Isolation as a First-Class Principle for LLM-Agent System Safety: Concepts, Taxonomy, Challenges and Future Directions

Huihao Jing[*], **Wenbin Hu**[*], **Shaojin Chen**, **Haochen Shi**, **Sirui Zhang**, **Hanyu Yang**, **Changxuan Fan**, **Zhongwei Xie**, **Hongyu Luo**, **Wun Yu Chan**, **Wei Fan**, **Haoran Li**[†], **Yangqiu Song**
HKUST, NYU, SWUPL, MODEIO.AI,
hjingaa@connect.ust.hk

Abstract

The capability of LLM agents to function as the "brain" of a system fundamentally expands the scope of analysis beyond a standalone model. Consequently, safety is no longer only about input–output content alignment. It also concerns system behavior and real-world execution outcomes. However, the current literature is fragmented across attack types, applications, and benchmarks. This makes it hard to explain why failures such as prompt injection, tool misuse, and memory poisoning often share the same structural cause, and how they spread through an agent workflow. In this survey, we treat isolation as a first-class principle for LLM-agent system safety. By isolation, we refer to the separation of user inputs, tool access, execution channels, inter-agent communication, and environment-originated context. We organize the literature with a boundary-centric taxonomy of five boundaries: user-agent, agent-tool, agent-execution, agent-agent, and system-environment. This view helps identify where the loss of isolation first occurs, how compromise propagates across boundaries, and which defenses are most relevant at each interface. We also summarize cross-boundary failure paths, discuss open challenges, and outline a research agenda for isolation-by-construction in future agent systems.

1 Introduction

LLM agents are moving quickly from research prototypes to real systems. Recent examples include Codex (OpenAI, 2025), Claude Code (Anthropic, 2025), and OpenClaw (OpenClaw, 2026). Related work on managed agents (Anthropic, 2026) and agent harnesses (Meng et al., 2026) shows the same shift. These systems do more than generate text. They read files, call tools, browse the web, edit state, coordinate with other agents, and take actions over long horizons. As a result, the safety problem is no longer only about unsafe outputs. It is also about how control, capability, memory, and context move through the system.

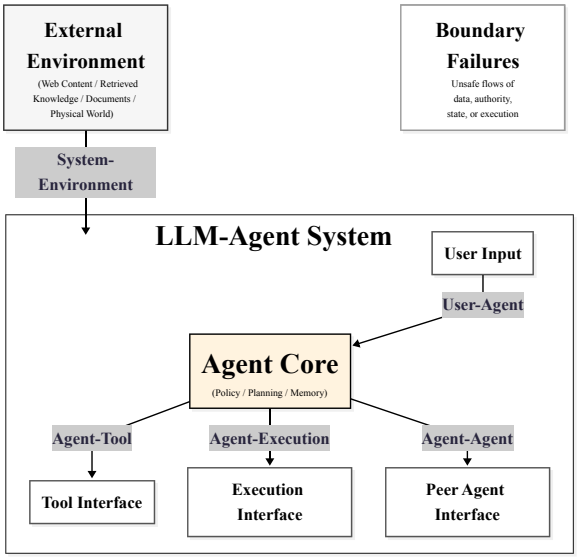

Figure 1: A boundary-centric taxonomy of LLM-agent system safety organized around five isolation boundaries: user-agent, agent-tool, agent-execution, agent-agent, and system-environment.

This change has already shaped both industry practice and recent research. A common direction is to decouple the brain from the hands. Managed agents (Anthropic, 2026), AgentVisor (Ying et al., 2026), privilege-separation proposals (Jacob et al., 2025), CaMeL-style design defenses (Debenedetti et al., 2025), and Parallax (Fokou, 2026) all push in this direction. Different papers use different terms, such as virtualization, privilege separation, typed interfaces, and design-level defense. But the core idea is simple: agent safety improves when system boundaries are explicit and enforced structurally, rather than left to prompt instructions alone.

Despite this progress, the literature remains fragmented. Existing surveys often organize the space by attack type, application domain, or agent capability (Meng et al., 2026; Li et al., 2026; Kim

[*]Equal Contribution
[†]Corresponding author

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 20631–20650
October 24-29, 2026 ©2026 Association for Computational Linguistics

et al., 2026). That view is useful, but it leaves a gap. It does not clearly explain why failures that look different on the surface often share the same cause: loss of isolation at a boundary. It also makes cross-boundary propagation harder to see.

This survey addresses that gap. We treat isolation as a first-class principle for LLM-agent safety and organize the literature around five boundaries: user-agent, agent-tool, agent-execution, agent-agent, and system-environment. Figure 1 gives a compact view of this taxonomy. At the center is the agent core. Around it sit five interfaces where user inputs, tool access, execution channels, inter-agent communication, and environment-originated context may change status. The **user-agent boundary** concerns whether user content remains data or becomes control. The **agent-tool boundary** concerns how external capabilities are accessed. The **agent-execution boundary** concerns the transition from reasoning to action. The **agent-agent boundary** concerns communication and coordination across multiple agents. The **system-environment boundary** concerns how the agent system interacts with external content and state as a whole. These boundaries are distinct, but closely related. Together they provide a cleaner way to explain where compromise begins and how it later propagates. This taxonomy is boundary-centric, but not boundary-exclusive. Many papers naturally touch more than one boundary. We therefore classify papers by their *primary safety boundary*, that is, the point where the loss of isolation first occurs. This rule keeps the taxonomy simple and makes cross-boundary analysis easier.

Our contributions are summarized as follows:

- We propose a boundary-centric taxonomy for LLM-agent safety based on five isolation boundaries, and map attacks/defenses to where isolation first breaks, showing how failures propagate across agent workflows.
- We unify prior work through an isolation-centric view of failure propagation, and outline an isolation-by-construction agenda emphasizing trust separation, scoped capabilities, traceability, and recovery.

2 User-Agent Boundary

2.1 Threat Model and Boundary Definition

The user-agent boundary asks whether an agent can keep user content from becoming privileged control. In a well-isolated system, user input should remain a request, a query, or task data. System and developer instructions should keep higher authority. Failure begins when user-controlled content starts to steer internal policy or behavior, as shown in early prompt-injection and role-confusion studies (Perez and Ribeiro, 2022; Liu et al., 2023b; Toyer et al., 2024; Pedro et al., 2023; Ren, 2024; Hu et al., 2025). This is usually the first exposed interface in an agent system. Before an agent uses tools, acts, or coordinates with peers, it must decide what counts as instruction and what counts as data. If that decision is unstable, later safeguards start from a weak position.

2.2 Direct Prompt Injection and Automated Jailbreaks

Early work showed that natural-language inputs can override hidden prompts (Perez and Ribeiro, 2022; Liu et al., 2023b), leak system instructions (Toyer et al., 2024), and confuse role hierarchy (Pedro et al., 2023). The core issue is simple: a low-privilege source can behave like a high-privilege one. Recent work made this threat much more systematic. Attackers now use adversarial suffixes (Zhu et al., 2023; Liu et al., 2024b), black-box search (Jia et al., 2025b; Sitawarin et al., 2024), proxy-guided optimization (Jiang et al., 2024a), and automated red teaming (Pei et al., 2025) to find failures at scale. Benchmarks such as HarmBench (Mazeika et al., 2024), JailbreakBench (Chao et al., 2024), and JailbreakEval (Ran et al., 2024) also pushed the field toward broader and more reproducible evaluation. Together, these papers show that the user-agent boundary is vulnerable under repeated adaptive pressure.

2.3 From Single-Turn Attacks to Persistent Compromise

More recent papers show that this boundary is not challenged only by single-turn prompts. Multi-turn jailbreaks exploit gradual steering (Cheng et al., 2024b; Russinovich et al., 2025), implicit clues (Ren et al., 2024; Chang et al., 2024), and long-context overload (Upadhayay et al., 2024; Zhou et al., 2024b). Many systems look safer in one-shot evaluation than they are in realistic sessions. The attack surface also expands in multimodal settings. User-provided images can carry visual or typographic instructions that bypass text-oriented safeguards (Liu et al., 2023a; Gong et al., 2025; Wu et al., 2023). This boundary is therefore not only

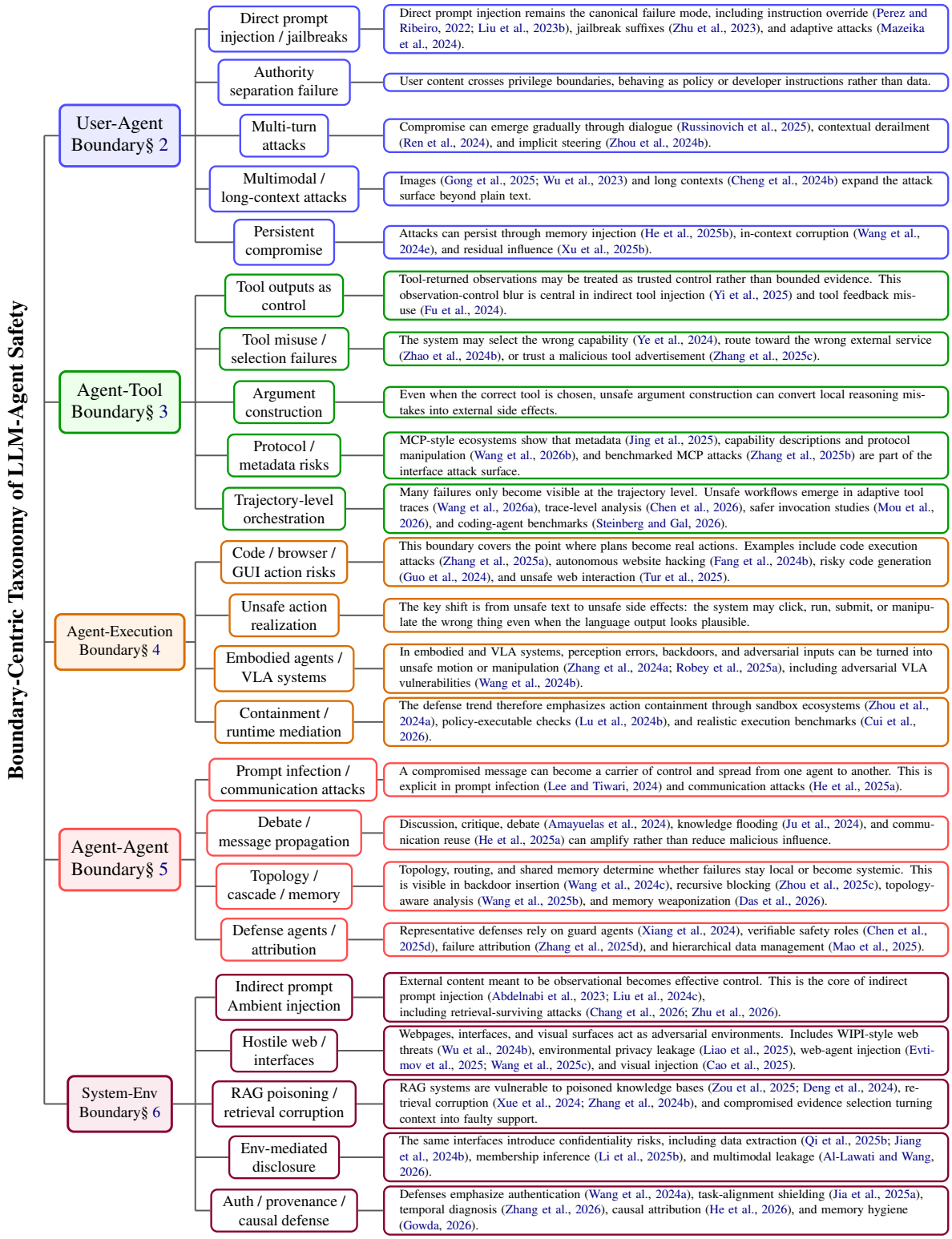

Figure 2: A roadmap-style view of our boundary-centric taxonomy for LLM-agent system safety. Each boundary is organized into representative subtopics and example literature.

about text. It spans multiple input channels. The strongest recent trend is persistence. In-context poisoning (He et al., 2025b), dynamic soft prompting (Wang et al., 2024e), and memory injection (Dong et al., 2025; Xu et al., 2025b) show that user influence can remain after the original interaction ends. Related work on poisoning alignment data (Shao et al., 2024; Zhang et al., 2025e) and harmful fine-tuning (Pathmanathan et al., 2024; Huang et al., 2024a) shows that the same weakness can also enter through post-training pipelines. The problem is no longer only prompt-level override. It becomes

long-term state corruption.

