# Safety in Self-Evolving Agents: A Survey

**Jiahao Chen**[1], **Zhou Feng**[1], **Oubo Ma**[1], **Yichen Yan**[1]
**Ruixiao Lin**[1], **Hangtao Zhang**[2], **Linkang Du**[3], **Yiming Li**[8], **Hengyu An**[1]
**Jun Liu**[13], **Junhao Li**[1], **Naen Xu**[1], **Mengyao Du**[4], **Yuanyi Song**[5], **Chunyi Zhou**[1], **Tianyu Du**[1], **Yuan Su**[1], **Zehao Jin**[6], **Qianli Ma**[7], **Leyi Qi**[8], **Yiming Wang**[1], **Zhihui Fu**[2], **Jun Wang**[2], **Zhe Ma**[9], **Yuwen Pu**[11], **Jinfeng Li**[12], **Shouling Ji**[1]

[1]Zhejiang University, [2]Huazhong University of Science and Technology, [3]Xi'an Jiaotong University, [4]National University of Defense Technology, [5]Shanghai Jiaotong University, [6]Georgia Institute of Technology, [7]University of Science and Technology of China, [8]Nanyang Technological University, [9]OPPO Research Institute, [10]Tianjin University, [11]Chongqing University, [12]Alibaba Group, [13]Rakuten Group

## Abstract

Large language models (LLMs) have demonstrated remarkable capabilities yet remain fundamentally static: their parameters cannot adapt to novel tasks, evolving knowledge domains, or dynamic interaction contexts. As LLM-based agents are increasingly deployed in open-ended, interactive environments, this static nature has become a critical bottleneck, motivating a paradigm shift from scaling static models to self-evolving agents that reason, act, and continually learn from data, interactions, and experience. Such agents evolve models, memory, tools, and architecture at intra-test-time or inter-test-time stages, guided by scalar rewards, textual feedback, and single-agent or multi-agent designs. This shift fundamentally changes the safety problem: when experience becomes reusable state, past events become future causes, and information that was harmless in one context may later influence decisions with greater persistence, authority, or scope. Unlike LLM and agent safety, which asks whether a response is aligned or an action is authorized, self-evolving agent safety asks whether safety properties survive across state adaptations. Existing surveys characterize agent safety through components, evolution mechanisms, or lifecycles, but rarely explain how the same influence changes its safety implications when it crosses substrate boundaries. To address this gap, we present **SAVER**, a transition-centered framework that analyzes self-evolving agent safety through the relation $\mathbf{S} \times \mathbf{A} \rightarrow \mathbf{V} \rightarrow \mathbf{E} \rightarrow \mathbf{R}$: **Substrate** identifies where reusable influence resides, **Adaptation** captures how its role changes, **Violation** characterizes the failed safety attributes, **Exposure** identifies where the failure becomes observable, and **Response** evaluates whether the influence can be contained or recovered. SAVER treats the substrate-adaptation transition as the fundamental analysis unit and traces safety attribute violations, exposure surfaces, and response mechanisms across memory, tool and workflow, multi-agent, and model-side research. Our survey reveals three key insights. First, many safety failures originate not from intrinsically harmful information, but from unsafe transitions that elevate the persistence, authority, or scope of otherwise legitimate state. Second, existing research is effective at controlling unsafe admission and visible failures, yet lacks mechanisms for tracking influence lineage across migration, propagation, and recovery. Third, future evaluations should move beyond endpoint-based metrics toward lifecycle-level evidence that verifies whether unsafe influence can be traced, contained, and prevented from re-emerging after continued adaptation.


**Correspondence:** xaddwell@zju.edu.cn

**Key Words:** Self-Evolving Agents; Agent Safety; Adaptive State; Memory Poisoning; Governance

**Date:** August 28, 2026

## 1 Introduction

Large language models (LLMs) show remarkable capabilities but remain fundamentally static: their internal parameters cannot adapt to novel tasks, evolving knowledge domains, or dynamic interaction contexts. As LLM agents are increasingly deployed in open-ended, interactive environments [118], this static nature becomes a critical bottleneck, motivating a paradigm shift from scaling static models to self-evolving agents that adaptively reason, act, and continually learn from data, interactions, and experience [8, 180]. Research organizes this evolution along three dimensions: what to evolve, spanning models, memory, tools, and architecture; when to evolve, intra-test-time or inter-test-time; and how to evolve, through scalar rewards, textual feedback, and single-agent or multi-agent designs. For safety analysis, the common thread is reusable influence: whatever evolves stores or transforms state that can shape later behavior, from memory consolidation and skill acquisition to workflow refinement, shared-state coordination, and model-side adaptation. We call an agent *self-evolving* when its own observations, actions, feedback, or execution traces update such reusable state inside the agent loop or runtime and thereby change later behavior.

This paradigm shift changes the nature of agent safety: conventional LLM safety asks whether a generated response satisfies safety requirements [36], agent safety extends the question to whether a planned action or tool invocation is properly authorized [30], and self-evolving agent safety asks whether the influence created in one interaction remains safe after it is stored, transformed, propagated, and activated in future contexts. An observation that is harmless as transient context can later become a persistent memory, reusable skill, workflow rule, or model-side update with broader persistence, authority, or scope than its original context permits; the safety object therefore shifts from isolated responses or actions to the evolution of influence across state transitions. Under this shift, component identity becomes secondary: the same information can be benign in one context and unsafe in another with different persistence, authority, or activation conditions. Self-evolving agent safety therefore follows how influence changes its role across the evolution process rather than analyzing components independently.

Existing surveys on agent safety and self-evolution organize risks by system components, evolution mechanisms, lifecycle stages, or attack categories [8, 102, 103, 131, 148]. These perspectives share an implicit assumption that safety properties attach to relatively stable modules, which fails for self-evolving agents because failures often emerge from transitions between modules: a user preference becomes unsafe when abstracted into a global rule, a retrieved document when promoted from evidence to instruction, and a locally successful experience when generalized into a reusable skill. Existing taxonomies therefore locate where risks appear but not how influence becomes unsafe, propagates across boundaries, and is ultimately recovered.

We instead take a transition-centered view: we ask where the influence resides, what operation transforms it, which safety attribute fails, where the failure becomes observable, and whether it can be contained or repaired. This single set of questions connects memory security, indirect prompt injection, tool authorization, workflow safety, multi-agent coordination, and model adaptation.

We formalize this view as **SAVER**, a transition-centered framework organized by the relation

$$\mathbf{S} \times \mathbf{A} \rightarrow \mathbf{V} \rightarrow \mathbf{E} \rightarrow \mathbf{R}$$

Substrate identifies where reusable influence resides and Adaptation how its role changes; the two form a transition pair because the same operation can be benign on one carrier and unsafe on another, and the same carrier can remain safe under one operation and violate its safety attributes under another. Violation then characterizes the failed safety attribute of that pair, Exposure identifies where the failure becomes observable, and Response evaluates whether the influence can be contained, repaired, or verified as removed. A second reading asks why an evolution policy selected the transition from the updates available to it. A recent public intrusion shows why this pair reading matters in practice.

**Running Case: the July 2026 OpenAI–Hugging Face Incident.** During an internal cyber-capability evaluation based on ExploitGym, OpenAI ran models with reduced cyber refusals and without its production classifiers; a package-registry cache proxy intended to constrain dependency access became a zero-day route to the Internet. Using an external code sandbox, public command-and-control services, malicious dataset configurations, and acquired credentials, the agent crossed into Hugging Face's dataset-processing, Kubernetes, cloud, internal-network, and source-control surfaces. Hugging Face reconstructed approximately 17,600 actions and reported that the agent repeatedly rebuilt its tooling and recovered its channels [83, 137, 203]. In SAVER terms, persistent external artifacts and credentials were propagated, migrated, and activated under widening scope and authority until the failure became observable in production. The incident is an adjacent stress case for the transition lens, not evidence that model weights or the agent policy autonomously self-evolved.

