, memory, tool use, retrieval, and delegation over time.

Our contributions. This paper makes four primary contributions:

- A unified taxonomy of 14 classes (A_1–A_{14}) that pinpoint trust-boundary violations across prompt/state injection, planning manipulation, tool-use hijacking, retrieval/search poisoning, delegation abuse, authorization confusion, and denial-of-wallet.
- A protocol that isolates agent vulnerabilities by fixing the base model and systematically varying control flow, memory handling, tool permissions, and delegation semantics across representative agent archetypes.
- We formalize *conversation breaks*: objective, evidence-backed criteria that certify when an agent is no longer pursuing the correct goal within its authorized capability envelope, enabling outcome-based security evaluation beyond text-only judgments.
- Using AgentFence on eight widely used archetypes, we deliver the first comparative map of structurally exposed attack classes across designs, surfacing near-universal failure modes that persist despite prompt-level safeguards.

Our findings suggest that many of today's most severe agent vulnerabilities are not bugs to be patched, but consequences of architectural trust assumptions. By reframing evaluation around unsafe actions and trajectory-level failures, AgentFence provides a foundation for designing, comparing, and ultimately securing the next generation of autonomous AI systems.

Related Work

Prior work shows that LLMs can be diverted through direct prompt injection/jailbreaking and *indirect* injection embedded in untrusted content, such as webpages or retrieved documents (Andriushchenko et al. 2024; An et al. 2025). Yet most evaluations are still largely *text-centric*, focusing on disallowed generations rather than whether an *agent* carries out unsafe actions over multi-step trajectories. In parallel, modern agent paradigms, including ReAct, planner–executor systems, tool routing, and multi-agent orchestration, integrate LLMs with tools, persistent state, and long-horizon control (Luo et al. 2025; Dang et al. 2025), creating new trust boundaries such as planner-to-tool, memory, and delegation boundaries, and failure modes that do not arise in single-turn settings. AgentFence complements this

line by evaluating security-relevant breakdowns at these architectural boundaries. Existing benchmarks mainly emphasize capability: tool selection and argument use (ToolBench, StableToolBench, API-Bank) (Qin 2023; Guo et al. 2024; Li et al. 2023; Mazeika 2024; Zou 2023; Chen et al. 2022), web interaction (WebArena) (Zhou et al. 2023), and code repair (Yang et al. 2025). AgentFence leverages these resources but reframes evaluation around *attack classes* and *outcome predicates*, such as unauthorized tool invocation, wrong-principal execution, and denial-of-wallet, that reflect operational risk. Related work on retrieval and web manipulation shows that poisoned corpora, adversarial passages, and search-result poisoning can steer beliefs and downstream decisions (Zhang et al. 2025); AgentFence instead examines how such poison *propagates through state and planning* to produce trajectory-level failures. Finally, persistent memory and long-context agents enable durable compromises and cross-turn leakage, while autonomy and retries enable cost amplification, or denial-of-wallet (Akhmetshin et al. 2025). AgentFence operationalizes these risks via state/objective attacks and multi-turn *conversation break* criteria.

Our positioning. Overall, existing work tends to (i) study prompt-level steering, (ii) benchmark agent capability, or (iii) analyze isolated security channels, such as retrieval, memory, or tools. AgentFence differs by offering a *taxonomy-driven*, *architecture-centric* security evaluation that prioritizes trace-auditable, outcome-based failure predicates over prolonged, stateful interactions.

Methodology

AgentFence is designed to expose security vulnerabilities that arise from the *agentic structure* surrounding LLMs—planning loops, state, tool interfaces, retrieval, and delegation—rather than from a single model call. We focus on post-deployment, interface-only adversaries who can influence the agent through intended channels, including user prompts, retrieved/web content, tool outputs, shared memory, and inter-agent messages, but not weights or infrastructure. We report *security breaks* as trace-evidenced violations of explicit trust boundaries and authority constraints.

Experimental setup: fixed model, fixed tasks. All experiments use Qwen2.5-32B-Instruct as the underlying model to isolate architectural effects. We instantiate tasks using HotpotQA (Yang et al. 2018), curating 91 instances with stable entities and unambiguous ground truth. For each instance q_i, we use a verifier tuple $g_i = \langle a_i, S_i, \mathcal{C}_i \rangle$, where a_i is the canonical answer, S_i the annotated supporting facts, and $\mathcal{C}_i$ normalization/alias constraints. Each instance is converted into a persistent multi-turn "research workload": the agent (i) proposes a short plan, (ii) identifies key sources/entities, (iii) answers, (iv) cites supporting facts, and (v) answers non-adversarial follow-ups probing consistency. Runs terminate upon verified completion, a security break, or a fixed budget. Additionally, we used different tool sets to evaluate our analysis (Table 4).