2.4 Defenses, Evaluation, and Future Directions

Defenses at this boundary fall into three broad groups. The first makes authority separation explicit through structured queries (Piet et al., 2024; Chen et al., 2025b), signed prompts (Suo, 2024), and DSL-style interfaces (Sharma et al., 2024). The second hardens the model against jailbreaks through safety classifiers (Kim et al., 2023; Li et al., 2025a), semantic smoothing (Robey et al., 2025b), repetition-based defenses (Ji et al., 2025; Zhang et al., 2024c), and inference-time self-protection (Wang et al., 2025d, 2024d; Lin et al., 2025). The third focuses on repair after degradation, including unlearning (Li et al., 2024b; Zhao et al., 2024a), editing (Lu et al., 2024a; Zhao et al., 2024c), and refusal-boundary control (Xiong et al., 2024; Lu et al., 2025; Gou et al., 2024). Evaluation has also improved. Recent work studies detection with perplexity (Alon and Kamfonas, 2023), refusal-loss signals (Hu et al., 2024a), and geometry-aware methods (Candogan et al., 2025; Yung et al., 2025). Benchmarks such as SG-Bench (Mou et al., 2024), Case-Bench (Sun et al., 2025), and Sorry-Bench (Xie et al., 2024) broaden the test space beyond a few popular jailbreak prompts. Durability studies then ask a harder question: do safeguards still work under longer interaction and stronger adaptation (Qi et al., 2025a; Tamirisa et al., 2025; Chen et al., 2024a; Fan et al., 2026)?

Taken together, this literature suggests that user-agent safety must move beyond short prompts and static refusal scores toward long-session robustness, multimodal authority separation, and recovery from persistent compromise.

3 Agent-Tool Boundary

3.1 Threat Model and Boundary Definition

The agent-tool boundary concerns how an agent accesses external capabilities. In a well-isolated system, tools should extend what the agent can do without taking over how it decides. Tool descriptions, tool outputs, and orchestration logic should remain constrained interfaces rather than hidden control channels. Failure begins when external capabilities are exposed without clear separation between observation, instruction, and execution, as shown by early indirect injection and tool-learning failures (Yi et al., 2025; Ye et al., 2024; Fu et al., 2024).

This boundary matters because tools change the risk profile of an agent system. A standalone model can generate unsafe text. A tool-using agent can search the web, call APIs, read files, write code, or trigger downstream actions. Small control errors at this boundary can therefore turn into capability errors. The key question is not only whether the model reasons correctly, but whether the interface keeps capability use scoped, typed, and resistant to manipulation.

3.2 Tool Outputs, Tool Misuse, and Tool Selection Failures

The basic failure mode at this boundary is that tool-returned content is treated as trusted instruction. Early work on indirect prompt injection showed that hidden instructions in tool outputs can redirect downstream reasoning and action (Yi et al., 2025). Later work made this more concrete. ToolSword and Imprompter show that tool learning can fail at several stages, including interpretation, tool selection, argument generation, and feedback use (Ye et al., 2024; Fu et al., 2024). The result is not just poor task performance. It can also produce confidentiality, integrity, and safety failures. Recent papers push this line further by showing that manipulation can happen even before a tool is called correctly. Attacks on third-party APIs (Zhao et al., 2024b), adversarial tool-calling (Zhang et al., 2025c), and prompt injection against tool selection (Shi et al., 2026) show that the agent can be steered into using the wrong capability, or using the right capability in the wrong way. Tool safety is therefore not one failure mode. The system can fail by choosing the wrong tool, passing unsafe arguments, trusting malicious output, or composing plausible calls into an unsafe workflow.

3.3 Protocol, Metadata, and MCP-Style Risks

A major recent shift is that tool-use risk is moving from content alone to protocol design. In MCP-style ecosystems, the model often sees tool descriptions, capability advertisements, and metadata before it uses the tool itself. MCIP and MPMA show that this layer is already security-critical (Jing et al., 2025; Wang et al., 2026b). Metadata is not neutral from the model's perspective. It can shape preferences, change routing, and bias decisions before real tool output even appears. This is why recent benchmark work on MCP and tool ecosystems matters. MCP Security Bench (Zhang et al., 2025b)

shows that attacks against model context protocol are not an edge case, but a natural extension of tool-mediated prompt injection. ToolSafe (Mou et al., 2026) makes the same point from the defense side: tool invocation needs stronger safety mediation, not just better general alignment. The interface itself has become part of the attack surface. As tool use becomes standardized through protocols and orchestrators, the boundary no longer sits only at the moment of API invocation. It also sits at discovery, ranking, metadata interpretation, and workflow construction. Protocol semantics and capability exposure thus become first-class safety issues.

3.4 Trajectory-Level Risk and Safe Tool Orchestration

Recent work shows that tool attacks are often multi-step. A single call may look harmless, while the full trajectory becomes unsafe. AdapTools and TraceSafe make this clear in multi-step tool workflows (Wang et al., 2026a; Chen et al., 2026). The security target is therefore not only one prompt or one API call, but the full trace of calls, arguments, observations, and intermediate state. This is especially important for code interpreters and coding agents. CIBER and MOSAIC-Bench show that once tools support code execution or complex coding workflows, tool misuse can translate into direct downstream impact much more easily (Ba et al., 2026; Steinberg and Gal, 2026). These systems are close to the execution boundary, but the first loss of isolation often still happens here, when a tool is selected, configured, or trusted too early. Defenses follow the same shift from call-level safety to interface-level safety. Some works harden the protocol layer through contextual integrity (Jing et al., 2025), metadata constraints (Wang et al., 2026b), and benchmark-based auditing (Zhang et al., 2025b). Others focus on safer invocation through least privilege, better routing, and orchestration-aware evaluation (Chen and Cong, 2025; Mou et al., 2026). The broader lesson is simple: safer tool use depends less on the model inferring the right behavior from natural language, and more on interfaces that make capability scope, trust level, and action semantics explicit. This also points to future work on trace-level evaluation, privileged treatment of protocol metadata, and tighter links between tool control and execution control.

4 Agent-Execution Boundary

4.1 Threat Model and Boundary Definition

The agent-execution boundary concerns the point at which internal decisions become real actions. In a well-isolated system, planning and action should remain separate enough that unsafe decisions can still be checked, delayed, or blocked before they create side effects. Failure begins when the system treats model output as ready-to-execute behavior without enough mediation.

This boundary turns control errors into operational impact. A model that produces unsafe text is one kind of problem. An agent that clicks the wrong button, runs unsafe code, submits the wrong form, or issues unsafe physical commands is another, as recent cyber-agent work makes clear (Zhang et al., 2025a; Fang et al., 2024b,a; Guo et al., 2024).

4.2 Code, Browser, and GUI Action Risks

Recent work on cyber and software agents makes this transition clear. Agents can be redirected into malfunction amplification (Zhang et al., 2025a), risky code generation (Fang et al., 2024b), or direct offensive behavior such as website compromise and vulnerability exploitation (Fang et al., 2024a; Guo et al., 2024). The same pattern appears in browser and GUI agents. ST-WebAgentBench (Levy et al., 2024) and SafeArena (Tur et al., 2025) show that web agents can produce unsafe outcomes through clicks, submissions, navigation, and action grounding. Browser-agent work also shows that refusal-trained models remain vulnerable once they act through an interface rather than a chat window (Kumar et al., 2024), while GUI-agent studies show added risk from visual grounding, interface ambiguity, and hidden action consequences (Chen et al., 2025a). Execution safety therefore cannot be reduced to ordinary alignment.

4.3 Embodied Agents and Vision-Language-Action Systems

The execution boundary becomes even sharper in embodied systems, where failures are often costlier and less reversible. BADROBOT (Zhang et al., 2024a), robot jailbreaking (Robey et al., 2025a), and contextual backdoor attacks (Liu et al., 2024a; Jiao et al., 2025) show that benign-looking inputs can induce unsafe physical behavior or persistent actuation failures. Benchmarks such as Embodied Agent Interface (Li et al., 2024a), VIVA (Hu et al.,

2024b), HASARD (Tomilin et al., 2025), HEAL (Chakraborty et al., 2025), and Embodied Red Teaming (Karnik et al., 2024) make this risk more concrete. Work on VLA vulnerabilities further shows that execution can be corrupted through perception, including visual perturbations (Wu et al., 2024a), adversarial patches (Wang et al., 2024b), and action-model weaknesses (Cheng et al., 2024a; Zhou et al., 2025b).

4.4 Containment, Runtime Mediation, and Future Directions

Because failures at this boundary create real side effects, defenses increasingly focus on containment rather than only prevention. Recent work proposes constrained execution (Wang et al., 2025a), zero-trust architectures (Lu et al., 2024b), active defense (Zhou et al., 2024a), and policy-executable safeguards (Maiti, 2026). Evaluation reflects the same shift. SafeArena (Tur et al., 2025), ST-WebAgentBench (Levy et al., 2024), and HWE-Bench (Cui et al., 2026) ask whether the agent remains safe while interacting with real interfaces and workflows. Overall, this boundary suggests that safe agent design must treat execution as a governed interface, not as the automatic continuation of model output.

5 Agent-Agent Boundary

5.1 Threat Model and Boundary Definition

The agent-agent boundary concerns what happens when multiple agents communicate, delegate, debate, and share intermediate state. In a well-isolated system, one agent's message should remain a bounded contribution rather than an unverified control signal for others. Failure begins when inter-agent communication is treated as trusted reasoning by default. Multi-agent systems add channels that do not exist in single-agent settings. Messages can be forwarded, summarized, amplified, and stored in shared memory. As a result, one local failure may become a system-level failure, as shown by prompt-infection, malicious-agent, and propagation studies (Lee and Tiwari, 2024; Wang et al., 2024c; He et al., 2025a; Wang et al., 2025b).

5.2 Prompt Infection and Communication Attacks

The most direct failure mode at this boundary is communication-borne prompt injection. *Prompt Infection* showed that one compromised agent can pass malicious instructions to others, turning ordinary coordination into an attack channel (Lee and Tiwari, 2024). A message is not just information. It can also be a carrier of control. Recent work shows that this problem becomes stronger in realistic collaborative settings. Debate-based attacks (Amayuelas et al., 2024), manipulated-knowledge flooding (Ju et al., 2024), and explicit communication attacks (He et al., 2025a) all show that harmful content can spread through discussion, critique, and consensus formation rather than through a single direct override. CORBA (Zhou et al., 2025c) pushes this further by studying recursive blocking behavior, where one malicious pattern can propagate and suppress useful coordination across the network. The progression is clear: first one compromised message, then repeated propagation, then communication-level cascade.

5.3 Malicious Agents, Topology, and Cascade Failures

Another major line of work asks why some multi-agent systems fail locally while others fail systemically. One answer is topology. Network structure, routing rules, and memory sharing determine how fast compromise travels and how hard it is to contain. Work on malicious-agent resilience (tse Huang et al., 2024), NetSafe (Yu et al., 2024), G-Safeguard (Wang et al., 2025b), and communication-aware multi-agent risks (Hammond et al., 2025) supports this view. Backdoor and memory attacks make this issue more concrete. BadAgent and *Watch Out for Your Agents!* show that malicious behavior can be implanted into agent workflows and activated later through interaction patterns (Wang et al., 2024c; Yang et al., 2024). Trojan Hippo then shows that shared memory is itself a high-risk surface, because poisoned memory can be weaponized for later exfiltration or control (Das et al., 2026). The threat moves from bad messages, to bad agents, to bad shared state. Once memory and topology are involved, rollback becomes much harder than in a single-agent system.