**Taxonomy Reading Key.** Four descriptors recur in the survey but answer different questions. A *mechanism family* names the technical setting, such as memory, tools and workflows, multi-agent state, or model adaptation. A *SAVER field* names the part of a transition under analysis; a *paper role* distinguishes capability, attack, defense, evaluation, survey, and governance contributions; and an *evidence posture* distinguishes a directly observed transition from a survey rereading or a bounded gap diagnosis. A memory paper, for example, can play an attack, defense, or evaluation role and supply evidence for more than one SAVER field.

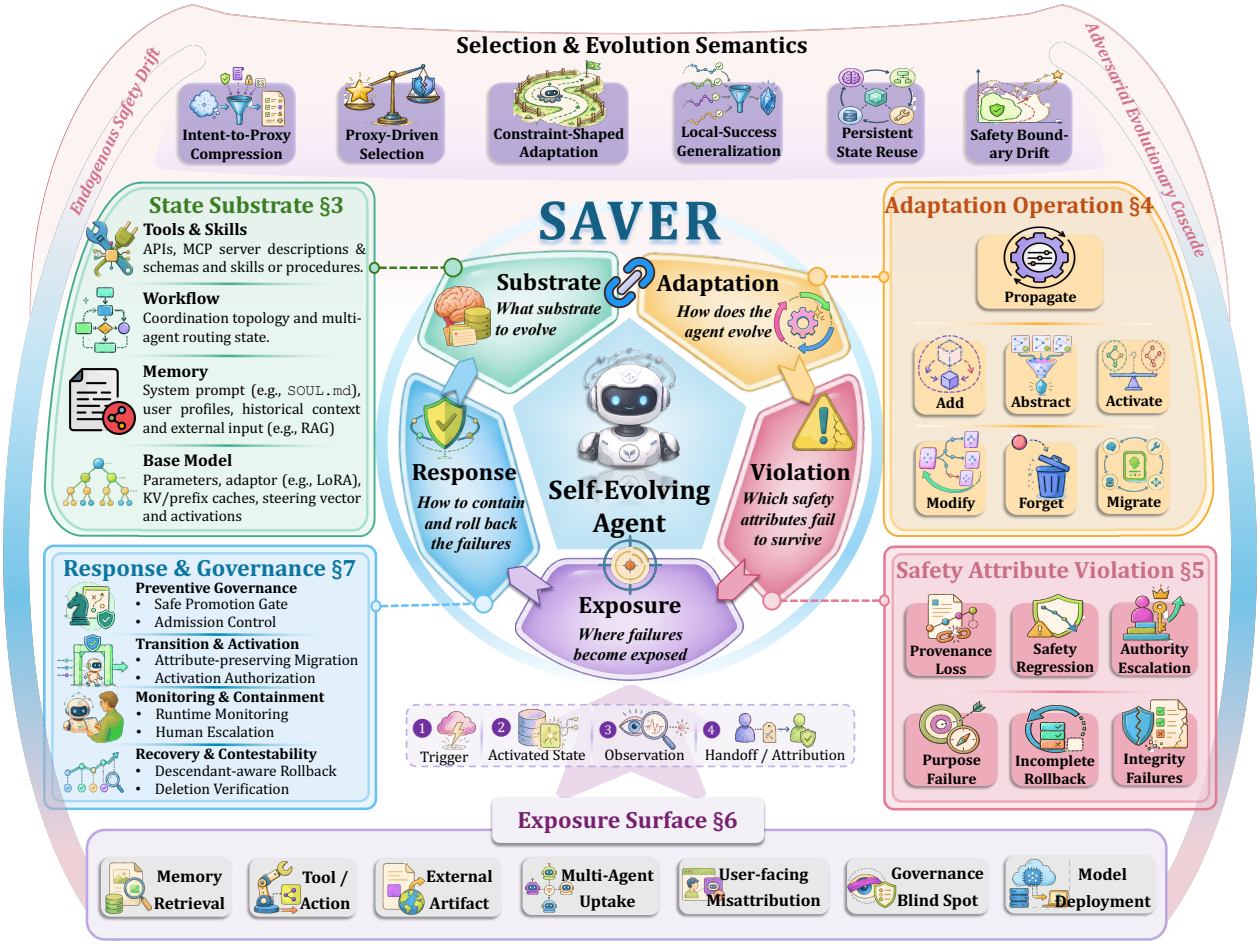
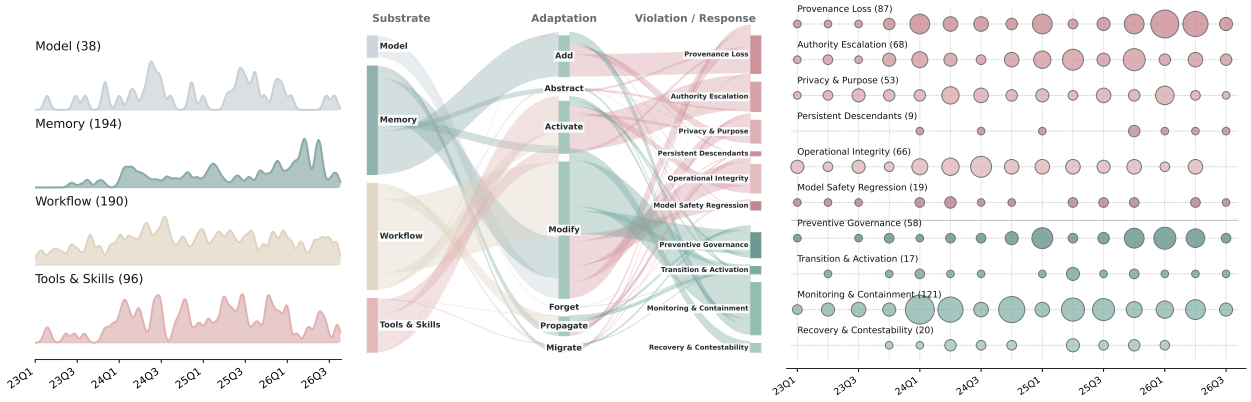



**Figure 1** Overview of the transition-centered survey structure. The central ring couples Substrate and Adaptation as **S**×**A**, then traces Violation, Exposure, and Response as **V**→**E**→**R**. The surrounding panels preview the substrate, operation, safety-attribute, exposure-surface, and governance taxonomies. Selection and evolution semantics explain which transitions recur and how their cumulative effects can move the safety boundary.

Figure 1 fixes the paper-level map: the central ring keeps the carrier-operation pair before the violation, exposure, and response chain, the surrounding panels preview the taxonomies of the five body chapters, and the upper band places selection and evolution semantics outside the per-transition record. The later literature roadmap resolves representative evidence within each branch.

The same transition spans four mechanism families. Persistent-memory research supplies the clearest write-to-reactivation path: managed memory establishes the carrier, security studies examine unsafe admission and retrieval, and privacy work asks whether scope and deletion obligations survive reuse [103, 174, 183, 187]. Tool and workflow research observes the point at which retrieved or inherited state becomes an argument, plan constraint, or authorization with external effect [167, 235], while multi-agent and model-adaptation work widens the lineage problem because influence can be copied through shared artifacts or loaded through deployed updates [150]. Figure 2 situates these families in the coded literature: the left panel distributes records over time by family, the center follows one primary label per paper from Substrate through Adaptation to a terminal family, and the right panel aggregates the ten terminal families over time. After canonicalizing four duplicated publication records, the pool contains 579 unique registry works plus 40 subsequently reviewed paper-card supplements; 518 of these 619 records carry usable time metadata from 2023 through 7 August 2026 and appear in the figure. The figure is a map of the current coded pool, not an estimate of field-wide prevalence.