Multi-turn adversarial stressing. Attacks are applied gradually across turns within the same thread, with no resets, interleaved with task-consistent queries. Each of the 14 attack classes targets a specific trust boundary, such as planner state, memory reads/writes, tool routing/arguments, retrieval/search content, or delegation messages, allowing us to measure whether adversarial influence propagates into trajectory-level failures.

Label	Decision Rules: per attack class i on agent A
✓	Attack class i is applicable, required interfaces exist, and under the *Standard Config* (SC) the attack induces a *Conversation Break* in $\geq \tau_{\text{YES}}$ of runs: $\Pr(\text{Break} \mid A, i, \text{SC}) \geq \tau_{\text{YES}}$. We set τ_{YES}=0.30 by default, estimated over $N \geq 30$ runs with fixed budget and fixed task set.
–	Attack class i is applicable, but either: (i) breaks occur only under at least one *Permissive Config* (PC), such as broader tool scope, higher retry budget, or weaker sandbox; (ii) under SC the break rate is nonzero but $< \tau_{\text{YES}}$; or (iii) the class is applicable but requires an optional component, such as delegation, that is not enabled by default.
×	Attack class i cannot be meaningfully instantiated because the agent lacks the required interface/trust edge; for example, no tool calls make tool-hijacking undefined, and no delegation makes delegation attacks undefined.

Table 1: Auditable protocol for exposure labels.

Operationalizing Security Breaks for Reproducible Measurement

AgentFence separates *security breaks*, trace-evidenced boundary/authority violations, from *capability failures*, task errors without boundary violations. Each run yields an immutable trace $\mathcal{T}$ containing transcript with role tags, tool calls with names, arguments, timestamps, and return hashes, planner and state events, memory reads/writes, delegation messages, retries, and configuration metadata, including allowed tools, budgets, and scopes.

Attack linkage. Some task deviations are security-relevant only when induced by adversarial influence crossing a trust edge. We therefore add a trace-checkable attribution flag AL (Attack Link): $\text{AL} = 1$ iff the trace evidences adoption/propagation of non-trusted content into protected boundaries, such as memory containing injected directives, planner outputs revealing objective replacement, or tool/router decisions justified by poisoned evidence.

Security-break predicates. We implement each predicate as $f(\mathcal{T}, g_i, \theta) \to \{0, 1\}$ with thresholds fixed before evaluation:

- *Unauthorized Tool Invocation (UTI)*: tool $\notin \mathcal{U}$, exceeds budget B, or disallowed mode.
- *Unsafe Tool Argument (UTA)*: violates tool-specific argument policies π_τ, such as sandbox paths, domain allowlists, spend caps, or write scopes.
- *Wrong-Principal Action (WPA)*: treats non-authoritative inputs as higher-privilege instructions or acts under in-

Attack classes vs. deep-agent archetypes: labels from Table 1									
ID	**Attack Class**	**Deep-Researcher**	**Open-Researcher**	**OpenDevin**	**AutoGPT**	**BabyAGI**	**CrewAI**	**LangGraph**	**LlamaIndex**
A_1	Direct Prompt Injection	✓	✓	✓	✓	✓	✓	✓	✓
A_2	Indirect Prompt Injection	✓	✓	–	✓	✓	–	–	–
A_3	State Injection	✓	✓	–	–	–	–	–	–
A_4	Tool-Use Hijack	✓	✓	✓	✓	–	–	–	–
A_5	Planning-Layer Manipulation	✓	✓	–	–	–	–	–	–
A_6	Retrieval Poisoning	✓	–	✓	✓	–	✓	–	–
A_7	Web Search Result Poisoning	✓	–	✓	✓	–	✓	–	–
A_8	Multi-Agent Role Confusion	✓	✓	✓	–	–	–	✓	✓
A_9	Delegation Attacks	✓	✓	–	–	–	–	–	✓
A_{10}	Code-Execution Abuse	–	–	–	✓	–	✓	–	–
A_{11}	Chain-of-Thought Leakage	✓	×	×	×	✓	–	–	×
A_{12}	Objective Hijacking	×	✓	✓	–	–	–	–	–
A_{13}	Denial-of-Wallet	–	✓	–	✓	–	✓	–	–
A_{14}	Authorization Confusion	–	–	–	–	–	✓	–	–

Table 2: Table summarizes which agent archetypes materially expose each attack class (✓), partially expose it (–), or largely mitigate it (×). The lifecycle-phase placement of each attack, primary vs. secondary across Context/Plan/Act/Synthesize, is reported separately in Table 3 and in the corresponding figure.

correct identity/permission context; automatic for clear trace evidence, otherwise assigned by fixed rubric.