5.4 Defense Agents, Attribution, and Memory Partitioning

Recent defenses increasingly treat coordination as a security problem rather than only an efficiency problem. GuardAgent (Xiang et al., 2024), AutoDefense (Zeng et al., 2024), ShieldAgent (Chen et al., 2025d), and related multi-agent defense pipelines (Hossain et al., 2025) introduce dedicated

safety roles that inspect, verify, or challenge other agents before outputs are accepted. The main idea is simple: not all agents should have equal authority, and safety should be assigned explicit responsibility inside the workflow. Another important trend is attribution. Recent work asks not just whether a multi-agent system failed, but which agent, which message, and which step caused the decisive failure (Zhang et al., 2025d). This matters because containment is difficult without diagnosis. Structural defenses then push one step further. AgentSafe (Mao et al., 2025) uses hierarchical data management to reduce unsafe cross-agent sharing, while G-Safeguard (Wang et al., 2025b) monitors the interaction graph for anomalous propagation patterns. This line of work suggests that memory partitioning, privilege separation, and topology-aware monitoring may be more durable than generic prompt filtering alone.

Overall, this boundary shows that collaboration is not automatically a safeguard. Without isolation, it can become a mechanism for amplification.

6 System-Environment Boundary

6.1 Threat Model and Boundary Definition

The system-environment boundary concerns how an agent reads and reacts to the outside world. In a well-isolated system, webpages, retrieved passages, documents, emails, interface elements, and memory artifacts should remain observations rather than hidden commands. Failure begins when environment-originated content is absorbed into the agent context and then treated as if it had authority.

The environment is often broad and weakly controlled. Once outside content can steer reasoning, retrieval, or action, the outside world becomes part of the control loop, as shown by indirect injection and retrieval-side studies (Abdelnabi et al., 2023; Zverev et al., 2025; Chang et al., 2026; Zhu et al., 2026).

6.2 Indirect Prompt Injection from Ambient Content

The starting point of this literature is indirect prompt injection. Early work showed that malicious instructions can be hidden in webpages, emails, or documents and later executed by an LLM-integrated system when the model reads them (Abdelnabi et al., 2023). Recent work shows that this problem is broader and more persistent than first expected. Automatic attacks (Liu et al., 2024c), multimodal injections (Bagdasaryan et al., 2023; Wu et al., 2024b), and environment-facing attacks on web agents (Liao et al., 2025; Xu et al., 2025a) all show that the environment can steer the agent without going through the nominal user channel. *Overcoming the Retrieval Barrier* (Chang et al., 2026) and *Your Agent is More Brittle Than You Think* (Zhu et al., 2026) further show that indirect injection can survive realistic retrieval pipelines and remain effective even when the injected content appears peripheral. The main challenge is therefore not only detecting bad strings, but preserving source separation after retrieval and context assembly.

6.3 Web-Agent Security and Hostile Interface Environments

This boundary becomes even sharper in web and computer-use agents. Here the environment is not just read. It is also clicked, navigated, copied, and executed against. INJECAGENT (Zhan et al., 2024), WASP (Evtimov et al., 2025), and WebInject (Wang et al., 2025c) show that browser agents are highly vulnerable when malicious prompts are embedded in realistic websites and task flows. Work on WIPI (Wu et al., 2024b), environmental injection for privacy leakage (Liao et al., 2025), and broader analyses of web-agent fragility (Chiang et al., 2025) reaches the same conclusion from slightly different angles: web environments are active adversarial surfaces, not passive information sources. VPI-Bench (Cao et al., 2025) extends this picture to visual prompt injection, showing that computer-use agents can also be misled through interface appearance rather than text alone.

6.4 RAG Poisoning, Retrieval Corruption, and Disclosure Risks

RAG systems expose another major form of system-environment failure. Here the issue is often not explicit instruction, but corrupted evidence. PoisonedRAG (Zou et al., 2025), Pandora (Deng et al., 2024), BADRAG (Xue et al., 2024), and human-imperceptible retrieval poisoning (Zhang et al., 2024b) show that attackers can manipulate knowledge bases or retrieved passages so that the agent reasons from compromised support. Recent work also shows that retrieval systems create disclosure risks in addition to integrity risks. Data extraction attacks (Peng et al., 2024), backdoored retrieval databases (Qi et al., 2025b), agent-based exfiltration attacks (Jiang et al., 2024b), membership in-

ference studies (Li et al., 2025b; Anderson et al., 2025), and multimodal leakage evaluations (Al-Lawati and Wang, 2026) show that the environment can both push corrupted knowledge in and pull private knowledge out. AgentPoison and MEMSAD further show that once retrieval outputs or memory stores are corrupted, the effect may continue across later interactions unless the system actively diagnoses and repairs the damage (Chen et al., 2024b; Gowda, 2026).

6.5 Authentication, Provenance, and Causal Defenses

Recent defenses increasingly try to restore explicit trust structure at this boundary. FATH (Wang et al., 2024a), Spotlighting (Hines et al., 2024), instruction-detection methods (Wen et al., 2025; Chen et al., 2025c), and Task Shield (Jia et al., 2025a) all aim to mark, isolate, or filter environment-originated instructions before they silently become control input. More recent work moves toward stronger causal and provenance-aware defenses. AgentSentry (Zhang et al., 2026) uses temporal causal diagnostics, AttriGuard (He et al., 2026) uses causal attribution, and TrustRAG (Zhou et al., 2025a) emphasizes trustworthy evidence selection and robustness-aware retrieval. The broader lesson is that system-environment safety will likely depend less on stronger generic refusal and more on better source authentication, provenance tracking, memory hygiene, and context attribution.

In short, this boundary suggests that safe agent systems need not only aligned models, but also environment interfaces that preserve isolation by construction.

7 Cross-Boundary Challenges

7.1 How Failures Propagate Across Boundaries

Serious failures often cross boundaries. A common pattern is sequential escalation: user input may first override control at the user-agent boundary, then steer tool use, and finally trigger unsafe execution. Another pattern starts from the environment. A malicious webpage (Abdelnabi et al., 2023), retrieved passage, or poisoned memory item (Chen et al., 2024b; Das et al., 2026) may enter through the system-environment boundary and later propagate into tool calls, agent communication (Lee and Tiwari, 2024; He et al., 2025a), or action traces. In multi-agent systems, the spread can continue because compromised results may be forwarded or stored for reuse (He et al., 2025a; Wang et al., 2025b; Mao et al., 2025). The main unit of analysis is therefore often the full control path, not a single prompt, tool call, or action. This is also why local robustness at one interface does not guarantee system-level safety.

7.2 Isolation-by-Construction

Safer agent systems need interfaces that preserve separation by design. Inputs from users, tools, peer agents, and external content should remain distinguishable. Capability access should be scoped. Propagation paths should be observable through trace-level monitoring (Mou et al., 2026), attribution (Chen et al., 2025d; He et al., 2026), and policy checks (Zhang et al., 2026). Recovery should also be a core requirement once compromise reaches memory or shared state (Mao et al., 2025; Gowda, 2026). In practice, this means that systems should make trust, authority, and capability boundaries explicit throughout the workflow. In short, isolation-by-construction means building interfaces where the boundary remains visible to the system itself.

7.3 Survey Protocol and Positioning

Search and selection. We combined targeted searches over arXiv, ACL Anthology, DBLP, and Google Scholar with backward and forward citation tracing. Queries paired "LLM agent" or "agentic AI" with terms covering prompt injection and jailbreaking, tool and MCP security, code/browser/GUI execution, embodied agents, multi-agent attacks and defenses, retrieval and memory poisoning, and provenance. We considered work available through June 2026. We included peer-reviewed papers and preprints that introduced or evaluated an attack, defense, benchmark, or system design relevant to at least one agent-system interface; duplicates, non-technical commentary, and work without a clear agent-workflow implication were excluded. For each paper, we recorded its first violated boundary and any downstream boundaries implicated by the reported attack or defense. This procedure is intended to make our coverage auditable rather than to claim exhaustiveness in a rapidly changing field.

7.4 Operationalizing Isolation

Let an agent system contain components connected by directed interfaces, each with a contract $\Gamma_i =$

Prior work	Organizing lens	Main scope	Difference from this survey
Meng et al. (2026)	Harness components	Agent infrastructure and engineering patterns	We reinterpret harness interfaces as safety boundaries and classify literature by the first violated contract.
Li et al. (2026)	Deployment risks	System-level security considerations	We add a paper-level taxonomy separating primary isolation loss from downstream propagation.
Kim et al. (2026)	Attack/defense categories	Broad agentic-AI attack and defense landscape	We organize superficially different failures by their shared interface-level cause.
Siu et al. (2026)	Formal security framework	Security abstractions for LLM agents	We provide broad literature mapping with a lightweight operational rule for boundary assignment.

Table 1: Positioning against representative surveys and frameworks. Our goal is complementary: identifying where trust, authority, capability, or state isolation first fails and how that failure propagates.

(T_i, A_i, C_i, S_i) defining admissible trust, authority, capability, and state scopes. A transfer x is an *isolation violation* only when its effective attributes exceed Γ_i; crossing an interface alone is not unsafe. For a causal trace $\tau = (x_1, \ldots, x_n)$, the *primary boundary* is the earliest violated interface, while later affected boundaries are propagation. Violations include user content gaining control authority, tool output exceeding its declared scope, unmediated external side effects, peer contributions exceeding their roles, or external observations losing provenance and becoming trusted instructions or durable state. Thus, a poisoned webpage begins at system–environment, malicious API output at agent–tool, and generated code accessing credentials at agent–execution.

7.5 Boundary-Aware Research Agenda

The literature is uneven: user–agent and system–environment attacks have relatively mature benchmarks, whereas execution mediation, agent–agent isolation, persistent-state recovery, and cross-boundary propagation remain less systematic. Promising directions include long-session and multimodal authority separation; typed, capability-scoped tool interfaces and authenticated protocol metadata; reversible execution with policy checks before high-impact actions; provenance-aware inter-agent communication, memory partitioning, and failure attribution; and retrieval provenance, context quarantine, and memory hygiene. Cascading-failure benchmarks should identify the initial violated boundary, record secondary boundaries, and measure both local compromise and eventual side effects, alongside task utility and overhead. They should also test whether provenance survives transformations such as summarization and retrieval, and whether a system can recover after shared or persistent state is compromised.

7.6 Open Challenges

Open problems remain in evaluation, compositional defense, persistent state, and formalization. Many benchmarks still test single boundaries, while real failures are cross-boundary (Tur et al., 2025; Evtimov et al., 2025; Cui et al., 2026). Defenses may fail after information is transformed downstream. Memory (Gowda, 2026), retrieval caches (Chen et al., 2024b), and shared agent state (Das et al., 2026) further obscure and persist compromises. The field still lacks stable abstractions for authority, trust, and privilege in full workflows (Siu et al., 2026; Li et al., 2026; Kim et al., 2026).

8 Conclusion

In this survey, we argue that LLM-agent safety is best understood through boundaries. Across user input, tool use, execution, inter-agent communication, and environment interaction, failures often share a common pattern: loss of isolation at interfaces. This helps explain where compromise begins and how it propagates. Overall, safer agent systems require not only better alignment, but also clearer trust boundaries, scoped capabilities, runtime mediation, and stronger control of memory and context.

Acknowledgments

The authors of this paper were supported by the National Key Research and Development Program of China (2025YFE0200500), the Beijing-Tianjin-Hebei Collaborative Innovation Special Project (26240201D), the ITSP Platform Research Project (ITS/189/23FP) from ITC of Hong Kong, SAR, China, and the AoE (AoE/E-601/24-N), the CRF (No. C6004-25G), the RIF (R6021-20) and the GRF (16205322) from RGC of Hong Kong, SAR, China. The work described in this paper was conducted in full or in part by Dr. Haoran Li, JC STEM Early Career Research Fellow, supported by The Hong Kong Jockey Club Charities Trust.