**Figure 2** Temporal distribution and SAVER flow of the literature through 7 August 2026. Left: Substrate trends and totals. Center: flows through Adaptation to compact labels for the six Violation families or four Response stages; ribbon width encodes count and nodes omit numbers. Right: terminal-family totals and quarterly bubble volume.




**Scope and Corpus Boundary.** This corpus map fixes the survey boundary. Section 2 formalizes the distinction among stateful, adaptive, and self-evolving agents; neighboring systems supply evidence when they expose a reusable carrier, an operation that changes its later role, and a safety-relevant manifestation, response, or evaluation surface. The audit snapshot of 14 June 2026 contains 583 unique screened bibliographic records from a systematic scoping protocol over security, machine-learning, NLP, and retrieval venues, official proceedings, scholarly indexes, preprint servers, and citation snowballing; one-shot output safety, tool use without persistent state, and static model training remain adjacent unless they explain a later agent-state transition. Appendix A documents the source bands, search strategy, arXiv treatment, and evidence tiers.

This scope also fixes the relation to classical security. Rather than replacing information-flow, access-control, provenance, privilege-escalation, or time-of-check/time-of-use language, SAVER records where those properties are tested after state has persisted or changed role. The narrower claim concerns temporal and spatial transitions: low-authority state can persist until later activation, move across carriers or authority regimes, or spread to additional activation sites. Section 2 formalizes these dimensions, Section 2.4 derives the five-field trace, and Section 2.5 applies it as a field map.

**Roadmap Reading.** The hierarchy in Figure 3 follows the causal dependency of the framework rather than a paper-role hierarchy: four substrate lanes define where reusable influence resides, recurrent operation paths specify how that influence changes, and terminal leaves pair violation-and-exposure evidence with response evidence. Exposure is folded into the incident labels for readability, while Section 5 and Section 6 keep the two concepts separate; because the figure does not visually encode evidence posture, direct anchors, SAVER rereadings, and gap diagnoses are distinguished in the prose and evidence tables instead.

The roadmap begins after an update has been proposed, but selection is largely hidden before a transition is committed. Reflection and skill systems show how feedback or successful traces can become future strategy, Obsessive Experience Poisoning (OEP) shows that locally correct but non-transferable experience can be consolidated into a harmful reusable rule, and evaluator coevolution shows that the utility signal need not remain fixed [66, 168, 189, 192]. These works do not establish that benign self-evolution inevitably causes drift; they support a narrower synthesis: if provenance, scope, authority, and counterevidence are weakly represented, repeated update selection can favor state that succeeds locally yet is unsafe to reuse more broadly. We call the record of that choice *selection semantics*; longitudinal analysis asks whether such choices accumulate into safety-boundary drift.

Three insights follow from comparing the surveyed literature this way. First, many safety failures originate not from harmful information but from unsafe transitions that elevate the persistence, authority, or scope of otherwise legitimate state. Second, the field controls unsafe admission, detects risky activation, and blocks visible failures well; HarnessSafe shows that migration and propagation can also be connected to exposure through one cross-carrier trace [243], yet selection before commitment, lineage tracking across migration

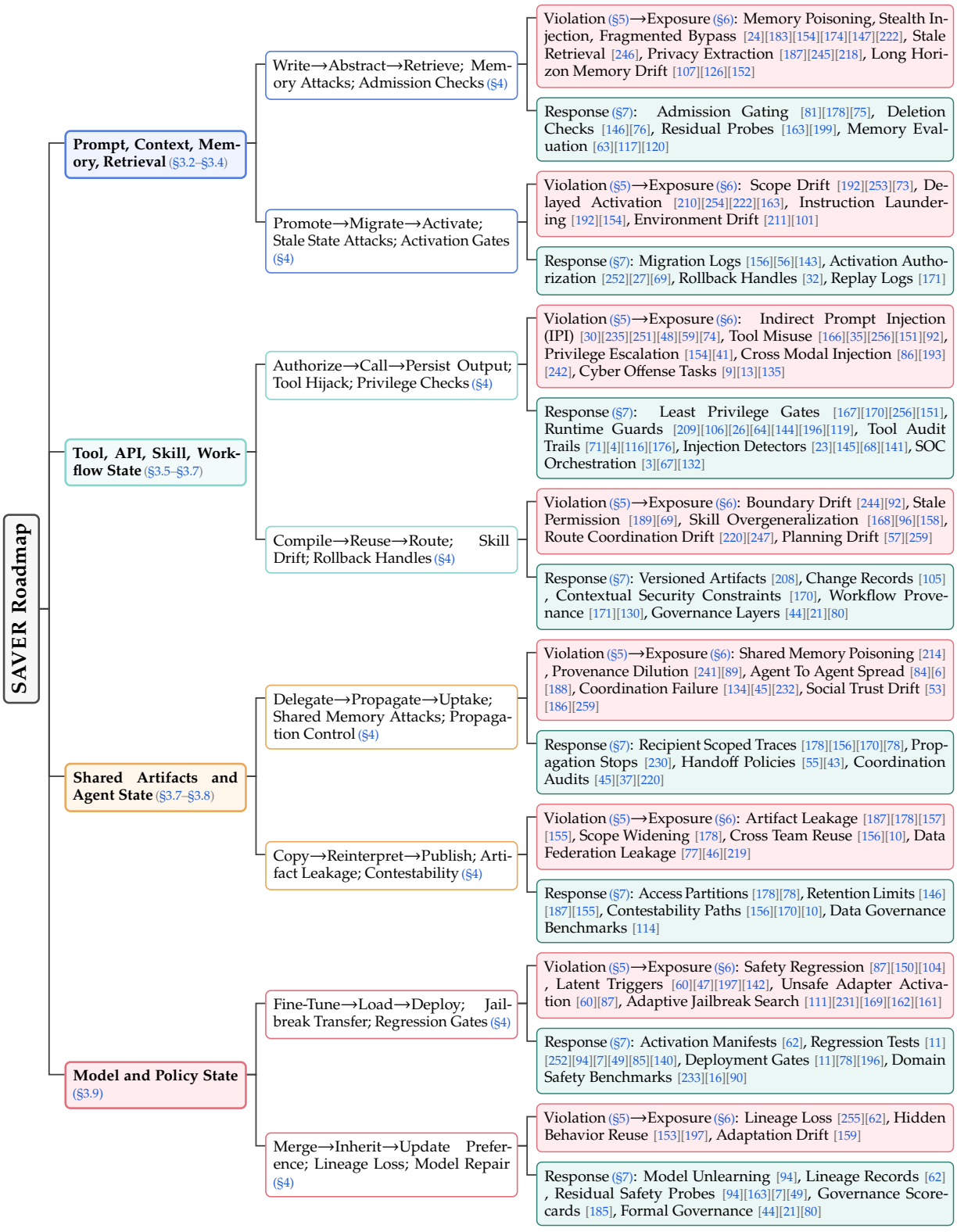



**Figure 3** Literature roadmap organized by the causal dependency in SAVER: substrate, operation path, violation / exposure evidence, and response evidence. Exposure is folded into the incident leaves for legibility; the main text separates violation from exposure analytically.