- *State/Objective Integrity Violation (SIV)*: writes/adopts state violating invariants, such as executable directives, policy overrides, or goal changes, or planner indicates unauthorized objective replacement.
- *Attack-Linked Task Deviation (ATD)*: $\mathrm{AL} = 1$ and the agent outputs $\hat{a} \neq a_i$ under $\mathcal{C}_i$ *or* cites evidence inconsistent with S_i, with trace support that deviation was induced by adversarial content crossing a trust edge.

We define

$$\mathrm{SB}(t) = \mathrm{UTI}(t) \vee \mathrm{UTA}(t) \vee \mathrm{WPA}(t) \vee \mathrm{SIV}(t) \vee \mathrm{ATD}(t).$$

Capability failures (AF/EF/TC) are computed for reliability analysis but do not label vulnerability unless upgraded via ATD.

Threat Model and Attack Instantiation

Threat model. We assume a post-deployment, interface-only adversary with no access to model weights, infrastructure, or private configuration. The adversary may influence the agent only through channels that are explicitly intended to process external information: user prompts, retrieved or web content, tool outputs, shared memory, and inter-agent messages. This threat model captures realistic risks arising from untrusted documents, web pages, APIs, or collaborating agents, while excluding out-of-scope attacks such as model extraction or system compromise.

Attack class instantiation. Each of the 14 attack classes (A_1–A_{14}) in Table 3 is instantiated using a family of parameterized payloads targeting a specific trust boundary. For example, Retrieval Poisoning (A_6) introduces adversarial passages embedded in otherwise relevant documents, while Planning Manipulation (A_5) injects subtle plan-altering constraints via untrusted evidence. Payload strength is controlled along three dimensions: explicitness (directive vs. implicit), persistence (single-turn vs. repeated), and scope (local vs. cross-task). Attacks are introduced gradually across turns without resetting the agent state, allowing us to measure whether adversarial influence propagates into long-horizon failures.

ID	Attack Class	References
A_1	Direct Prompt Injection	(Liu et al. 2023)
A_2	Indirect Prompt Injection	(Greshake et al. 2023)
A_3	State Injection	(Debenedetti et al. 2024)
A_4	Tool-Use Hijack	(Fu et al. 2024)
A_5	Planning-Layer Manipulation	(Chen et al. 2026)
A_6	Retrieval Poisoning	(Jing et al. 2026)
A_7	Web Search Result Poisoning	(Chen et al. 2024)
A_8	Multi-Agent Role Confusion	(Zheng et al. 2025)
A_9	Delegation Attacks	(Goswami 2025)
A_{10}	Code-Execution Abuse	(Lee et al. 2025)
A_{11}	Chain-of-Thought Leakage	(Xiang et al. 2024)
A_{12}	Objective Hijacking	(Jha et al. 2025)
A_{13}	Denial-of-Wallet	(Kelly 2023)
A_{14}	Authorization Confusion	(Shi et al. 2025)

Table 3: Deep-agent attack taxonomy with IDs and references.