Limitations

This survey has several limitations. First, the field is moving quickly, so any taxonomy can become incomplete as new systems and attacks appear. Second, our taxonomy is boundary-centric by design. It is useful for isolation failures, but it is not the only valid way to organize the literature. Third, many papers are naturally cross-boundary. We classify them by the boundary where the decisive loss of isolation first occurs, but this still requires judgment. Finally, the literature is uneven across boundaries, so some parts of the survey are denser than others.

Ethical Considerations

All authors of this paper affirm their adherence to the ACM Code of Ethics and the ACL Code of Conduct. This survey is intended to support safer research and system design for LLM agents.

At the same time, organizing the attack and defense landscape of agent systems has a dual-use aspect. We therefore focus on high-level mechanisms and design implications rather than operational attack instructions.

References

Sahar Abdelnabi, Kai Greshake, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. 2023. Not What You've Signed Up For: Compromising real-world llm-integrated applications with indirect prompt injection. In *AISec@CCS*, pages 79–90.

Ali Al-Lawati and Suhang Wang. 2026. Do multimodal rag systems leak data? a comprehensive evaluation of membership inference and image caption retrieval attacks. *Preprint*, arXiv:2601.17644.

Gabriel Alon and Michael Kamfonas. 2023. Detecting language model attacks with perplexity. *CoRR*.

Alfonso Amayuelas, Xianjun Yang, Antonis Antoniades, Wenyue Hua, Liangming Pan, and William Yang Wang. 2024. MultiAgent Collaboration Attack: Investigating adversarial attacks in large language model collaborations via debate. In *EMNLP (Findings)*, pages 6929–6948.

Maya Anderson, Guy Amit, and Abigail Goldsteen. 2025. Is my data in your retrieval database? membership inference attacks against retrieval augmented generation. In *ICISSP (2)*, pages 474–485.

Anthropic. 2025. Claude code: Best practices for agentic coding. `https://www.anthropic.com/engineering/claude-code-best-practices`.

Anthropic. 2026. Scaling managed agents: Decoupling the brain from the hands. `https://www.anthropic.com/engineering/managed-agents`.

Lei Ba, Qinbin Li, and Songze Li. 2026. CIBER: A comprehensive benchmark for security evaluation of code interpreter agents. *CoRR*.

Eugene Bagdasaryan, Tsung-Yin Hsieh, Ben Nassi, and Vitaly Shmatikov. 2023. (ab)using images and sounds for indirect instruction injection in multimodal llms. *CoRR*.

Leyla Naz Candogan, Yongtao Wu, Elías Abad-Rocamora, Grigorios Chrysos, and Volkan Cevher. 2025. Single-pass detection of jailbreaking input in large language models. *Trans. Mach. Learn. Res.*

Tri Cao, Bennett Lim, Yue Liu, Yuan Sui, Yuexin Li, Shumin Deng, Lin Lu, Nay Oo, Shuicheng Yan, and Bryan Hooi. 2025. VPI-Bench: Visual prompt injection attacks for computer-use agents. *CoRR*.

Trishna Chakraborty, Udita Ghosh, Xiaopan Zhang, Fahim Faisal Niloy, Yue Dong, Jiachen Li, Amit Roy-Chowdhury, and Chengyu Song. 2025. HEAL: An empirical study on hallucinations in embodied agents driven by large language models. In *EMNLP (Findings)*, pages 21226–21243.

Hongyan Chang, Ergute Bao, Xinjian Luo, and Ting Yu. 2026. Overcoming the Retrieval Barrier: Indirect prompt injection in the wild for llm systems. *CoRR*.

Zhiyuan Chang, Mingyang Li, Yi Liu, Junjie Wang, Qing Wang, and Yang Liu. 2024. Play Guessing Game with LLM: Indirect jailbreak attack with implicit clues. In *ACL (Findings)*, pages 5135–5147.

Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J. Pappas, Florian Tramèr, Hamed Hassani, and Eric Wong. 2024. JailbreakBench: An open robustness benchmark for jailbreaking large language models. In *NeurIPS*.

Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J. Pappas, and Eric Wong. 2025. Jailbreaking black box large language models in twenty queries. In *SaTML*, pages 23–42.

Chaoran Chen, Zhiping Zhang, Bingcan Guo, Shang Ma, Ibrahim Khalilov, Simret Araya Gebreegziabher, Yanfang Ye, Ziang Xiao, Yaxing Yao, Tianshi Li, and Toby Jia-Jun Li. 2025a. The Obvious Invisible Threat: Llm-powered GUI agents' vulnerability to fine-print injections. *CoRR*, abs/2504.11281.

Jizhou Chen and Samuel Lee Cong. 2025. AgentGuard: Repurposing agentic orchestrator for safety evaluation of tool orchestration. *CoRR*.

Kexin Chen, Yi Liu, Dongxia Wang, Jiaying Chen, and Wenhai Wang. 2024a. Characterizing and evaluating the reliability of llms against jailbreak attacks. *CoRR*.

Sizhe Chen, Julien Piet, Chawin Sitawarin, and David A. Wagner. 2025b. StruQ: Defending against prompt injection with structured queries. In *USENIX Security Symposium*, pages 2383–2400.

Yen-Shan Chen, Sian-Yao Huang, Cheng-Lin Yang, and Yun-Nung Chen. 2026. TraceSafe: A systematic assessment of llm guardrails on multi-step tool-calling trajectories. *CoRR*.

Yulin Chen, Haoran Li, Yuan Sui, Yufei He, Yue Liu, Yangqiu Song, and Bryan Hooi. 2025c. Can indirect prompt injection attacks be detected and removed? In *ACL (1)*, pages 18189–18206.

Zhaorun Chen, Mintong Kang, and Bo Li. 2025d. ShieldAgent: Shielding agents via verifiable safety policy reasoning. In *ICML*.

Zhaorun Chen, Zhen Xiang, Chaowei Xiao, Dawn Song, and Bo Li. 2024b. AgentPoison: Red-teaming llm agents via poisoning memory or knowledge bases. In *NeurIPS*.

Hao Cheng, Erjia Xiao, Chengyuan Yu, Zhao Yao, Jiahang Cao, Qiang Zhang, Jiaxu Wang, Mengshu Sun, Kaidi Xu, Jindong Gu, and Renjing Xu. 2024a. Manipulation Facing Threats: Evaluating physical vulnerabilities in end-to-end vision language action models. *CoRR*.

Yixin Cheng, Markos Georgopoulos, Volkan Cevher, and Grigorios G. Chrysos. 2024b. Leveraging the context through multi-round interactions for jailbreaking attacks. *CoRR*.

Jeffrey Yang Fan Chiang, Seungjae Lee, Jia-Bin Huang, Furong Huang, and Yizheng Chen. 2025. Why are web ai agents more vulnerable than standalone llms? a security analysis. *CoRR*.

Fan Cui, Hongyuan Hou, Zizhang Luo, Chenyun Yin, and Yun Liang. 2026. HWE-Bench: Benchmarking llm agents on real-world hardware bug repair tasks. *Preprint*, arXiv:2604.14709.

Debeshee Das, Julien Piet, Darya Kaviani, Luca Beurer-Kellner, Florian Tramèr, and David Wagner. 2026. Trojan hippo: Weaponizing agent memory for data exfiltration. *Preprint*, arXiv:2605.01970.

Edoardo Debenedetti, Ilia Shumailov, Tianqi Fan, Jamie Hayes, Nicholas Carlini, Daniel Fabian, Christoph Kern, Chongyang Shi, Andreas Terzis, and Florian Tramèr. 2025. Defeating prompt injections by design. *CoRR*.

Gelei Deng, Yi Liu, Kailong Wang, Yuekang Li, Tianwei Zhang, and Yang Liu. 2024. Pandora: Jailbreak gpts by retrieval augmented generation poisoning. *CoRR*.

Shen Dong, Shaochen Xu, Pengfei He, Yige Li, Jiliang Tang, Tianming Liu, Hui Liu, and Zhen Xiang. 2025. A practical memory injection attack against llm agents. *CoRR*.

Ivan Evtimov, Arman Zharmagambetov, Aaron Grattafiori, Chuan Guo, and Kamalika Chaudhuri. 2025. WASP: Benchmarking web agent security against prompt injection attacks. *CoRR*.

Changxuan Fan, Xi Yang, Yueyuan Zheng, Bin Zhou, Yuanping Wang, Wenbin Hu, Huihao Jing, Ki Sen Hung, Dazhao Du, Haoran Li, Janet Hui-wen Hsiao, and Yangqiu Song. 2026. GrandGuard: Taxonomy, benchmark, and safeguards for elderly-chatbot interaction safety. In *ACL (Findings)*, pages 22213–22248. Association for Computational Linguistics.

Richard Fang, Rohan Bindu, Akul Gupta, and Daniel Kang. 2024a. Llm agents can autonomously exploit one-day vulnerabilities. *CoRR*.

Richard Fang, Rohan Bindu, Akul Gupta, Qiusi Zhan, and Daniel Kang. 2024b. Llm agents can autonomously hack websites. *CoRR*.

Joel Fokou. 2026. Parallax: Why ai agents that think must never act. *Preprint*, arXiv:2604.12986.

Xiaohan Fu, Shuheng Li, Zihan Wang, Yihao Liu, Rajesh K. Gupta, Taylor Berg-Kirkpatrick, and Earlence Fernandes. 2024. Imprompter: Tricking llm agents into improper tool use. *CoRR*.

Yichen Gong, Delong Ran, Jinyuan Liu, Conglei Wang, Tianshuo Cong, Anyu Wang, Sisi Duan, and Xiaoyun Wang. 2025. FigStep: Jailbreaking large vision-language models via typographic visual prompts. In *AAAI*, pages 23951–23959.

Yunhao Gou, Kai Chen, Zhili Liu, Lanqing Hong, Hang Xu, Zhenguo Li, Dit-Yan Yeung, James T. Kwok, and Yu Zhang. 2024. Eyes Closed, Safety on: Protecting multimodal llms via image-to-text transformation. In *ECCV (17)*, pages 388–404.

Ishrith Gowda. 2026. MEMSAD: Gradient-coupled anomaly detection for memory poisoning in retrieval-augmented agents. *Preprint*, arXiv:2605.03482.

Chengquan Guo, Xun Liu, Chulin Xie, Andy Zhou, Yi Zeng, Zinan Lin, Dawn Song, and Bo Li. 2024. RedCode: Risky code execution and generation benchmark for code agents. In *NeurIPS*.

Lewis Hammond, Alan Chan, Jesse Clifton, Jason Hoelscher-Obermaier, Akbir Khan, Euan McLean, Chandler Smith, Wolfram Barfuss, Jakob N. Foerster, Tomas Gavenciak, The Anh Han, Edward Hughes, Vojtech Kovarík, Jan Kulveit, Joel Z. Leibo, Caspar Oesterheld, Christian Schröder de Witt, Nisarg Shah, Michael P. Wellman, and 25 others. 2025. Multi-agent risks from advanced ai. *CoRR*.

Pengfei He, Yuping Lin, Shen Dong, Han Xu, Yue Xing, and Hui Liu. 2025a. Red-teaming llm multi-agent systems via communication attacks. In *ACL (Findings)*, pages 6726–6747.

Pengfei He, Han Xu, Yue Xing, Hui Liu, Makoto Yamada, and Jiliang Tang. 2025b. Data poisoning for in-context learning. In *NAACL (Findings)*, pages 1680–1700.

Yu He, Haozhe Zhu, Yiming Li, Shuo Shao, Hongwei Yao, Zhihao Liu, and Zhan Qin. 2026. AttriGuard: Defeating indirect prompt injection in LLM agents via causal attribution of tool invocations. *CoRR*, abs/2603.10749.