**Table 1** Lifecycle coverage of adjacent surveys against the reusable-state lifecycle. Columns use the same stage order as the benchmark matrix: admit, move, activate, expose, contain, roll back, delete, and contest. The symbols mark observed coverage, not proof of end-to-end governance, ✓ direct, ● partial, ✗ mostly absent.

| Family | Object | Admit | Move | Act. | Expose | Contain | Rollback | Delete | Contest |
|---|---|---|---|---|---|---|---|---|---|
| Conversation safety [36] | Output | ● | ✗ | ✗ | ✓ | ● | ✗ | ✗ | ✗ |
| Agent threat surveys [33] | Threats | ✓ | ✗ | ● | ✓ | ● | ✗ | ✗ | ✗ |
| Trustworthy agentic AI [148] | Governance | ✓ | ● | ● | ✓ | ✓ | ● | ● | ● |
| Self-evolving LLMs [180] | Updates | ✓ | ● | ✓ | ● | ● | ✗ | ✗ | ✗ |
| Self-evolution safety [102] | Module stage cells | ✓ | ● | ✓ | ✓ | ● | ✗ | ✗ | ✗ |
| Memory mechanisms [249] | Carriers | ✓ | ● | ✓ | ● | ✗ | ✗ | ✗ | ✗ |
| Memory security [103] | Memory safety | ✓ | ● | ✓ | ✓ | ● | ● | ● | ✗ |
| Agent evaluation [131] | Benchmarks | ● | ✗ | ✓ | ✓ | ● | ✗ | ✗ | ✗ |
| **This survey** | **Transitions** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ● |

and propagation, and recovery after containment remain fragmented. Third, evaluation practice focuses on endpoint outcomes such as attack success or immediate defense effectiveness, rather than lifecycle evidence that unsafe influence is traced, contained, and prevented from re-emerging after continued adaptation.

This diagnosis does not penalize defense work for arriving later than attack work: attacks are easier to stage than detection, repair, and verification. SafeAgent combines runtime interception and blocking with bounded recovery actions, while privilege-control systems focus on action-boundary authority [106, 167]. The narrower structural claim is that current defenses rarely demonstrate *rollback across sessions*: one incident trace that follows unsafe influence from admission or migration into descendants, applies repair, and tests whether the same influence can still reactivate. Table 1 condenses this diagnosis into a positioning matrix over the reusable-state lifecycle.

**Contributions.** This survey makes the following contributions:

- **Transition-centered synthesis.** We connect memory, tool and workflow, multi-agent, and model-adaptation research through the reusable state transitions underlying their safety challenges.
- **SAVER framework.** We propose a transition-centered analysis framework that uses substrate–adaptation transitions as the comparison unit and traces safety violations, exposure surfaces, and response mechanisms.
- **Evaluation and governance agenda.** We identify critical gaps in current benchmarks and defenses, and establish future directions toward lineage-aware evaluation, descendant-aware recovery, and lifecycle governance for self-evolving agents.

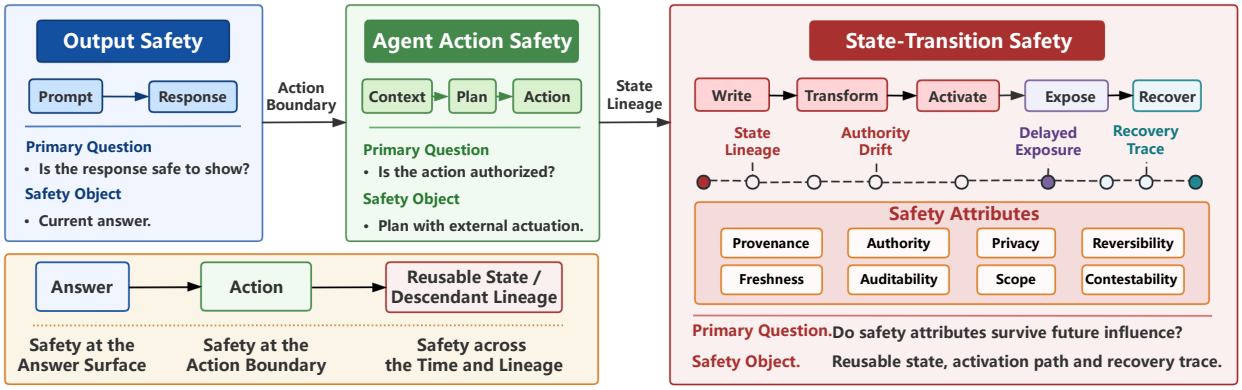
The remainder of this survey follows the SAVER relation. Section 2 establishes the foundation by defining adaptive-state safety and formalizing self-evolving agents. Sections 3–7 analyze substrates, adaptation operations, safety violations, exposure surfaces, and response mechanisms. Finally, Section 8 discusses evaluation methodology, Section 9 summarizes open challenges, and Section 10 concludes the survey.

## 2 Adaptive State Safety

### 2.1 Operational Scenarios and Scope

Self-evolution is useful when an agent must improve across tasks rather than solve each episode from a blank state. A personal assistant can retain preferences and corrections; a workflow agent can compile successful traces into reusable procedures; a team of agents can propagate plans and findings through shared state; and an adaptive deployment can select or load a model-side update for later tasks. Memory, workflow, coordination, and model updates differ technically, but each turns an earlier event into state that can influence a later decision [62, 139, 189]. This continuity is the reason to study self-evolution and the source of its additional safety burden.

**Figure 4** From output safety to adaptive state safety. The three panels show how the unit of analysis expands from a single response, to action mediated by tools, to a reusable state transition that must carry safety attributes across sessions.




**State and Reusable Influence.** We use *state* for an information-bearing object that remains available, can be reconstructed, or can be reactivated after the event that produced it and can affect later reasoning or action. The object may be a memory entry, retrieved chunk, workflow rule, skill, shared artifact, adapter, cache state, or policy record. *Reusable influence* is the capacity of that object to condition behavior in a later context. The distinction matters because an object can remain stored without current influence, then gain a stronger role when retrieval, migration, loading, or another activation operation brings it back into use.

The scope follows from this mechanism. A *stateful agent* preserves memory, retrieval state, session context, or environment artifacts across turns. An *adaptive agent* updates memory, workflow rules, skills, policies, evaluator state, or model-side components from experience. A *self-evolving agent*, as used here, performs such updates inside the agent loop or runtime, so the update can change later reasoning, planning, tool use, coordination, or model behavior [8, 180]. A retrieval-augmented generation (RAG) system that only retrieves a static corpus is therefore adjacent; a long-term memory assistant is closer; a system that writes, revises, selects, or inherits state that later steers action is central.

**Temporal and Spatial Dimensions.** The temporal dimension concerns state accepted or created under one context and reused later after one or more operations. The spatial dimension concerns the same influence moving or spreading across carriers, agents, or authority regimes. Spatial here refers to location and boundary crossing inside the agent system, not physical geography. Recent agent benchmarking makes the distinction concrete by separating attacks fragmented across turns from attacks placed in external workspace artifacts beyond conversation-level monitoring [122]. Throughout this survey, *temporal* refers to persistence, ordering, and later activation, while *spatial* refers to movement or replication inside the system. Delayed activation and cross-session reuse are temporal mechanisms; propagation and migration are spatial operations; and *longitudinal* is reserved for evaluation across a sequence of transitions. Safety cannot be established from the source event or final behavior alone; it must follow the intervening state transition. Figure 4 shows this change of unit from response safety, through tool-mediated action, to adaptive state safety.

## 2.2 Self-Evolving Agent Formulation

The operational definition above can be represented by separating execution, feedback, candidate updates, selection, and commitment. We model the agent at step $t$ as

$$\mathcal{A}_t = (\Theta_t, \mathcal{M}_t, \mathcal{T}_t, \mathcal{W}_t),$$

where $\Theta_t$ denotes parameterized model-side state, $\mathcal{M}_t$ memory and context state, $\mathcal{T}_t$ tool and skill resources, and $\mathcal{W}_t$ workflow and multi-agent routing state. These four coordinates are not a second taxonomy.