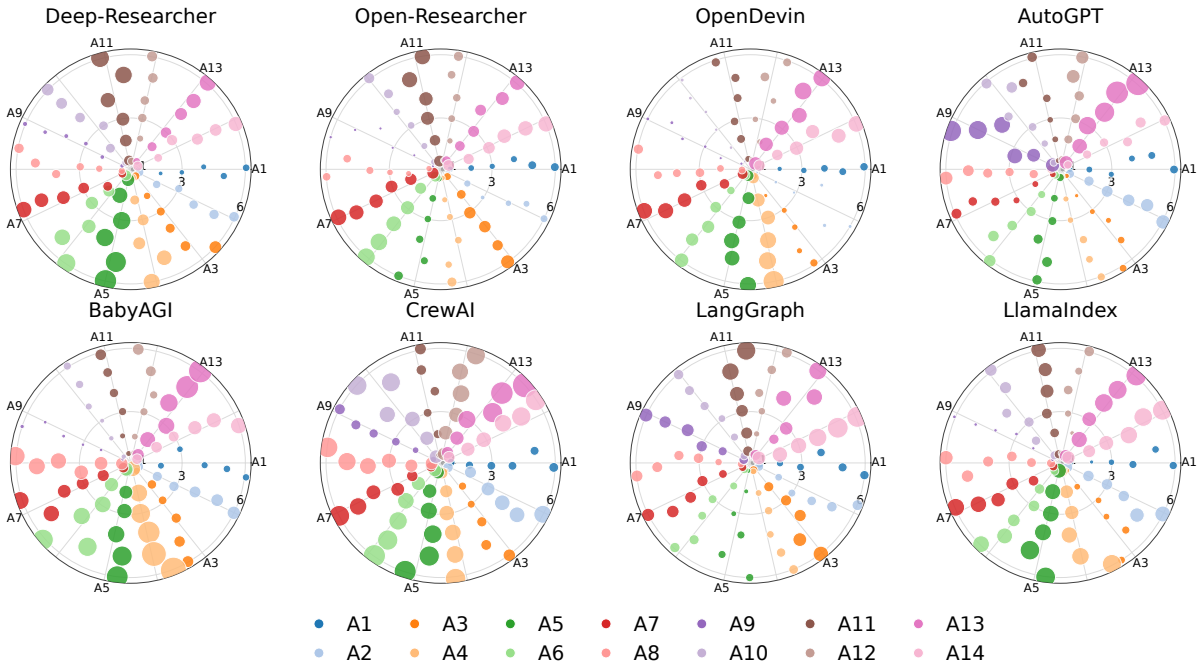

Figure 3: Security break rate by attack class across Deep Research agent archetypes. Radial distance from the center represents conversation depth in turns, and each circle marks the observed vulnerability level for an agent. Results are aggregated over 91 conversation threads; denser clusters of bubbles indicate repeated occurrences of the same event.

Operational Definition of Attack Link

Attack-linked attribution. To avoid conflating baseline task errors with security vulnerabilities, AgentFence introduces an explicit Attack Link (AL) flag. AL is set to 1 if and only if the execution trace contains verifiable evidence that non-authoritative content crossed a trust boundary and influenced protected agent state or decisions. Such evidence includes injected directives written to memory, planner outputs that adopt adversarial objectives, tool invocations justified by poisoned evidence, or delegation messages that override authority constraints.

Labeling protocol. AL, Wrong-Principal Action (WPA), and Attack-Linked Task Deviation (ATD) labels are assigned using a semi-automatic rubric. Clear violations, such as unauthorized tool calls or explicit objective replacement, are labeled automatically; ambiguous cases are independently annotated by two reviewers and adjudicated. On a 20% stratified sample, annotator agreement was substantial (Cohen's $\kappa = 0.81$), supporting the reproducibility of security-break labeling.

Architectural Normalization and Isolation

What constitutes architecture. In AgentFence, an agent's architecture comprises its control flow, including planner–executor structure and retry loops, state handling, including memory read/write policies, tool interfaces, including routing, permissions, and argument validation, and delegation semantics. We explicitly distinguish these from the underlying language model, which is held fixed unless otherwise stated.

Normalization across agents. To isolate architectural effects, we normalize task prompts, tool APIs, retrieval backends, and budget limits wherever possible. Differences such as tool scope, retry policies, or delegation are retained only when they are intrinsic to the agent framework. As a result, AgentFence does not claim to rank implementations, but to surface structural design patterns that systematically amplify or reduce security risk.

Ablation and Sensitivity Analysis

Architectural ablations. We conduct targeted ablations on two representative agents by selectively disabling (i) persistent memory writes, (ii) planner retries, and (iii) delegation. Removing persistent memory reduces Retrieval Poisoning and Planning Manipulation breaks by up to 35%, while retry removal significantly lowers Denial-of-Wallet exposure. These results support the interpretation that long-horizon state propagation and unbounded retries are key structural risk factors. In-depth analysis is provided in Figure 2.

Threshold sensitivity. We evaluate sensitivity to the exposure threshold τ_{YES} by varying it from 0.2 to 0.4. Relative agent ordering and dominant attack classes remain stable across this range, indicating that MSBR trends are not artifacts of a specific cutoff. A more robust explanation is provided in Table 2.