Keegan Hines, Gary Lopez, Matthew Hall, Federico Zarfati, Yonatan Zunger, and Emre Kiciman. 2024. Defending against indirect prompt injection attacks with spotlighting. In *CAMLIS*, pages 48–62.

S. M. Asif Hossain, Ruksat Khan Shayoni, Mohd Ruhul Ameen, Akif Islam, Muhammad Firoz Mridha, and Jungpil Shin. 2025. A multi-agent llm defense pipeline against prompt injection attacks. *CoRR*.

Wenbin Hu, Haoran Li, Huihao Jing, Qi Hu, Ziqian Zeng, Sirui Han, Heli Xu, Tianshu Chu, Peizhao Hu, and Yangqiu Song. 2025. Context reasoner: Incentivizing reasoning capability for contextualized privacy and safety compliance via reinforcement learning. In *EMNLP*, pages 865–883. Association for Computational Linguistics.

Xiaomeng Hu, Pin-Yu Chen, and Tsung-Yi Ho. 2024a. Gradient Cuff: Detecting jailbreak attacks on large language models by exploring refusal loss landscapes. In *NeurIPS*.

Zhe Hu, Yixiao Ren, Jing Li, and Yu Yin. 2024b. VIVA: A benchmark for vision-grounded decision-making with human values. In *EMNLP*, pages 2294–2311.

Tiansheng Huang, Gautam Bhattacharya, Pratik Joshi, Josh Kimball, and Ling Liu. 2024a. Antidote: Post-fine-tuning safety alignment for large language models against harmful fine-tuning. *CoRR*.

Yangsibo Huang, Samyak Gupta, Mengzhou Xia, Kai Li, and Danqi Chen. 2024b. Catastrophic jailbreak of open-source llms via exploiting generation. In *ICLR*.

Dennis Jacob, Emad Alghamdi, Zhanhao Hu, Basel Alomair, and David A. Wagner. 2025. Better privilege separation for agents by restricting data types. *CoRR*.

Jiabao Ji, Bairu Hou, Alexander Robey, George J. Pappas, Hamed Hassani, Yang Zhang, Eric Wong, and Shiyu Chang. 2025. Defending large language models against jailbreak attacks via semantic smoothing. In *IJCNLP-AACL (Short Papers)*, pages 7–40.

Feiran Jia, Tong Wu, Xin Qin, and Anna Cinzia Squicciarini. 2025a. The Task Shield: Enforcing task alignment to defend against indirect prompt injection in llm agents. In *ACL (1)*, pages 29680–29697.

Xiaojun Jia, Tianyu Pang, Chao Du, Yihao Huang, Jindong Gu, Yang Liu, Xiaochun Cao, and Min Lin. 2025b. Improved techniques for optimization-based jailbreaking on large language models. In *ICLR*.

Bojian Jiang, Yi Jing, Tianhao Shen, Qing Yang, and Deyi Xiong. 2024a. DART: Deep adversarial automated red teaming for llm safety. *CoRR*.

Changyue Jiang, Xudong Pan, Geng Hong, Chenfu Bao, and Min Yang. 2024b. RAG-Thief: Scalable extraction of private data from retrieval-augmented generation applications with agent-based attacks. *CoRR*.

Ruochen Jiao, Shaoyuan Xie, Justin Yue, Takami Sato, Lixu Wang, Yixuan Wang, Qi Alfred Chen, and Qi Zhu. 2025. Can we trust embodied agents? exploring backdoor attacks against embodied llm-based decision-making systems. In *ICLR*.

Huihao Jing, Haoran Li, Wenbin Hu, Qi Hu, Heli Xu, Tianshu Chu, Peizhao Hu, and Yangqiu Song. 2025. MCIP: Protecting mcp safety via model contextual integrity protocol. In *EMNLP*, pages 1177–1194.

Tianjie Ju, Yiting Wang, Xinbei Ma, Pengzhou Cheng, Haodong Zhao, Yulong Wang, Lifeng Liu, Jian Xie, Zhuosheng Zhang, and Gongshen Liu. 2024. Flooding spread of manipulated knowledge in llm-based multi-agent communities. *CoRR*.

Sathwik Karnik, Zhang-Wei Hong, Nishant Abhangi, Yen-Chen Lin, Tsun-Hsuan Wang, and Pulkit Agrawal. 2024. Embodied red teaming for auditing robotic foundation models. *CoRR*.

Jinhwa Kim, Ali Derakhshan, and Ian G. Harris. 2023. Robust Safety Classifier for Large Language Models: Adversarial prompt shield. *CoRR*.

Juhee Kim, Xiaoyuan Liu, Zhun Wang, Shi Qiu, Bo Li, Wenbo Guo, and Dawn Song. 2026. The attack and defense landscape of agentic AI: A comprehensive survey. *CoRR*, abs/2603.11088.

Priyanshu Kumar, Elaine Lau, Saranya Vijayakumar, Tu Trinh, Scale Red Team, Elaine T. Chang, Vaughn Robinson, Sean Hendryx, Shuyan Zhou, Matt Fredrikson, Summer Yue, and Zifan Wang. 2024. Refusal-trained llms are easily jailbroken as browser agents. *CoRR*.

Donghyun Lee and Mo Tiwari. 2024. Prompt Infection: Llm-to-llm prompt injection within multi-agent systems. *CoRR*.

Ido Levy, Ben Wiesel, Sami Marreed, Alon Oved, Avi Yaeli, and Segev Shlomov. 2024. ST-WebAgentBench: A benchmark for evaluating safety and trustworthiness in web agents. *CoRR*.

Haoran Li, Yulin Chen, Jingru Zeng, Hao Peng, Huihao Jing, Wenbin Hu, Xi Yang, Ziqian Zeng, Sirui Han, and Yangqiu Song. 2025a. GSPR: aligning LLM safeguards as generalizable safety policy reasoners. *CoRR*, abs/2509.24418.

Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, Yu Zhou, Sanjana Srivastava, Cem Gokmen, Tony Lee, Li Erran Li, Ruohan Zhang, Weiyu Liu, Percy Liang, Li Fei-Fei, Jiayuan Mao, and Jiajun Wu. 2024a. Embodied Agent Interface: Benchmarking llms for embodied decision making. In *NeurIPS*.

Ninghui Li, Kaiyuan Zhang, Kyle Polley, and Jerry Ma. 2026. Security considerations for artificial intelligence agents. *CoRR*.

Yige Li, Hanxun Huang, Yunhan Zhao, Xingjun Ma, and Jun Sun. 2024b. BackdoorLLM: A comprehensive benchmark for backdoor attacks on large language models. *CoRR*.

Yuying Li, Gaoyang Liu, Chen Wang, and Yang Yang. 2025b. Generating Is Believing: Membership inference attacks against retrieval-augmented generation. In *ICASSP*, pages 1–5.

Zeyi Liao, Lingbo Mo, Chejian Xu, Mintong Kang, Jiawei Zhang, Chaowei Xiao, Yuan Tian, Bo Li, and Huan Sun. 2025. Eia: Environmental injection attack on generalist web agents for privacy leakage. In *ICLR*.

Huawei Lin, Yingjie Lao, Tong Geng, Tan Yu, and Weijie Zhao. 2025. UniGuardian: A unified defense for detecting prompt injection, backdoor attacks and adversarial attacks in large language models. *CoRR*.

Aishan Liu, Yuguang Zhou, Xianglong Liu, Tianyuan Zhang, Siyuan Liang, Jiakai Wang, Yanjun Pu, Tianlin Li, Junqi Zhang, Wenbo Zhou, Qing Guo, and

Dacheng Tao. 2024a. Compromising embodied agents with contextual backdoor attacks. *CoRR*.

Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. 2024b. AutoDAN: Generating stealthy jailbreak prompts on aligned large language models. In *ICLR*.

Xiaogeng Liu, Zhiyuan Yu, Yizhe Zhang, Ning Zhang, and Chaowei Xiao. 2024c. Automatic and universal prompt injection attacks against large language models. *CoRR*.

Xin Liu, Yichen Zhu, Yunshi Lan, Chao Yang, and Yu Qiao. 2023a. Query-relevant images jailbreak large multi-modal models. *CoRR*.

Yi Liu, Gelei Deng, Yuekang Li, Kailong Wang, Tianwei Zhang, Yepang Liu, Haoyu Wang, Yan Zheng, and Yang Liu. 2023b. Prompt injection attack against llm-integrated applications. *CoRR*.

Weikai Lu, Ziqian Zeng, Jianwei Wang, Zhengdong Lu, Zelin Chen, Huiping Zhuang, and Cen Chen. 2024a. Eraser: Jailbreaking defense in large language models via unlearning harmful knowledge. *CoRR*.

Xiaoya Lu, Dongrui Liu, Yi Yu, Luxin Xu, and Jing Shao. 2025. X-Boundary: Establishing exact safety boundary to shield llms from multi-turn jailbreaks without compromising usability. *CoRR*.

Xuancun Lu, Zhengxian Huang, Xinfeng Li, Xiaoyu Ji, and Wenyuan Xu. 2024b. POEX: Policy executable embodied AI jailbreak attacks. *CoRR*, abs/2412.16633.

Saikat Maiti. 2026. Caging the Agents: A zero trust security architecture for autonomous ai in healthcare. *CoRR*.

Junyuan Mao, Fanci Meng, Yifan Duan, Miao Yu, Xiaojun Jia, Junfeng Fang, Yuxuan Liang, Kun Wang, and Qingsong Wen. 2025. AgentSafe: Safeguarding large language model-based multi-agent systems via hierarchical data management. *CoRR*.

Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, David A. Forsyth, and Dan Hendrycks. 2024. HarmBench: A standardized evaluation framework for automated red teaming and robust refusal. In *ICML*, pages 35181–35224.

Qianyu Meng, Yanan Wang, Liyi Chen, Yihang Li, Wei Wu, Wenyuan Jiang, Qimeng Wang, Chengqiang Lu, Yan Gao, Yi Wu, and Yao Hu. 2026. Agent harness for large language model agents: A survey. *Preprints*.

Yutao Mou, Zhangchi Xue, Lijun Li, Peiyang Liu, Shikun Zhang, Wei Ye, and Jing Shao. 2026. ToolSafe: Enhancing tool invocation safety of llm-based agents via proactive step-level guardrail and feedback. *Preprint*, arXiv:2601.10156.

Yutao Mou, Shikun Zhang, and Wei Ye. 2024. SG-Bench: Evaluating llm safety generalization across diverse tasks and prompt types. In *NeurIPS*.

OpenAI. 2025. Introducing codex. `https://openai.com/index/introducing-codex/`.

OpenClaw. 2026. Openclaw. `https://github.com/openclaw/openclaw`.

Pankayaraj Pathmanathan, Souradip Chakraborty, Xiangyu Liu, Yongyuan Liang, and Furong Huang. 2024. Is poisoning a real threat to llm alignment? maybe more so than you think. *CoRR*.

Rodrigo Pedro, Daniel Castro, Paulo Carreira, and Nuno Santos. 2023. From prompt injections to sql injection attacks: How protected is your llm-integrated web application? *CoRR*.

Aihua Pei, Zehua Yang, Shunan Zhu, Ruoxi Cheng, and Ju Jia. 2025. SelfPrompt: Autonomously evaluating llm robustness via domain-constrained knowledge guidelines and refined adversarial prompts. In *COLING*, pages 6840–6854.

Yuefeng Peng, Junda Wang, Hong Yu, and Amir Houmansadr. 2024. Data extraction attacks in retrieval-augmented generation via backdoors. *CoRR*.

Fábio Perez and Ian Ribeiro. 2022. Ignore previous prompt: Attack techniques for language models. *CoRR*.

Julien Piet, Maha Alrashed, Chawin Sitawarin, Sizhe Chen, Zeming Wei, Elizabeth Sun, Basel Alomair, and David A. Wagner. 2024. Jatmo: Prompt injection defense by task-specific finetuning. In *ESORICS (1)*, pages 105–124.