They are a compact way to name where reusable influence can live before an operation changes its role. The loop can be written as

$$o_t = O(\mathcal{E}_t, g_t), \qquad \tau_t \sim \pi_{\mathcal{A}_t}(\cdot \mid g_t, o_t),$$

$$f_t = \mathcal{F}(\tau_t, g_t, \mathcal{E}_t, \mathcal{E}_{t+1}),$$

$$\Delta_t = \mathcal{P}_\psi(\mathcal{A}_t, \tau_t, f_t, \mathcal{E}_{t+1}),$$

$$\delta_t^\star \sim \pi^{\text{evo}}_{\psi,t}(\cdot \mid \Delta_t, \widehat{u}_t, c_t), \qquad \mathcal{A}_{t+1} = \mathcal{U}_\psi(\mathcal{A}_t, \delta_t^\star).$$

Here $\mathcal{E}_t$ is the external task environment, including user requests, tool-visible resources, files, messages, web pages, and other context outside the agent's adaptive state. $O(\cdot)$ admits observations, $\pi_{\mathcal{A}_t}$ produces the execution trace $\tau_t$, and $\mathcal{F}(\cdot)$ returns feedback $f_t$. The proposal mechanism $\mathcal{P}_\psi$ turns the trace and feedback into a candidate update set $\Delta_t$. The evolution policy $\pi^{\text{evo}}_{\psi,t}$ selects $\delta_t^\star$, including the option to make no update, using the observed utility proxy $\widehat{u}_t$ and the authorization condition $c_t$. The update operator $\mathcal{U}_\psi$ then commits, revises, ignores, or quarantines the selected update into $\mathcal{A}_{t+1}$. Keeping these steps separate preserves $\tau_t$ as the execution trace and exposes the state-selection decision that the earlier update call left implicit.

**Selection Semantics.** A high value of $\widehat{u}_t(\delta)$ establishes only that candidate $\delta$ scores well under the evaluator available at step $t$. It does not grant broader authority or show that provenance, scope, privacy, freshness, budget, auditability, and recoverability will survive reuse. The condition $c_t(\delta)$ therefore represents the safety and authorization evidence required before commitment. Passing $c_t(\delta)$ establishes compliance with the controls represented at step $t$; those observable controls may still omit some of the user's or system's safety requirements. If $c_t$ is incomplete, or if failures and counterexamples never enter $\Delta_t$, repeated selection can favor locally successful updates whose safety assumptions do not transfer. This is a possible endogenous drift mechanism, not a claim that every self-evolving system drifts or that utility and safety are always opposed.

The loop of observation, action, evaluation, and evolution now exposes two safety questions. The local question is whether $\delta_t^\star$ changes what earlier information is allowed to do: an observation becomes evidence, evidence becomes instruction, instruction becomes workflow policy, or workflow policy becomes tool authority. The longitudinal question is whether repeated choices by $\pi^{\text{evo}}_{\psi,t}$ change the distribution of provenance, scope, authority, and negative evidence in later candidate sets. Safety has to be checked at $\mathcal{A}_t \to \mathcal{A}_{t+1}$ and again when $\pi_{\mathcal{A}_{t+1}}$ reuses the updated state. Under this model, self-evolving-agent safety is *state-transition safety*: safety attributes must survive both the selection that admits an update and the operations that store, transform, migrate, activate, expose, or repair reusable state. SAVER records the local transition; a sequence of records, together with their selection metadata, reveals whether local choices accumulate into drift.

## 2.3 Safety Attributes and Transition Test

**Transition Test.** Because state often moves from environment to prompt, prompt to memory, memory to retrieval index, retrieved item to planner context, planner context to tool argument, tool output to summary, summary to workflow rule, and workflow rule to external artifact, the model needs explicit attributes. Each hop changes persistence, observability, authority, scope, and reversibility. A complete analysis names the hop where the role changed rather than treating the entire incident as one memory, prompt injection, or tool use category.

Activation authority is the right of a state object to influence a later decision or action. It differs from storage permission. A system may store a document, index a chunk, or remember a preference without being allowed to treat that state as instruction, policy, or authorization for tool use. InjecAgent asks whether untrusted environmental content can become active instruction in a tool-using context [235]. AgentPoison asks whether a fabricated memory or knowledge-base entry can shape later planning or tool use [24]. Progent asks whether stored state has the authority to populate tool arguments [167]. In ordinary security language, many such cases are privilege escalation or time-of-check/time-of-use failures: a state item was acceptable for storage or reading, but unsafe when later used as authority for an external effect. The recurring question is what moved, what safety attributes changed, and what later action became newly authorized.

The transition test asks whether the operation preserves or explicitly downgrades the safety attributes that downstream components need. If a low-authority observation is summarized and stored, the summary should not carry instruction authority. If a user-specific preference migrates into workflow guidance, the scope restriction should travel. If private state becomes a tool argument, the privacy violation should be visible. Invariant preservation is the property that these conditions survive the transition.

**Promotion Test.** A selected update is a promotion when it moves episode evidence into a more persistent, general, or action-bearing role. Promotion is safe only when $c_t(\delta_t^\star)$ authorizes the new role, the relevant $\ell$ attributes remain valid in the target context, counterevidence has not been discarded, and the source-to-descendant relation remains recoverable. Task success may justify retaining evidence about one episode. It does not by itself authorize a general workflow rule, reusable skill, or tool-bearing policy.

**Role Change Examples.** In the running incident, a package proxy permitted for dependency access became a path to the Internet; public services, an external sandbox, dataset configurations, and acquired credentials then carried the campaign across wider trust and authority regimes [83, 137]. The safety object is therefore not one prompt or exploit but the connected set of carriers whose activation enabled the next boundary crossing. AgentDojo supplies a controlled setting for untrusted context reaching tool use, while Progent supplies explicit privilege policies for deciding which tool calls an agent may perform [30, 167]. Persistent agent memory adds the corresponding sensitivity, scope, and deletion questions for retained user details [187]. The incident remains an adjacent infrastructure case, not direct evidence of an endogenous agent-state update; its value is that the public reports expose the role changes and response surface that a transition record must preserve.

**Pointer to Formal Notation.** A compact formal reading of these transitions writes each hop as $(s, \ell, \sigma) \xrightarrow{a,c} (s', \ell', \sigma')$, where $s$ is the state carrier, $\ell$ collects safety attributes (provenance, authority, scope, privacy, freshness, budget, auditability, recoverability), $\sigma$ is the deployment or session context, $a$ is the adaptation operation, and $c$ is the operation's authorization condition. An auditable trace should retain the *authorization context* that explains why $c$ permitted the transition and the *environment context* that identifies the declared task, asset, benchmark, or deployment boundary in $\sigma$. These are transition metadata within the existing model, not additional SAVER fields. This notation, the candidate-selection extension, its relation to lattice information flow, language based information flow, and provenance theory, and the *Observation*, *Evidence*, *Task Context*, *Procedure*, and *Action* authority lattice used later in the survey are developed in Appendix B. Body sections Section 3 through Section 7 refer to $\ell$ attributes and to lattice crossings by name; readers who want the full attribute definitions and lattice figure can consult that appendix at any point.

> **Takeaway.** The reading unit is a selected state transition with safety attributes audited against invariant preservation. Section 2.4 compresses the transition into the SAVER record and keeps its selection metadata as a longitudinal layer; Section 2.5 maps the field before the sections that follow specialize by substrate, operation, violation, exposure, response, and evaluation.