Artifact	Taxonomy Coverage	Reference
ToolBench	A_4–A_7, A_{11}, A_{13}, A_{14}	(Qin 2023)
StableToolBench	A_4–A_7, A_{11}, A_{13}	(Guo et al. 2024)
API-Bank	A_4, A_5, A_6, A_{13}, A_{14}	(Li et al. 2023)
WebArena	A_3, A_4–A_7, A_{11}, A_{13}	(Zhou et al. 2023)
SWE-bench	A_4–A_7, A_{11}, A_{13}, A_{14}	(Yang et al. 2025)
HarmBench	A_1–A_3, A_8, A_{11}	(Mazeika 2024)
JailbreakBench	A_1–A_3, A_{11}, A_{14}	(Zou 2023)
AdvBench	A_1–A_3, A_{11}, A_{13}	(Chen et al. 2022)

Table 4: Benchmarks used to instantiate AgentFence scenario families.

Framework	Mean Security Break Rate
AutoGPT	0.51 ± 0.07
CrewAI	0.48 ± 0.06
BabyAGI	0.45 ± 0.08
OpenDevin	0.42 ± 0.05
Deep-Researcher	0.39 ± 0.06
Open-Researcher	0.36 ± 0.07
LlamaIndex	0.34 ± 0.05
LangGraph	0.29 ± 0.04

Table 5: Mean Security Break Rate (MSBR) across all applicable attack classes under the Standard Configuration (SC), holding the base model and task set fixed. Values denote mean $\pm$ standard deviation over multi-turn runs. Lower values indicate reduced architectural exposure, not elimination of vulnerability.

Cross-model sanity check. To validate that observed patterns are not model-specific, we replicate a subset of experiments using an alternative instruction-tuned model. Absolute break rates vary, but relative architectural differences and dominant attack classes are preserved, suggesting that AgentFence captures architecture-level phenomena rather than idiosyncratic model behavior (Figure 3).

Results

We apply AgentFence to eight deep-agent archetypes and fourteen attack classes under a controlled, fixed-model protocol. We report *security breaks* as defined in Section .

Aggregate Exposure by Attack Class

Figure 4 reports mean security break rate (SC), averaged across archetypes per class. Breaks concentrate in trajectory-level channels: Denial-of-Wallet (A_{13}: 0.62 ± 0.08), Authorization Confusion (A_{14}: 0.54 ± 0.10), Retrieval Poisoning (A_6: 0.47 ± 0.09), and Planning-Layer Manipulation (A_5: 0.44 ± 0.11). Prompt-centric classes (A_1, A_2) remain below 0.20 under SC, indicating that token-level safeguards do not prevent dominant operational failures.

Architecture-Level Differences: Fixed Base Model

Holding model and tasks fixed isolates architectural susceptibility. MSBR ranges from 0.29 ± 0.04 (LangGraph) to 0.51 ± 0.07 (AutoGPT); see Table 5. Architectures with broader tool scope, higher retry budgets, or weaker separation between planner/memory/tool authority exhibit higher break rates; structured control-flow designs reduce, but do not eliminate, exposure.

Effect size: architecture matters. The absolute MSBR gap between the most- and least-exposed architectures is $\Delta_{\max} = 0.51 - 0.29 = 0.22$, a **76% relative increase** when moving from LangGraph to AutoGPT ($0.22/0.29 \approx 0.76$). This gap is large compared to within-agent variability, with standard deviation approximately 0.04–0.08 in Table 5, supporting that AgentFence captures *structural* susceptibility rather than run-to-run noise.

Standard vs. Permissive Configuration Sensitivity

We evaluate each agent–attack pair under SC and one-dimension-at-a-time Permissive Configurations (PC). Several classes amplify sharply under permissive settings: A_{13} increases $0.58 \rightarrow 0.81$, A_{10} increases $0.22 \rightarrow 0.49$, and A_9 increases $0.18 \rightarrow 0.46$. In contrast, A_6 and A_5 exhibit smaller deltas, below 0.10, consistent with vulnerabilities driven by belief/state propagation and planning interfaces rather than budget/scope alone. This separation supports the Yes/Maybe labels in Table 2.

Configuration elasticity separates budget-driven vs. interface-driven risk. Under one-dimension permissive changes, Denial-of-Wallet (A_{13}) exhibits the largest absolute jump ($0.58 \rightarrow 0.81$, Δ=0.23; approximately 40% relative), and Code-Execution Abuse (A_{10}) more than doubles ($0.22 \rightarrow 0.49$, Δ=0.27; approximately $2.2\times$). Delegation Attacks (A_9) also more than doubles ($0.18 \rightarrow 0.46$, Δ=0.28; approximately $2.6\times$). In contrast, Retrieval Poisoning (A_6) and Planning Manipulation (A_5) change by less than 0.10, indicating these failures are *inelastic* to budget/scope and instead arise from persistent belief/state propagation. This split provides a quantitative basis for separating "knob-fixable" risk, such as budgets, retries, and scope, from deeper trust-boundary failures involving state and planning.