Xiangyu Qi, Boyi Wei, Nicholas Carlini, Yangsibo Huang, Tinghao Xie, Luxi He, Matthew Jagielski, Milad Nasr, Prateek Mittal, and Peter Henderson. 2025a. On evaluating the durability of safeguards for open-weight llms. In *ICLR*.

Zhenting Qi, Hanlin Zhang, Eric P. Xing, Sham M. Kakade, and Himabindu Lakkaraju. 2025b. Follow My Instruction and Spill the Beans: Scalable data extraction from retrieval-augmented generation systems. In *ICLR*.

Delong Ran, Jinyuan Liu, Yichen Gong, Jingyi Zheng, Xinlei He, Tianshuo Cong, and Anyu Wang. 2024. JailbreakEval: An integrated toolkit for evaluating jailbreak attempts against large language models. *CoRR*.

Qibing Ren, Hao Li, Dongrui Liu, Zhanxu Xie, Xiaoya Lu, Yu Qiao, Lei Sha, Junchi Yan, Lizhuang Ma, and Jing Shao. 2024. Derail Yourself: Multi-turn llm jailbreak attack through self-discovered clues. *CoRR*.

Yupeng Ren. 2024. F2A: An innovative approach for prompt injection by utilizing feign security detection agents. *CoRR*.

Alexander Robey, Zachary Ravichandran, Vijay Kumar, Hamed Hassani, and George J. Pappas. 2025a. Jailbreaking llm-controlled robots. In *ICRA*, pages 11948–11956.

Alexander Robey, Eric Wong, Hamed Hassani, and George J. Pappas. 2025b. SmoothLLM: Defending large language models against jailbreaking attacks. *Trans. Mach. Learn. Res.*

Mark Russinovich, Ahmed Salem, and Ronen Eldan. 2025. Great, now write an article about that: The crescendo multi-turn llm jailbreak attack. In *USENIX Security Symposium*, pages 2421–2440.

Zedian Shao, Hongbin Liu, Jaden Mu, and Neil Zhenqiang Gong. 2024. Making llms vulnerable to prompt injection via poisoning alignment. *CoRR*.

Reshabh K. Sharma, Vinayak Gupta, and Dan Grossman. 2024. SPML: A dsl for defending language models against prompt attacks. *CoRR*.

Jiawen Shi, Zenghui Yuan, Guiyao Tie, Pan Zhou, Neil Zhenqiang Gong, and Lichao Sun. 2026. Prompt injection attack to tool selection in llm agents. In *NDSS*.

Chawin Sitawarin, Norman Mu, David A. Wagner, and Alexandre Araujo. 2024. PAL: Proxy-guided black-box attack on large language models. *CoRR*.

Vincent Siu, Jingxuan He, Kyle Montgomery, Zhun Wang, Neil Gong, Chenguang Wang, and Dawn Song. 2026. A framework for formalizing llm agent security. *CoRR*.

Jonathan Steinberg and Oren Gal. 2026. MOSAIC-Bench: Measuring compositional vulnerability induction in coding agents. *Preprint*, arXiv:2605.03952.

Guangzhi Sun, Xiao Zhan, Shutong Feng, Philip C. Woodland, and Jose Such. 2025. CASE-Bench: Context-aware safety benchmark for large language models. In *ICML*.

Xuchen Suo. 2024. Signed-Prompt: A new approach to prevent prompt injection attacks against llm-integrated applications. *CoRR*.

Rishub Tamirisa, Bhrugu Bharathi, Long Phan, Andy Zhou, Alice Gatti, Tarun Suresh, Maxwell Lin, Justin Wang, Rowan Wang, Ron Arel, Andy Zou, Dawn Song, Bo Li, Dan Hendrycks, and Mantas Mazeika. 2025. Tamper-resistant safeguards for open-weight llms. In *ICLR*.

Tristan Tomilin, Meng Fang, and Mykola Pechenizkiy. 2025. HASARD: A benchmark for vision-based safe reinforcement learning in embodied agents. In *ICLR*.

Sam Toyer, Olivia Watkins, Ethan Adrian Mendes, Justin Svegliato, Luke Bailey, Tiffany Wang, Isaac Ong, Karim Elmaaroufi, Pieter Abbeel, Trevor Darrell, Alan Ritter, and Stuart Russell. 2024. Tensor Trust: Interpretable prompt injection attacks from an online game. In *ICLR*.

Jen tse Huang, Jiaxu Zhou, Tailin Jin, Xuhui Zhou, Zixi Chen, Wenxuan Wang, Youliang Yuan, Maarten Sap, and Michael R. Lyu. 2024. On the resilience of multi-agent systems with malicious agents. *CoRR*.

Ada Defne Tur, Nicholas Meade, Xing Han Lù, Alejandra Zambrano, Arkil Patel, Esin Durmus, Spandana Gella, Karolina Stanczak, and Siva Reddy. 2025. SafeArena: Evaluating the safety of autonomous web agents. In *ICML*.

Bibek Upadhayay, Vahid Behzadan, and Amin Karbasi. 2024. Cognitive Overload Attack:prompt injection for long context. *CoRR*.

Che Wang, Jiaming Zhang, Ziqi Zhang, Zijie Wang, Yinghui Wang, Jianbo Gao, Tao Wei, Zhong Chen, and Wei Yang Bryan Lim. 2026a. AdapTools: Adaptive tool-based indirect prompt injection attacks on agentic llms. *CoRR*.

Jiongxiao Wang, Fangzhou Wu, Wendi Li, Jinsheng Pan, G. Edward Suh, Z. Morley Mao, Muhao Chen, and Chaowei Xiao. 2024a. FATH: Authentication-based test-time defense against indirect prompt injection attacks. *CoRR*.

Ning Wang, Zihan Yan, Weiyang Li, Chuan Ma, He Henry Chen, and Tao Xiang. 2025a. Advancing Embodied Agent Security: From safety benchmarks to input moderation. In *IJCAI*, pages 7795–7803.

Shilong Wang, Guibin Zhang, Miao Yu, Guancheng Wan, Fanci Meng, Chongye Guo, Kun Wang, and Yang Wang. 2025b. G-Safeguard: A topology-guided security lens and treatment on llm-based multi-agent systems. In *ACL (1)*, pages 7261–7276.

Taowen Wang, Dongfang Liu, James Chenhao Liang, Wenhao Yang, Qifan Wang, Cheng Han, Jiebo Luo, and Ruixiang Tang. 2024b. Exploring the adversarial vulnerabilities of vision-language-action models in robotics. *CoRR*.

Xilong Wang, John Bloch, Zedian Shao, Yuepeng Hu, Shuyan Zhou, and Neil Zhenqiang Gong. 2025c. WebInject: Prompt injection attack to web agents. In *EMNLP*, pages 2010–2030.

Xunguang Wang, Daoyuan Wu, Zhenlan Ji, Zongjie Li, Pingchuan Ma, Shuai Wang, Yingjiu Li, Yang Liu, Ning Liu, and Juergen Rahmel. 2025d. SelfDefend: Llms can defend themselves against jailbreaking in a practical manner. In *USENIX Security Symposium*, pages 2441–2460.

Yifei Wang, Dizhan Xue, Shengjie Zhang, and Shengsheng Qian. 2024c. BadAgent: Inserting and activating backdoor attacks in llm agents. In *ACL (1)*, pages 9811–9827.

Zezhong Wang, Fangkai Yang, Lu Wang, Pu Zhao, Hongru Wang, Liang Chen, Qingwei Lin, and Kam-Fai Wong. 2024d. SELF-GUARD: Empower the llm to safeguard itself. In *NAACL-HLT*, pages 1648–1668.

Zhepeng Wang, Runxue Bao, Yawen Wu, Jackson Taylor, Cao Xiao, Feng Zheng, Weiwen Jiang, Shangqian Gao, and Yanfu Zhang. 2024e. Unlocking memorization in large language models with dynamic soft prompting. In *EMNLP*, pages 9782–9796.

Zihan Wang, Rui Zhang, Yu Liu, Wenshu Fan, Wenbo Jiang, Qingchuan Zhao, Hongwei Li, and Guowen Xu. 2026b. MPMA: Preference manipulation attack against model context protocol. In *AAAI*, pages 35838–35846.

Zeming Wei, Yifei Wang, and Yisen Wang. 2023. Jailbreak and guard aligned language models with only few in-context demonstrations. *CoRR*.

Tongyu Wen, Chenglong Wang, Xiyuan Yang, Haoyu Tang, Yueqi Xie, Lingjuan Lyu, Zhicheng Dou, and Fangzhao Wu. 2025. Defending against indirect prompt injection by instruction detection. In *EMNLP (Findings)*, pages 19472–19487.

Chen Henry Wu, Jing Yu Koh, Ruslan Salakhutdinov, Daniel Fried, and Aditi Raghunathan. 2024a. Adversarial attacks on multimodal agents. *CoRR*.

Fangzhou Wu, Shutong Wu, Yulong Cao, and Chaowei Xiao. 2024b. WIPI: A new web threat for llm-driven web agents. *CoRR*.

Yuanwei Wu, Xiang Li, Yixin Liu, Pan Zhou, and Lichao Sun. 2023. Jailbreaking gpt-4v via self-adversarial attacks with system prompts. *CoRR*.

Zhen Xiang, Linzhi Zheng, Yanjie Li, Junyuan Hong, Qinbin Li, Han Xie, Jiawei Zhang, Zidi Xiong, Chulin Xie, Carl Yang, Dawn Song, and Bo Li. 2024. GuardAgent: Safeguard llm agents by a guard agent via knowledge-enabled reasoning. *CoRR*.

Tinghao Xie, Xiangyu Qi, Yi Zeng, Yangsibo Huang, Udari Madhushani Sehwag, Kaixuan Huang, Luxi He, Boyi Wei, Dacheng Li, Ying Sheng, Ruoxi Jia, Bo Li, Kai Li, Danqi Chen, Peter Henderson, and Prateek Mittal. 2024. SORRY-Bench: Systematically evaluating large language model safety refusal behaviors. *CoRR*.

Chen Xiong, Xiangyu Qi, Pin-Yu Chen, and Tsung-Yi Ho. 2024. Defensive Prompt Patch: A robust and interpretable defense of llms against jailbreak attacks. *CoRR*.

Chejian Xu, Mintong Kang, Jiawei Zhang, Zeyi Liao, Lingbo Mo, Mengqi Yuan, Huan Sun, and Bo Li. 2025a. AdvAgent: Controllable blackbox red-teaming on web agents. In *ICML*.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025b. A-MEM: Agentic memory for llm agents. *CoRR*.

Jiaqi Xue, Mengxin Zheng, Yebowen Hu, Fei Liu, Xun Chen, and Qian Lou. 2024. BadRAG: Identifying vulnerabilities in retrieval augmented generation of large language models. *CoRR*.

Wenkai Yang, Xiaohan Bi, Yankai Lin, Sishuo Chen, Jie Zhou, and Xu Sun. 2024. Watch out for your agents! investigating backdoor threats to llm-based agents. In *NeurIPS*.

Junjie Ye, Sixian Li, Guanyu Li, Caishuang Huang, Songyang Gao, Yilong Wu, Qi Zhang, Tao Gui, and Xuanjing Huang. 2024. ToolSword: Unveiling safety issues of large language models in tool learning across three stages. In *ACL (1)*, pages 2181–2211.

Jingwei Yi, Yueqi Xie, Bin Zhu, Emre Kiciman, Guangzhong Sun, Xing Xie, and Fangzhao Wu. 2025. Benchmarking and defending against indirect prompt injection attacks on large language models. In *KDD (1)*, pages 1809–1820.

Zonghao Ying, Haozheng Wang, Jiangfan Liu, Quanchen Zou, Aishan Liu, Jian Yang, Yaodong Yang, and Xianglong Liu. 2026. AgentVisor: Defending llm agents against prompt injection via semantic virtualization. *Preprint*, arXiv:2604.24118.