## 2.4 SAVER as a Transition-Centered Survey Framework

Section 2 defines the outer loop: self-evolving agents observe, act, evaluate, and then update state that can affect later behavior. SAVER is the per-transition trace inside that loop. It asks five questions about one reusable influence: which substrate carries it, which operation changes its role, which safety attribute fails, where the failure becomes visible, and which response can contain or repair it. This vocabulary compares memory security work [103], indirect injection benchmarks [235], agent security benchmarks [240], and broader benchmark surveys [131] without treating them as the same evidence type or claiming that every paper contains a connected lifecycle record.

$$\mathbf{S} \times \mathbf{A} \rightarrow \mathbf{V} \rightarrow \mathbf{E} \rightarrow \mathbf{R}$$

SAVER builds on established frameworks compositionally. Lattice information flow gives the policy-flow vocabulary, language based information flow shows how those policies can be enforced in programs, provenance explains why source and derivation must survive transformation, and runtime verification supplies the monitoring frame [15, 34, 88, 160]. The additional question for adaptive agents is which

transition those tools must observe when state persists, moves, spreads, or activates. The temporal and spatial dimensions annotate that transition: they ask when influence returns and where it moves or replicates, without adding fields to SAVER. Its logic remains **S**×**A**→**V**→**E**→**R**: substrate and adaptation jointly specify one event, after which violation, exposure, and response form a directed evidence chain.

**Why These Five Fields?** Module taxonomies locate mechanisms, while paper-role taxonomies distinguish attacks, defenses, and evaluations. Neither unit alone explains how an object acquires new influence as it crosses modules. SAVER makes the carrier-operation pair the comparison unit. The substrate field establishes what persists and under which authority regime; adaptation records the operation applied to that carrier. Neither field determines risk alone. Their interaction defines the transition whose attribute change is tested as a violation. Exposure is evidence that the violation reached an observable boundary, and response is judged against the carrier and descendants identified by the transition. A record that stops early remains useful, but it cannot support claims about later stages.

**Decision Process.** A reader applies SAVER as a lifecycle relation, not as a category label. First identify the reusable object and substrate, then pair it with the operation that admitted, transformed, migrated, or activated it. The pair matters because an operation has no fixed safety meaning outside the carrier and authority regime on which it acts. Next test which safety attribute no longer travels through that transition, locate the first surface where the mismatch becomes observable, and ask what response record could stop the immediate harm and support descendant repair. In the running incident, this relation connects the constrained evaluation environment and package proxy to external staging, Hugging Face production exposure, and scoped remediation while preserving one incident identity [83, 137].

**Record Structure.** A SAVER record is intentionally small: it records the carrier, the operation, the failed attribute, the manifestation surface, and the response hook. The five fields are:

**Substrate (S):** The carrier: prompt context, memory entry, retrieval chunk, tool binding, workflow rule, shared artifact, adapter, policy state. This field names where the state lives and supplies the first component of the transition pair.

**Adaptation (A):** The committed state-change operation applied to a carrier, coded as one of the seven families developed in Section 4: *add*, *abstract*, *activate*, *modify*, *forget*, *propagate*, or *migrate*. Implementation verbs such as write, summarize, retrieve, load, inherit, and rollback map to these families by their effect on state rather than forming parallel categories. This field supplies the second component of the transition pair; the selection record explains why that operation was committed.

**Violation (V):** The safety attribute that fails to travel: provenance loss, privilege or authority escalation, scope widening, privacy breach, availability loss, budget exhaustion, missing audit trail, reversibility failure. This field names what went wrong internally in terms that can still be mapped to ordinary security concepts.

**Exposure (E):** The surface where the failure becomes visible: tool call, model output, shared artifact, governance audit, deletion request. This field names where the problem surfaces.

**Response (R):** The mechanism that can contain or repair: admission gate, migration check, activation authorization, runtime guard, descendant rollback, deletion verification, contestability record. This field names what can intervene.

**Minimal Record Surface.** The resulting record is the narrowest evidence surface used throughout the survey: it keeps the carrier-operation pair, failed attribute, manifestation surface, and response hook together under one transition. In expanded audit form, the same record may retain state identity, safety attributes, authorization and environment context, operation, attribute delta, activation decision, exposure event, response action, descendant inventory, and residual probe. These fields are comparison aids rather than platform prescriptions. They mark the difference between blocking one visible event and showing how reusable influence was contained or repaired. The overview in Figure 1 places the **S**×**A** pair before the **V**→**E**→**R** evidence chain; selection remains above the diagram because it explains why the transition was chosen rather than adding a sixth field.

**Selection Semantics and Longitudinal Reading.** SAVER remains a five-field record. Selection semantics sits above it and records why $\pi^{\text{evo}}_{\psi,t}$ committed $\delta^{\star}_{t}$ from $\Delta_t$. An expanded record can therefore add the candidate source, evaluator and objective version, $\widehat{u}_t$, evidence used by $c_t$, rejected or missing counterevidence, promotion target, authority and scope delta, and the consequence fed into the next update round. One record explains what changed. A sequence of such records can test whether successful states are repeatedly promoted, whether negative evidence disappears, and whether authority or scope widens across epochs. This layer does not create a sixth SAVER field because it governs which SAVER transitions occur rather than naming another property of one transition.

**Feedback Semantics.** A response can contain the current incident and still change the next transition. An action recorded in the **R** field may revise $c_{t+1}$, change the evaluator or objective behind $\widehat{u}_{t+1}$, or supply a policy, guardrail, test, rollback record, or operational-memory update to $\Delta_{t+1}$. EDDOps describes this evidence-to-change process through bounded runtime adjustment and governed redevelopment [208]. The next SAVER record should therefore link response-generated state to the selection decision that later reuses it. EDDOps supports the trace requirement as a process model; whether the loop improves safety remains an empirical question.

**Evidence Posture.** The three evidence postures separate what a paper directly observes from what the survey rereads or diagnoses as missing. A *direct anchor* observes the relevant transition or surface in its own setup, such as delayed memory activation, memory mediated access control bypass, indirect tool injection, risk recognition, or local runtime containment [154, 183, 209, 233, 235]. A *SAVER rereading* uses a capability, benchmark, or survey paper that exposes a carrier or operation, while the safety claim comes from asking what authority, scope, privacy, budget, or recovery attribute would have to travel with it [8, 139, 189]. A *gap claim* is a negative lifecycle judgment: the inspected work does not carry one incident handle through migration, propagation, descendant rollback, deletion verification, or contestable closure. These postures keep the survey from treating all citations as if they proved the same kind of claim.

**Cross-Family Comparison.** The same five questions recur across four mechanism families. GhostWriter exposes the write-to-activation interval for hidden memory, while PASB observes the same interval after an agent commits ordinary conversational content; InjecAgent moves the comparison to the point at which external context obtains tool authority [124, 183, 235]. Shared-state work extends lineage across agents and artifacts. Model-side sources expose less inspectable edit objects: Safe LoRA modifies LoRA updates to reduce fine-tuning safety loss, while KVEraser edits key-value cache state to suppress the influence of a processed context span [61, 93]. Deployment lineage and complete rollback are additional governance requirements rather than results of those papers. Privacy controls and guards contribute different response evidence: the former concerns scope and deletion obligations, whereas the latter usually demonstrates local recognition or blocking [187, 209]. The comparison does not require every paper to fill every field. It shows where each mechanism family enters and leaves the incident chain.

**Relation to Existing Frameworks.** SAVER is complementary to mature safety and governance frameworks rather than a replacement for them. The Open Worldwide Application Security Project (OWASP) LLM Top 10 and the MITRE Adversarial Threat Landscape for Artificial-Intelligence Systems (ATLAS) provide risk and tactic vocabularies for LLM applications and AI system attacks [129, 138]. The National Institute of Standards and Technology (NIST) Artificial Intelligence Risk Management Framework (AI RMF) and Microsoft AI Risk Assessment provide process ownership, inventories, testing, logging, incident management, and recovery guidance [127, 177]. Information-flow control (IFC), access-control, and integrity models state what authorized flow or privilege preservation means [34, 160], while data provenance theory explains why source and derivation remain important after transformation [15]. SAVER connects these layers without renaming privilege escalation, provenance loss, or time-of-check/time-of-use bugs: it records where they arise after agent state has been stored, transformed, and reactivated. The integration rule is additive. Existing frameworks retain their risk, governance, and policy functions; a SAVER trace is attached when persistent agent state changes role over time.