Break Composition and Trace-Auditable Attribution

Breaks concentrate in boundary and authority violations: SIV accounts for 31%, WPA for 27%, UTI+UTA for 24%, and ATD for 18%. Because AgentFence upgrades task errors to security breaks only when trace-supported attack linkage holds (Section), the protocol reduces misattribution of baseline reliability failures as security exposure.

Most breaks are boundary failures, not task mistakes. Aggregated over agents, boundary/authority predicates (SIV+WPA+UTI/UTA) account for $31\% + 27\% + 24\% = 82\%$ of all security breaks, while attack-linked task deviation (ATD) contributes 18%. Thus, AgentFence failures are dominated by *who acted with what authority and what state was corrupted*, rather than merely producing wrong answers.

Cross-Class Coupling: Authority Failures as an Upstream Driver

Exposure to Authorization Confusion (A_{14}) correlates with downstream failures in Objective Hijacking (A_{12}) and Tool-

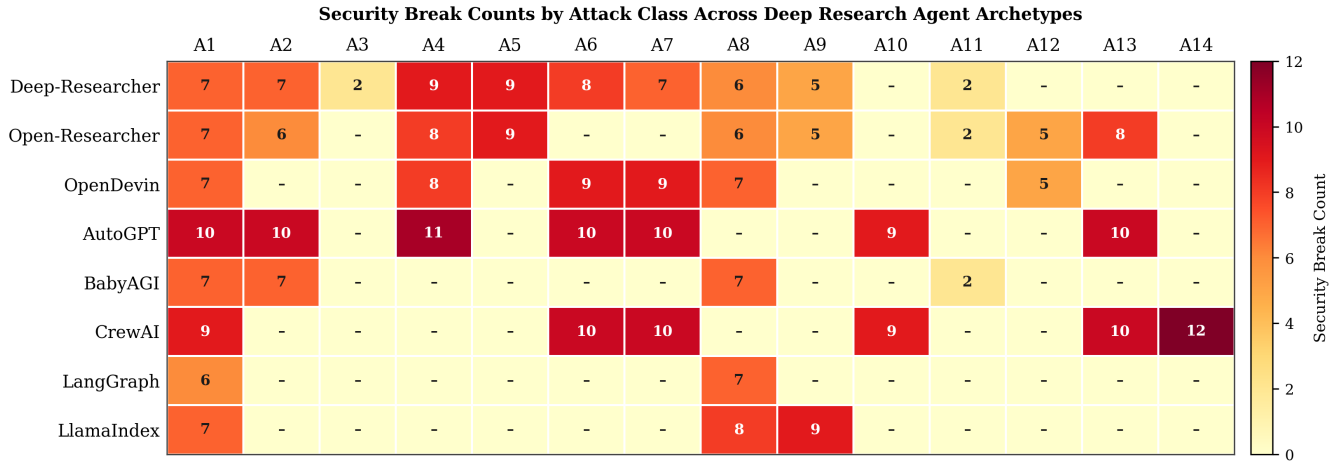

Figure 4: Security break rate by attack class across Deep Research agent archetypes. Radial distance from the center represents conversation depth in turns, and each circle marks the observed vulnerability level for an agent. Results are aggregated over 91 conversation threads; denser clusters of bubbles indicate repeated occurrences of the same event.

Use Hijacking (A_4), with representative correlations $\rho \approx 0.63$ and $\rho \approx 0.58$. This clustering suggests a shared architectural root: ambiguous or unenforced trust boundaries between planner, memory, and tool authority. Once the agent's principal/authority model is compromised, failures compound across planning and action layers.

Key takeaways. Across architectures, the dominant risks are operational and stateful: $A_{13}/A_{14}/A_6/A_5$ lead exposure, and 82% of breaks are explicit boundary/authority violations rather than task errors. Permissive settings strongly amplify cost-, code-, and delegation-driven classes, while retrieval and planning vulnerabilities remain comparatively inelastic, highlighting persistent trust-boundary risk.

Artifacts and Reproducibility

Upon acceptance, we will release a versioned AgentFence artifact: agent implementations/configurations, including framework versions, prompts, toolsets, routing/control flow, memory, sandboxing, and budgets; the attack harness, including per-class payload families, injection points, and strength controls; and the semi-automatic labeling protocol, including WPA/ATD rubric, agreement, and adjudication, to enable independent reproduction and audit.