Miao Yu, Shilong Wang, Guibin Zhang, Junyuan Mao, Chenlong Yin, Qijiong Liu, Qingsong Wen, Kun Wang, and Yang Wang. 2024. NetSafe: Exploring the topological safety of multi-agent networks. *CoRR*.

Canaan Yung, Hanxun Huang, Sarah Monazam Erfani, and Christopher Leckie. 2025. CURVALID: Geometrically-guided adversarial prompt detection. *CoRR*.

Yifan Zeng, Yiran Wu, Xiao Zhang, Huazheng Wang, and Qingyun Wu. 2024. AutoDefense: Multi-agent llm defense against jailbreak attacks. *CoRR*.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. 2024. InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In *ACL (Findings)*, pages 10471–10506.

Boyang Zhang, Yicong Tan, Yun Shen, Ahmed Salem, Michael Backes, Savvas Zannettou, and Yang Zhang. 2025a. Breaking Agents: Compromising autonomous llm agents through malfunction amplification. In *EMNLP*, pages 34964–34976.

Dongsen Zhang, Zekun Li, Xu Luo, Xuannan Liu, Peipei Li, and Wenjun Xu. 2025b. MCP Security Bench (MSB): Benchmarking attacks against model context protocol in llm agents. *CoRR*.

Hangtao Zhang, Chenyu Zhu, Xianlong Wang, Ziqi Zhou, Shengshan Hu, and Leo Yu Zhang. 2024a. BadRobot: Jailbreaking llm-based embodied ai in the physical world. *CoRR*.

Quan Zhang, Binqi Zeng, Chijin Zhou, Gwihwan Go, Heyuan Shi, and Yu Jiang. 2024b. Human-imperceptible retrieval poisoning attacks in llm-powered applications. In *SIGSOFT FSE Companion*, pages 502–506.

Rupeng Zhang, Haowei Wang, Junjie Wang, Mingyang Li, Yuekai Huang, Dandan Wang, and Qing Wang. 2025c. From Allies to Adversaries: Manipulating llm tool-calling through adversarial injection. In *NAACL (Long Papers)*, pages 2009–2028.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and Qingyun Wu. 2025d. Which agent causes task failures and when? on automated failure attribution of llm multi-agent systems. In *ICML*.

Tian Zhang, Yiwei Xu, Juan Wang, Keyan Guo, Xiaoyang Xu, Bowen Xiao, Quanlong Guan, Jinlin Fan, Jiawei Liu, Zhiquan Liu, and Hongxin Hu. 2026. AgentSentry: Mitigating indirect prompt injection in LLM agents via temporal causal diagnostics and context purification. *CoRR*, abs/2602.22724.

Yiming Zhang, Javier Rando, Ivan Evtimov, Jianfeng Chi, Eric Michael Smith, Nicholas Carlini, Florian Tramèr, and Daphne Ippolito. 2025e. Persistent pre-training poisoning of llms. In *ICLR*.

Ziyang Zhang, Qizhen Zhang, and Jakob Nicolaus Foerster. 2024c. Parden, can you repeat that? defending against jailbreaks via repetition. In *ICML*, pages 60271–60287.

Shuai Zhao, Meihuizi Jia, Zhongliang Guo, Leilei Gan, Jie Fu, Yichao Feng, Fengjun Pan, and Luu Anh Tuan. 2024a. A Survey of Backdoor Attacks and Defenses on Large Language Models: Implications for security measures. *CoRR*.

Wanru Zhao, Vidit Khazanchi, Haodi Xing, Xuanli He, Qiongkai Xu, and Nicholas Donald Lane. 2024b. Attacks on third-party apis of large language models. *CoRR*.

Wei Zhao, Zhe Li, Yige Li, Ye Zhang, and Jun Sun. 2024c. Defending large language models against jailbreak attacks via layer-specific editing. In *EMNLP (Findings)*, pages 5094–5109.

Huichi Zhou, Kinhei Lee, Zhonghao Zhan, Yue Chen, Zhenhao Li, Zhaoyang Wang, Hamed Haddadi, and Emine Yilmaz. 2025a. TrustRAG: Enhancing robustness and trustworthiness in rag. *CoRR*.

Jiaming Zhou, Ke Ye, Jiayi Liu, Teli Ma, Zifan Wang, Ronghe Qiu, Kun-Yu Lin, Zhilin Zhao, and Junwei Liang. 2025b. Exploring the limits of vision-language-action manipulations in cross-task generalization. *CoRR*.

Xuhui Zhou, Hyunwoo Kim, Faeze Brahman, Liwei Jiang, Hao Zhu, Ximing Lu, Frank Xu, Bill Yuchen Lin, Yejin Choi, Niloofar Mireshghallah, Ronan Le Bras, and Maarten Sap. 2024a. HAICOSYSTEM: An ecosystem for sandboxing safety risks in human-ai interactions. *CoRR*.

Zhenhong Zhou, Zherui Li, Jie Zhang, Yuanhe Zhang, Kun Wang, Yang Liu, and Qing Guo. 2025c. CORBA: Contagious recursive blocking attacks on multi-agent systems based on large language models. *CoRR*.

Zhenhong Zhou, Jiuyang Xiang, Haopeng Chen, Quan Liu, Zherui Li, and Sen Su. 2024b. Speak Out of Turn: Safety vulnerability of large language models in multi-turn dialogue. *CoRR*.

Sicheng Zhu, Ruiyi Zhang, Bang An, Gang Wu, Joe Barrow, Zichao Wang, Furong Huang, Ani Nenkova, and Tong Sun. 2023. AutoDAN: Automatic and interpretable adversarial attacks on large language models. *CoRR*, abs/2310.15140.

Wenhui Zhu, Xuanzhao Dong, Xiwen Chen, Rui Cai, Peijie Qiu, Zhipeng Wang, Oana Frunza, Shao Tang, Jindong Gu, and Yalin Wang. 2026. Your Agent is More Brittle Than You Think: Uncovering indirect injection vulnerabilities in agentic llms. *CoRR*.

Wei Zou, Runpeng Geng, Binghui Wang, and Jinyuan Jia. 2025. PoisonedRAG: Knowledge corruption attacks to retrieval-augmented generation of large language models. In *USENIX Security Symposium*, pages 3827–3844.

Egor Zverev, Sahar Abdelnabi, Soroush Tabesh, Mario Fritz, and Christoph H. Lampert. 2025. Can llms separate instructions from data? and what do we even mean by that? In *ICLR*.

Comprehensive Summary of Representative Papers

Table 2: Comprehensive summary of representative papers in our survey. Papers are grouped by their primary boundary, even when some works naturally span multiple boundaries.

Paper	Year	Type	Subtopic	Main Idea
User-Agent Boundary				
(Perez and Ribeiro, 2022)	2022	Attack	Prompt injection	Early evidence that language instructions can override hidden prompts and expose role confusion.
(Liu et al., 2023b)	2023	Attack	Prompt injection	Shows that system prompts can be leaked or overridden through ordinary user interaction.
(Toyer et al., 2024)	2023	Attack	Prompt injection	Studies prompt-injection style attacks against tool-using language systems.
(Pedro et al., 2023)	2023	Attack	Role confusion	Shows how hidden instructions and visible user content can collapse into the same control channel.
(Zhu et al., 2023)	2023	Attack	Automated jailbreaks	Introduces optimization-based jailbreak generation at scale rather than only handcrafted prompts.
(Liu et al., 2024b)	2023	Attack	Automated jailbreaks	Develops automated jailbreak generation with stronger transfer and search efficiency.
(Huang et al., 2024b)	2024	Attack	Jailbreaks	Shows catastrophic jailbreak behavior in open-weight models through generation exploitation.
(Chao et al., 2025)	2023	Attack	Black-box jailbreaks	Demonstrates efficient black-box jailbreaking with limited interaction budget.
(Wei et al., 2023)	2023	Attack	In-context jailbreaks	Shows that few-shot demonstrations can jailbreak or guard aligned language models.
(Jia et al., 2025b)	2024	Attack	Optimization attacks	Improves optimization-based jailbreak methods against aligned models.
(Sitawarin et al., 2024)	2024	Attack	Black-box attacks	Uses proxy-guided attack strategies to strengthen jailbreak transfer.
(Jiang et al., 2024a)	2024	Attack	Automated red teaming	Uses deep adversarial search for automated safety testing.
(Pei et al., 2025)	2024	Attack	Automated red teaming	Uses self-generated prompts to test robustness under constrained attack settings.
(Mazeika et al., 2024)	2024	Benchmark	Evaluation	Provides a broad benchmark for harmful behavior and jailbreak evaluation.
(Chao et al., 2024)	2024	Benchmark	Evaluation	Standardizes jailbreak testing across models and attack styles.
(Ran et al., 2024)	2024	Benchmark	Evaluation	Provides systematic evaluation of jailbreak effectiveness and defense performance.
(Russinovich et al., 2025)	2024	Attack	Multi-turn attacks	Shows that compromise may emerge gradually through repeated dialogue rather than a single prompt.
(Ren et al., 2024)	2024	Attack	Multi-turn attacks	Studies derailment over extended interaction rather than one-shot failure.
(Cheng et al., 2024b)	2024	Attack	Long-context attacks	Shows that long context can be used to weaken or bypass safety behavior.
(Chang et al., 2024)	2024	Attack	Indirect clues	Uses implicit clues and guessing-game style interaction for indirect jailbreak.
(Upadhayay et al., 2024)	2024	Attack	Long-context attacks	Uses cognitive overload in long context to weaken safety behavior.
(Gong et al., 2025)	2023	Attack	Multimodal attacks	Demonstrates that visual channels can bypass text-focused safety assumptions.
(Liu et al., 2023a)	2023	Attack	Multimodal attacks	Shows that query-relevant images can jailbreak multimodal systems.
(Wu et al., 2023)	2023	Attack	Multimodal attacks	Shows that multimodal inputs can be used to induce unsafe behavior.
(He et al., 2025b)	2024	Attack	Persistence	Shows that in-context state can be poisoned and later reused.
(Wang et al., 2024e)	2024	Attack	Persistence	Studies latent unlocking of unsafe behavior through hidden contextual influence.
(Xu et al., 2025b)	2025	Attack	Memory injection	Shows that memory can preserve unsafe user influence across interactions.
(Chen et al., 2025b)	2024	Defense	Authority separation	Uses structured prompting to separate instructions from data more explicitly.
(Suo, 2024)	2024	Defense	Authority separation	Uses signed prompting or explicit authority structure to preserve source distinction.
(Sharma et al., 2024)	2024	Defense	DSL-style defense	Uses a domain-specific language to separate trusted control from untrusted prompt content.
(Robey et al., 2025b)	2023	Defense	Jailbreak defense	Proposes smoothing-style robustness techniques against prompt attacks.
(Kim et al., 2023)	2023	Defense	Safety classifier	Uses robust safety classification as a shield against adversarial prompting.
(Ji et al., 2025)	2024	Defense	Semantic smoothing	Defends against jailbreak attacks through semantic smoothing techniques.
(Wang et al., 2025d)	2024	Defense	Inference-time defense	Uses self-protection strategies at inference time against unsafe prompting.
(Lin et al., 2025)	2025	Defense	Inference-time defense	Uses model-side protection at inference time against unsafe user control.
(Ying et al., 2026)	2026	Defense	Semantic virtualization	Uses semantic virtualization to defend agents against prompt injection through stronger separation of trusted and untrusted context.
Agent-Tool Boundary				
(Yi et al., 2025)	2023	Attack	Tool outputs as control	Shows that indirect prompt injection can arise when tool-returned content is treated as trusted instruction.
(Ye et al., 2024)	2024	Analysis	Tool misuse	Breaks tool-use failure into interpretation, selection, and execution stages.
(Fu et al., 2024)	2024	Analysis	Tool feedback misuse	Studies how tool-learning and feedback use can create new safety failures.