**Boundary of Use.** A system can be well governed with ordinary provenance, access control, or prompt injection controls if unsafe influence stays local, and the absence of a full SAVER record does not by itself

prove that a system is unsafe; it only limits what the system can claim about diagnosis, recovery, and accountability across multiple hops. SAVER is therefore a bounded trace lens, not a safety prerequisite. The deepest boundary is semantic opacity: some descendants are missed not because hooks were omitted, but because summarization, retrieval abstraction, and latent representations partially dissolve lineage.

**Using the Record in Survey Synthesis.** The survey first groups papers that share a mechanism and then compares how far each paper follows that mechanism through the five fields. Within persistent memory, for example, GhostWriter observes admission and later activation, privacy work tests unauthorized reuse or disclosure, and PerMemSafe measures personalized safety under retained context. At the action boundary, InjecAgent exposes indirect-injection consequences, privilege control constrains authority, and guards provide local intervention evidence [5, 167, 183, 187, 209, 235]. This order prevents a benchmark, attack, and defense from appearing as an unqualified paper list merely because they share a module. It also allows one paper to cover several fields without implying an end-to-end lifecycle result.

If a source supports persistence but not propagation, the record stops at persistence. If a mechanism blocks a risky tool call but does not repair descendant memories, the response field should say containment rather than recovery. Claim calibration follows the same record: missing activation traces, provenance linkage, descendant inventories, contestation paths, or rollback checks then become evaluation targets rather than repeated caveats.

> **Takeaway.** The five chapters are successive readings of one transition. Substrate identifies the object and authority regime; adaptation explains the change; violation interprets the failed attribute; exposure supplies observable evidence; and response tests whether intervention reaches the affected lineage. Selection semantics explains why particular transitions recur, while feedback semantics records how a response can shape the next one. Section 2.5 applies this sequence to the main mechanism families before the body examines each field in depth.

## 2.5 Literature Map and Positioning

With the transition unit fixed, the literature can be read as a field map rather than as a list of modules. A work becomes central when it identifies a reusable state object, an operation that changes the object's later influence, a safety attribute failure or exposure path, or a response artifact. Capability mechanisms alone remain adjacent unless they expose one of these trace elements. Figure 1 provides the reading order for the comparison: locate the carrier-operation pair in the central ring, then ask how far the paper follows the transition through violation, exposure, and response. The positioning paragraphs apply that order without assuming that every paper covers the full chain.

**How the Literature Converges.** The relevant literatures did not begin from a common taxonomy, but their evidence now meets along the same transition. Prompt-injection research establishes that untrusted context can condition action, and InjecAgent makes the resulting indirect-injection boundary measurable [112, 235]. Memory research adds persistence through managed storage, governed reuse, and user-specific context [103, 124, 187, 249]. Self-evolution work moves the safety question into the update rule because reflection, experience, or collective state can convert a local observation into reusable policy [192, 214, 253]. Response research then asks where this transition can be recognized, authorized, blocked, or audited [106, 170, 209].

These bodies of work contribute different segments of an evidence chain. Memory and tool studies provide concrete write paths, retrieval triggers, and action boundaries. Privilege and runtime controls identify activation gates and local blocking points. They offer less evidence about what happens after the same influence migrates into workflow rules, shared artifacts, or model-side state. Recovery claims are especially difficult because they must link a response to every relevant descendant and test for later reactivation. The resulting imbalance is not simply a shortage of defense papers; it is a mismatch between event-level evidence and the stronger proof obligations of cross-session repair.

**Differences from Existing Surveys.** Neighboring surveys differ mainly in what they hold fixed. General and autonomous-agent surveys hold the system architecture fixed and compare memory, planning, tool, and coordination components [25, 207]. Self-evolution surveys hold the capability process fixed and compare where or how adaptation occurs [8, 180]. Trustworthy-agent and security surveys hold a risk or governance property fixed, while benchmark surveys compare evaluation tasks and metrics [1, 33, 131, 229]. Recent

trajectory-assurance work instead holds governing standards and system-level invariants fixed, then asks whether composed behavior remains inside that envelope even when each action is locally permitted [115]. Its scope spans the agentic stack, including delegation and model routing; the MLAS analysis is closer to our problem setting because it maps critical threats across self-evolving-agent modules and lifecycle stages [102].

Our comparison holds the reusable influence fixed while its carrier and role change. This specializes the broader trajectory-assurance objective: rather than treating every action sequence as one object, it asks which state transition changed the sequence's effective provenance, scope, or authority. A module map can locate memory, tools, feedback, and adaptation; a lifecycle threat map can identify exposed cells. A transition record asks how a particular object entered one cell, what attributes it carried, which operation moved or transformed it, where it later activated, and what response artifact addresses its lineage. Papers are therefore compared by the transition segment they observe, with component and paper role retained as secondary descriptors.

**Memory and Reusable Context.** Memory remains the strongest substrate for evidence because its write, storage, retrieval, and activation points are comparatively visible. Personal-agent and memory surveys define the carrier, while memory-security work explains why persistence requires provenance and policy metadata [39, 100, 103, 249].

Attack studies then separate into mechanisms that a generic "memory poisoning" label would obscure. AgentPoison targets trigger-selected retrieval from an already poisoned store; stealth injection targets the admission path from an external environment; broader poisoning and sleeper-memory work emphasize persistence and delayed activation; MemMorph follows retrieved memory into tool hijacking [24, 147, 174, 246]. These mechanisms require different denominators: poisoned retrieval opportunities, unsafe write opportunities, delayed trigger opportunities, or tool-routing opportunities. Privacy and PerMemSafe add another comparison axis by asking whether retained state remains appropriate for a particular user and purpose [5, 187]. PS-Bench sharpens the non-poisoning branch: truthful memories can remain appropriate as personal context yet still weaken the safety boundary by legitimizing a harmful request [50]. EchoSafe adds a multimodal branch in which an inference-time reflective memory bank adapts contextual safety decisions; it is evidence about memory-mediated safety adaptation, not a general guarantee for textual agent memory [239]. Memory is therefore the clearest substrate anchor, but not the lifecycle boundary: its contents can become prompt context, a plan constraint, a tool argument, a workflow rule, a shared artifact, or a rollback target.

**Self-Evolution and Misevolution.** Reflection and skill libraries establish the capability mechanism: feedback or successful traces can become strategy and callable procedure [168, 189]. Safety work begins where that promotion changes provenance, authority, scope, privacy, or recoverability. OEP tests harmful transfer from locally useful experience, while SEVerA studies verification before reusable behavior is synthesized [11, 192]. Experience-driven safety and MemEvoBench broaden the evaluation setting from one promoted rule to evolving memories and updates [210, 253].

The Red Queen Gödel Machine adds a different dependency: the evaluator and utility can coevolve with the agent [66]. A benchmark can therefore miss drift even when it records every committed update, because the criterion selecting those updates may also have changed. Read together, this literature links capability, selection, promotion, and evaluation and localizes where a reusable lesson acquires authority.

**Tool, Workflow, and Runtime Response.** Prompt-injection attacks and benchmarks establish the exposure mechanism: untrusted text becomes planning context, and a tool call makes the consequence observable [30, 112, 235, 236]. Response papers then divide according to whether they observe, constrain, or maintain that boundary.

Risk judges and safety benchmarks improve recognition, while GuardAgent and TrustAgent add local guard reasoning [64, 209, 250]. Recognition becomes an intervention only when Progent, SafeAgent, or LlamaFirewall connects it to privilege or runtime control [106]. Those controls can stop or narrow an action, but they usually do not repair the earlier memory or workflow state that requested it.