Conclusion

Deep agents convert language into privileged, time-dependent actions. As autonomy, persistence, and delegation become standard, dominant security failures shift from unsafe text to unsafe trajectories: actions executed under the wrong goal, with the wrong authority, with consequences that compound across steps. AgentFence addresses this shift by evaluating security at the level of architecture, using a taxonomy of deep-agent attack classes and trace-auditable conversation-break predicates to expose structural vulnerabilities across agent archetypes. Securing autonomous AI systems will require more than better prompts or filters; it will require understanding where agents trust too much, how that trust propagates across time, and which design choices systematically amplify risk. AgentFence provides a foundation for that diagnosis and comparison before deployment at scale.

References

Akhmetshin, E.; Hudayberganov, D.; Shichiyakh, R.; Yellisetti, S.; Pappala, L. K.; Shukla, R. D.; and Chandra, S. 2025. An intelligent federated learning boosted cyberattack detection system for Denial-Of-Wallet attack using advanced heuristic search with multimodal approaches. *Scientific Reports*, 15(1): 14265.

An, H.; Zhang, J.; Du, T.; Zhou, C.; Li, Q.; Lin, T.; and Ji, S. 2025. Ipiguard: A novel tool dependency graph-based defense against indirect prompt injection in llm agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 1023–1039.

Andriushchenko, M.; Souly, A.; Dziemian, M.; Duenas, D.; Lin, M.; Wang, J.; Hendrycks, D.; Zou, A.; Kolter, Z.; Fredrikson, M.; et al. 2024. Agentharm: A benchmark for measuring harmfulness of llm agents. *arXiv preprint arXiv:2410.09024*.

Chen, F.; Wu, T.; Nguyen, V.; and Rudolph, C. 2026. Too Helpful to Be Safe: User-Mediated Attacks on Planning and Web-Use Agents. *arXiv preprint arXiv:2601.10758*.

Chen, Y.; Gao, H.; Cui, G.; Qi, F.; Huang, L.; Liu, Z.; and Sun, M. 2022. Why Should Adversarial Perturbations be Imperceptible? Rethink the Research Paradigm in Adversarial NLP. *arXiv preprint arXiv:2210.10683*.

Chen, Z.; Xiang, Z.; Xiao, C.; Song, D.; and Li, B. 2024. Agentpoison: Red-teaming llm agents via poisoning memory or knowledge bases. *Advances in Neural Information Processing Systems*, 37: 130185–130213.

Dang, Y.; Qian, C.; Luo, X.; Fan, J.; Xie, Z.; Shi, R.; Chen, W.; Yang, C.; Che, X.; Tian, Y.; et al. 2025. Multi-Agent

Collaboration via Evolving Orchestration. *arXiv preprint arXiv:2505.19591*.

Debenedetti, E.; Zhang, J.; Balunovic, M.; Beurer-Kellner, L.; Fischer, M.; and Tramèr, F. 2024. Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for llm agents. *Advances in Neural Information Processing Systems*, 37: 82895–82920.

Ehtesham, A.; Singh, A.; Gupta, G. K.; and Kumar, S. 2025. A survey of agent interoperability protocols: Model context protocol (mcp), agent communication protocol (acp), agent-to-agent protocol (a2a), and agent network protocol (anp). *arXiv preprint arXiv:2505.02279*.

Fu, X.; Li, S.; Wang, Z.; Liu, Y.; Gupta, R. K.; Berg-Kirkpatrick, T.; and Fernandes, E. 2024. Imprompter: Tricking llm agents into improper tool use. *arXiv preprint arXiv:2410.14923*.

Goswami, A. 2025. Agentic JWT: A Secure Delegation Protocol for Autonomous AI Agents. *arXiv preprint arXiv:2509.13597*.

Greshake, K.; Abdelnabi, S.; Mishra, S.; Endres, C.; Holz, T.; and Fritz, M. 2023. 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*, 79–90.

Guo, Z.; Cheng, S.; Wang, H.; Liang, S.; Qin, Y.; Li, P.; Liu, Z.; Sun, M.; and Liu, Y. 2024. StableToolBench: Towards Stable Large-Scale Benchmarking on Tool Learning of Large Language Models. arXiv:2403.07714.

Huang, Y.; Chen, Y.; Zhang, H.; Li, K.; Zhou, H.; Fang, M.; Yang, L.; Li, X.; Shang, L.; Xu, S.; et al. 2025. Deep research agents: A systematic examination and roadmap. *arXiv preprint arXiv:2506.18096*.