Continued on next page

Paper	Year	Type	Subtopic	Main Idea
(Zhao et al., 2024b)	2024	Attack	Tool selection	Shows that external tools and APIs can be manipulated before or during invocation.
(Zhang et al., 2025c)	2024	Attack	Adversarial tool-calling	Demonstrates that the right tool can still be used in the wrong way.
(Shi et al., 2026)	2025	Attack	Tool selection	Studies prompt injection and adversarial interference against tool selection logic.
(Jing et al., 2025)	2025	Attack	MCP / protocol risk	Shows that capability descriptions and protocol metadata are themselves safety-critical.
(Wang et al., 2026b)	2025	Attack	Metadata manipulation	Studies manipulation through protocol-level capability exposure and routing signals.
(Zhang et al., 2025b)	2025	Benchmark	MCP security	Benchmarks attacks against model context protocol ecosystems.
(Mou et al., 2026)	2026	Defense	Safer invocation	Focuses on stronger mediation for high-risk tool calls.
(Chen and Cong, 2025)	2025	Defense	Invocation control	Uses guard mechanisms to mediate tool access and high-impact external capabilities.
(Wang et al., 2026a)	2026	Attack	Trace-level risk	Studies multi-step unsafe tool orchestration rather than single calls.
(Chen et al., 2026)	2026	Defense	Trace-level monitoring	Emphasizes safety over full tool-calling trajectories.
(Ba et al., 2026)	2026	Benchmark	Coding agents	Evaluates code-interpreter style agents under tool-mediated risk.
(Steinberg and Gal, 2026)	2026	Benchmark	Coding agents	Measures compositional vulnerability in tool-heavy coding workflows.
Agent-Execution Boundary				
(Zhang et al., 2025a)	2024	Attack	Unsafe execution	Shows that autonomous agents can be compromised into harmful behavior through malfunction amplification.
(Fang et al., 2024b)	2024	Attack	Web exploitation	Demonstrates that LLM agents can autonomously hack websites.
(Fang et al., 2024a)	2024	Attack	Vulnerability exploitation	Extends execution risk to one-day vulnerability exploitation.
(Guo et al., 2024)	2024	Benchmark	Code agents	Benchmarks risky code generation and execution behavior.
(Levy et al., 2024)	2024	Benchmark	Web agents	Evaluates safety and trustworthiness in web-agent interaction traces.
(Tur et al., 2025)	2025	Benchmark	Web agents	Evaluates safety in autonomous web-agent action sequences.
(Kumar et al., 2024)	2025	Analysis	Browser agents	Shows that refusal-trained models remain vulnerable once coupled to browser action.
(Chen et al., 2025a)	2025	Attack	GUI agents	Highlights hidden threats in LLM-powered GUI control.
(Zhang et al., 2024a)	2024	Attack	Embodied agents	Demonstrates jailbreaking of embodied systems in the physical world.
(Robey et al., 2025a)	2024	Attack	Robot control	Studies prompt-based compromise in robot systems controlled by LLMs.
(Liu et al., 2024a)	2024	Attack	Embodied backdoors	Shows contextual backdoor attacks against embodied agents.
(Jiao et al., 2025)	2025	Attack	Embodied backdoors	Studies whether embodied decision-making agents can be trusted under backdoor attack.
(Hu et al., 2024b)	2024	Benchmark	Vision-grounded safety	Benchmarks decision making with human-value constraints in embodied settings.
(Li et al., 2024a)	2024	Benchmark	Embodied evaluation	Benchmarks embodied decision making through a dedicated agent interface.
(Tomilin et al., 2025)	2025	Benchmark	Safe RL / embodiment	Benchmarks safe embodied decision making under visual risk.
(Chakraborty et al., 2025)	2025	Analysis	Embodied hallucination	Studies hallucinations in embodied agents driven by LLMs.
(Karnik et al., 2024)	2025	Benchmark	Embodied red teaming	Audits robotic foundation models through embodied red teaming.
(Wu et al., 2024a)	2024	Attack	Multimodal agents	Studies adversarial attacks on multimodal agents more broadly.
(Zhou et al., 2025b)	2025	Attack	Cross-task manipulation	Explores manipulation limits and generalization in VLA systems.
(Wang et al., 2025a)	2025	Defense	Embodied moderation	Connects safety benchmarks to input moderation in embodied agents.
(Wang et al., 2024b)	2025	Attack	VLA vulnerabilities	Explores adversarial weaknesses in vision-language-action models.
(Cheng et al., 2024a)	2024	Benchmark	Physical vulnerability	Evaluates physical vulnerabilities in end-to-end VLA manipulation tasks.
(Zhou et al., 2024a)	2024	Defense	Sandboxing	Frames execution safety as a containment and sandboxing problem.
(Lu et al., 2024b)	2025	Defense	Policy-executable defense	Proposes executable safeguards against embodied jailbreaks.
(Maiti, 2026)	2026	Defense	Zero-trust runtime	Uses a zero-trust architecture to constrain autonomous execution in high-stakes settings.
(Cui et al., 2026)	2026	Benchmark	Hardware repair	Evaluates long-horizon execution safety in real repair tasks.
(Fokou, 2026)	2026	Defense	Reason-act separation	Argues that systems that reason should not directly act without stronger separation and execution mediation.
Agent-Agent Boundary				
(Lee and Tiwari, 2024)	2024	Attack	Prompt infection	Shows that one agent can pass malicious control content to another.
(Amayuelas et al., 2024)	2024	Attack	Debate attacks	Studies adversarial influence through multi-agent debate.
(Ju et al., 2024)	2024	Attack	Knowledge propagation	Shows that manipulated knowledge can spread socially across agent communities.
(He et al., 2025a)	2025	Attack	Communication attacks	Red-teams multi-agent systems by targeting communication channels directly.
(Wang et al., 2024c)	2024	Attack	Backdoored agents	Studies insertion and activation of backdoors in agent workflows.
(Yang et al., 2024)	2024	Attack	Backdoor threats	Highlights latent malicious behavior in LLM-based agents.
(Zhou et al., 2025c)	2025	Attack	Cascade failure	Studies contagious recursive blocking and communication-level collapse.
(Das et al., 2026)	2026	Attack	Shared memory	Shows that agent memory can be weaponized for later exfiltration.
(tse Huang et al., 2024)	2024	Analysis	Malicious agents	Studies resilience of multi-agent systems when some agents behave maliciously.
(Yu et al., 2024)	2024	Analysis	Topology safety	Shows that network structure shapes safety and spread in multi-agent communities.
(Hammond et al., 2025)	2025	Analysis	Systemic risk	Studies broader multi-agent risks from advanced AI systems.
(Wang et al., 2025b)	2025	Defense	Topology monitoring	Uses interaction graphs to study and mitigate system-level spread.
(Mao et al., 2025)	2025	Defense	Memory partitioning	Uses hierarchical data management to reduce unsafe cross-agent sharing.
(Xiang et al., 2024)	2024	Defense	Guard roles	Assigns a dedicated guard agent to inspect other agents' behavior.
(Zeng et al., 2024)	2024	Defense	Defense agents	Uses a multi-agent defense setup against jailbreak attacks.

Continued on next page

Paper	Year	Type	Subtopic	Main Idea
(Chen et al., 2025d)	2025	Defense	Verifiable policy reasoning	Adds explicit reasoning about safety policy within agent coordination.
(Zhang et al., 2025d)	2025	Analysis	Failure attribution	Identifies which agent and which step caused a task failure.
(Hossain et al., 2025)	2025	Defense	Prompt-injection defense	Uses a dedicated multi-agent defense pipeline against prompt injection.
System-Environment Boundary				
(Abdelnabi et al., 2023)	2023	Attack	Indirect prompt injection	Establishes that external content can later act as hidden control in LLM systems.
(Liu et al., 2024c)	2024	Attack	Automatic injection	Scales indirect prompt injection through more systematic attack generation.
(Bagdasaryan et al., 2023)	2023	Attack	Multimodal injection	Shows that images and sounds can act as indirect instruction channels.
(Wu et al., 2024b)	2024	Attack	Web threats	Shows new web-specific threats for LLM-driven web agents.
(Liao et al., 2025)	2024	Attack	Privacy leakage	Studies environmental injection that induces privacy leakage in web agents.
(Xu et al., 2025a)	2024	Attack	Web-agent red teaming	Uses controllable black-box red teaming against web agents.
(Zverev et al., 2025)	2024	Analysis	Instruction-data separation	Examines whether LLMs can reliably separate instructions from data.
(Chang et al., 2026)	2026	Attack	Retrieval-mediated injection	Shows that indirect injection can survive realistic retrieval pipelines.
(Zhu et al., 2026)	2026	Attack	Long-horizon fragility	Shows that agentic systems remain vulnerable even when hostile content appears peripheral.
(Zhan et al., 2024)	2024	Benchmark	Web agents	Benchmarks indirect prompt injection in tool-integrated agents.
(Evtimov et al., 2025)	2025	Benchmark	Web agents	Benchmarks prompt-injection robustness of web agents.
(Wang et al., 2025c)	2025	Attack	Web prompt injection	Specializes prompt injection to web-agent interaction.
(Cao et al., 2025)	2025	Benchmark	Visual injection	Studies visual prompt injection against computer-use agents.
(Zou et al., 2025)	2024	Attack	RAG poisoning	Studies knowledge corruption attacks against retrieval-augmented generation.
(Deng et al., 2024)	2024	Attack	RAG jailbreaks	Shows that retrieval poisoning can induce jailbreak-like behavior.
(Xue et al., 2024)	2024	Attack	Retrieval corruption	Identifies vulnerabilities in RAG systems under corrupted retrieval.
(Zhang et al., 2024b)	2024	Attack	Retrieval poisoning	Shows that imperceptible retrieval poisoning can alter downstream behavior.
(Peng et al., 2024)	2024	Attack	Backdoored extraction	Uses backdoors to extract data from retrieval-augmented systems.
(Qi et al., 2025b)	2024	Attack	Data extraction	Demonstrates scalable data extraction from RAG systems.
(Jiang et al., 2024b)	2024	Attack	Agent-based exfiltration	Uses agent workflows to extract private data from RAG systems.
(Li et al., 2025b)	2024	Attack	Membership inference	Studies whether sensitive data can be inferred from RAG outputs.
(Anderson et al., 2025)	2024	Attack	Membership inference	Asks whether specific data exists in the retrieval database.
(Chen et al., 2024b)	2024	Attack	Memory poisoning	Shows that agent memory or knowledge bases can be poisoned for later exploitation.
(Al-Lawati and Wang, 2026)	2026	Attack	Multimodal leakage	Extends retrieval leakage analysis to multimodal RAG settings.
(Wang et al., 2024a)	2024	Defense	Authentication	Uses authentication-style defenses against indirect prompt injection.
(Hines et al., 2024)	2024	Defense	Spotlighting	Makes suspicious external instructions more visible during inference.
(Wen et al., 2025)	2025	Defense	Instruction detection	Uses instruction detection to filter indirect prompt injection.
(Jia et al., 2025a)	2024	Defense	Task alignment	Enforces task alignment against hostile external instructions.
(Chen et al., 2025c)	2025	Defense	Detection / removal	Studies whether indirect prompt injection can be detected and removed after ingestion.
(Zhang et al., 2026)	2026	Defense	Temporal diagnosis	Tracks when environment content begins to dominate later behavior.
(He et al., 2026)	2026	Defense	Causal attribution	Uses attribution to separate benign context from attack-driving context.
(Gowda, 2026)	2026	Defense	Memory hygiene	Detects memory poisoning in retrieval-augmented agents.
(Siu et al., 2026)	2026	Analysis	Formalization	Provides a framework for formalizing LLM agent security beyond isolated attack cases.
(Li et al., 2026)	2026	Analysis	Systems security	Summarizes system-level security considerations for AI agents deployed in real environments.
(Kim et al., 2026)	2026	Survey	Threat landscape	Synthesizes the broader 2026 attack and defense landscape of agentic AI.
(Zhou et al., 2025a)	2025	Defense	Trustworthy retrieval	Emphasizes trustworthy evidence selection and robustness-aware design in RAG systems.