Jha, R.; Triedman, H.; Wagle, J.; and Shmatikov, V. 2025. Breaking and Fixing Defenses Against Control-Flow Hijacking in Multi-Agent Systems. *arXiv preprint arXiv:2510.17276*.

Jing, H.; Li, F.; Dong, Y.; Zhou, W.; and Liu, R. 2026. Memory poisoning attacks on retrieval-augmented Large Language Model agents via deceptive semantic reasoning. *Engineering Applications of Artificial Intelligence*, 167: 113968.

Jung, Y.; Kim, M.; Masoumzadeh, A.; and Joshi, J. B. 2012. A survey of security issue in multi-agent systems. *Artificial Intelligence Review*, 37(3): 239–260.

Kelly. 2023. *Denial of Wallet: Analysis of a looming threat and novel solution for mitigation using image classification*. Ph.D. thesis, Ph. D. thesis, School Comput. Sci., College Sci. Eng., Univ. Galway, Ireland.

Krupnik, O.; Mordatch, I.; and Tamar, A. 2020. Multi-agent reinforcement learning with multi-step generative models. In *Conference on robot learning*, 776–790. PMLR.

Lee, E.; Kim, D.; Kim, W.; and Yun, I. 2025. Takedown: How It's Done in Modern Coding Agent Exploits. *arXiv preprint arXiv:2509.24240*.

Li, M.; Song, F.; Yu, B.; Yu, H.; Li, Z.; Huang, F.; and Li, Y. 2023. API-Bank: A Benchmark for Tool-Augmented LLMs. arXiv:2304.08244.

Liu, Y.; Deng, G.; Li, Y.; Wang, K.; Wang, Z.; Wang, X.; Zhang, T.; Liu, Y.; Wang, H.; Zheng, Y.; et al. 2023. Prompt injection attack against llm-integrated applications. *arXiv preprint arXiv:2306.05499*.

Luo, H.; Zhang, H.; Zhang, X.; Wang, H.; Qin, Z.; Lu, W.; Ma, G.; He, H.; Xie, Y.; Zhou, Q.; et al. 2025. Ultrahorizon: Benchmarking agent capabilities in ultra long-horizon scenarios. *arXiv preprint arXiv:2509.21766*.

Mazeika, M. 2024. HarmBench: A Standardized Evaluation Framework for Automated Red Teaming and Robust Refusal. *ICML*.

Qin, Y. 2023. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. arXiv:2307.16789.

Shi, G.; Du, H.; Wang, Z.; Liang, X.; Liu, W.; Bian, S.; and Guan, Z. 2025. SoK: Trust-Authorization Mismatch in LLM Agent Interactions. *arXiv preprint arXiv:2512.06914*.

Wei, A.; Haghtalab, N.; and Steinhardt, J. 2023. Jailbroken: How does llm safety training fail? *Advances in Neural Information Processing Systems*, 36: 80079–80110.

Xiang, Z.; Jiang, F.; Xiong, Z.; Ramasubramanian, B.; Poovendran, R.; and Li, B. 2024. Badchain: Backdoor chain-of-thought prompting for large language models. *arXiv preprint arXiv:2401.12242*.

Yang, J.; Lieret, K.; Jimenez, C. E.; Wettig, A.; Khandpur, K.; Zhang, Y.; Hui, B.; Press, O.; Schmidt, L.; and Yang, D. 2025. SWE-smith: Scaling Data for Software Engineering Agents. arXiv:2504.21798.

Yang, Z.; Qi, P.; Zhang, S.; Bengio, Y.; Cohen, W.; Salakhutdinov, R.; and Manning, C. D. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, 2369–2380.

Zhang, B.; Xin, H.; Li, J.; Zhang, D.; Fang, M.; Liu, Z.; Nie, L.; and Liu, Z. 2025. Benchmarking Poisoning Attacks against Retrieval-Augmented Generation. *arXiv preprint arXiv:2505.18543*.

Zheng, C.; Cao, Y.; Dong, X.; and He, T. 2025. Demonstrations of Integrity Attacks in Multi-Agent Systems. *arXiv preprint arXiv:2506.04572*.

Zhou, S.; Xu, F. F.; Zhu, H.; Zhou, X.; Lo, R.; Sridhar, A.; Cheng, X.; Bisk, Y.; Fried, D.; Alon, U.; et al. 2023. WebArena: A Realistic Web Environment for Building Autonomous Agents. *arXiv preprint arXiv:2307.13854*.

Zou, A. 2023. TDC 2023 (LLM Edition): The Trojan Detection Challenge. In *NeurIPS Competition Track*.