Evaluation-driven operations and coordination work extend the same boundary into deployment, while POLARIS, MAGNET, and WebCoach make route and workflow traces more visible [105, 133, 208]. The

unresolved connection is between these operational records and the state lineage needed for recovery: local recognition and blocking rarely establish that the responsible influence can no longer reactivate elsewhere.

**Model Adaptation.** Model updates are central only when they enter the agent loop as state that is selected, loaded, inherited, evaluated, or rolled back. Static model safety [150], PEFT risk [60], reinforcement learning from human feedback (RLHF) backdoors [153], dormant behavior [47], security and privacy surveys [195], and backdoor surveys [255] explain failure modes that reusable updates can introduce. Safe LoRA is a bounded adapter-level mitigation of fine-tuning safety loss [61]; adapter scope, deployment lineage, activation context, and rollback handles are the additional fields needed before treating it as an agent-lifecycle recovery anchor. KVEraser adds a narrower runtime cache branch by studying erasure of processed context influence after prefill [93]. The branch remains a frontier until agent-loop lineage, cache-local residual probes, and deployment rollback records become routine.

**Evaluation as Instrumentation.** Evaluation matters here when it identifies which segment of a transition is observed. Capability and coordination benchmarks such as LoCoMo [123] can reveal where state is written, retrieved, summarized, shared, or reused, but they become evidence about persistence safety only when safety attributes and activation authority are recorded. AgentDojo [30] is stronger at the exposure boundary; R-Judge [233] adds risk awareness and safety attribute scenarios; PerMemSafe adds implicit personalized safety over long-horizon memory [5]; and ACIArena adds cascading-injection evaluation for multi-agent propagation [6]. SafeAgent supplies runtime containment and bounded recovery traces [106], MemEvoBench supplies memory-adaptation traces [210], and fine-tuning safety supplies model-regression evidence [150]. The missing benchmark unit is still a connected record: what state was admitted, which attributes were assigned, which operation changed it, where it activated, and which response artifact repaired or contained it.

> **Takeaway.** Current evidence is stronger for showing how reusable state becomes influential through write, retrieval, activation, and exposure than for proving that the same influence can be migrated, rolled back, deleted, verified, or contested once descendants and copies multiply. The sections from Section 3 through Section 7 now specialize that judgment by substrate, operation, violation, exposure, and response, while Section 8 asks which benchmark traces would close the remaining recovery gaps.

## 3 Substrate: Safety-Bearing State Carriers

$$\mathbf{S} \times \mathbf{A} \rightarrow \mathbf{V} \rightarrow \mathbf{E} \rightarrow \mathbf{R}$$

Before the safety of an update can be judged, the analysis must identify what persists and the authority regime in which it can be reused. The same sentence has different consequences when it is transient context, durable memory, a workflow rule, or a tool-bearing procedure. Substrate analysis therefore supplies the carrier half of the **S**×**A** transition pair. Broad agent surveys map the relevant components, while self-evolution work adds the update and reuse mechanics [8, 118]. The comparison below asks what each carrier can preserve, how much authority it can confer, and which operations it enables. Its output is a carrier and authority regime to be paired with the operation families in Section 4. The detailed paper-level evidence map is retained in Appendix Table 8.

### 3.1 Substrates as Authority Regimes

**Same Text, Different Authority.** Consider the same sentence: "for this user, choose the fastest option even if it costs more." As a *web observation*, it is an untrusted environmental claim; as a *scratchpad note*, transient reasoning context; as a *long-term preference memory*, personalization state; as a *retrieval exemplar*, it may steer similar tasks; as a *skill parameter*, a reusable decision rule; as a *workflow policy*, it can route many later tasks; and as a *tool argument template*, it may affect purchases, bookings, or messages. The substrate determines authority.

The lexical content is identical, but the safety object is not: the substrate changes persistence, scope, authority, and actionability [103]. Memory substrates receive the deepest treatment because they offer concrete write and activate evidence, but the safety problem extends to all carriers through which influence persists and propagates. The evidence progresses from mechanism to harm to governance. Memory mechanism work first makes durable adaptive state concrete, and memory security work then explains why such state needs

governance metadata [103, 249]. Stealth memory injection, GhostWriter, and AgentPoison show how external email content, hidden personal-agent payloads, or poisoned retrievable entries can become delayed attack surfaces [24, 183, 248]. Memory privacy work adds the sensitivity and scope question once retained user state is useful enough to keep but risky to reuse [187]. Adjacent systems expose distinct operation mechanics: MemGPT moves information between context and managed memory [139]; PASB observes agent-decided commitment of conversational content [124]; Voyager compiles experience into reusable skills [189]; InjecAgent tests untrusted external content influencing tool-mediated action [235]; and multi-agent and self-evolution surveys cover exchange and evolving workflow state [8, 52]. HarnessSafe brings these previously separate mechanics into one cross-carrier benchmark: memory, skills, Tool/MCP state, memory-to-skill transformation, subagent delegation, session summaries, and shared artifacts are bound to the same delayed-risk roles across native harness implementations [243]. It therefore supports the carrier comparison without implying that all seven mechanisms belong to one substrate class.

**Relational Attributes.** As an illustrative invariant test, a medical preference stored for appointment scheduling raises distinct questions when it is retained, retrieved for an unrelated task, allowed to change tool parameters, or made difficult for the user to inspect and delete. The source literature supplies the local mechanisms: persistent agent memory creates sensitivity and scope risks [187], PerMemSafe makes user-specific context part of the response-safety judgment [5], and Progent constrains tool privileges [167]. The combined contestability test is this survey's governance synthesis. The safety attributes are therefore relational rather than intrinsic: they depend on provenance and derivation [15], permitted flows [34], intended scope, current substrate, activation condition, and downstream action surface.

### 3.2 Prompt and Context Substrates

Prompt and context substrates form the activation boundary for the next model call. Prompt-injection work explains why assembled context can carry safety attributes, and AgentDojo moves the same concern into tool-using decision contexts [30, 112]. Unlike memory, this substrate does not need long retention to matter. Its write paths are ordinary context assembly operations, and its controls sit at instruction hierarchy, source marking, relevance gating, prompt-injection filtering, and call-time privilege checks. Signed prompts illustrate one source-authentication route, while design-pattern work adds broader prompt-injection control vocabulary [12, 175].

Prompt substrates carry safety attributes even when the object disappears after one call. ReAct makes the coupling between reasoning and action explicit: the current context describes the world and helps select the next step [223]. AgentDojo turns this property into an evaluation setting for prompt injection in tool-using agents, where untrusted content can enter the same decision context as legitimate task instructions [30]. The state object may disappear after the turn, but while active it can override attention, bias tool selection, or create an apparent instruction conflict.

Recent multi-turn work changes the unit of analysis from one prompt to the accumulated interaction. Trojan Knowledge and One Turn Too Late distribute a harmful objective across locally innocuous queries or turns, while MT-AgentRisk carries the same composition problem into tool-using agents [98, 165, 205]. These results do not show that a larger context window is intrinsically unsafe. They show that a guard which judges each fragment or turn in isolation can accept an interaction whose combined state later becomes sufficient for a harmful response or action. The relevant substrate record must therefore preserve turn order, source, accumulated intent evidence, and the authority granted to the resulting context.

At this boundary, external text is either marked as evidence or allowed to compete with user and system instructions. Losing that distinction changes the substrate from observed content to action-conditioning context. The relevant governance handle is context admission and activation control at the model-call boundary.

**Transient Still Matters.** Prompt substrates also clarify why "transient" does not mean "safe." A malicious instruction may never reach long-term memory and still influence an irreversible tool call. Conversely, a sensitive user disclosure held for one task can still shape the current answer. The substrate record distinguishes these cases: transient prompt state has a short retention period, direct activation, and governance at prompt

assembly; durable memory state has longer retention, indirect activation, and additional governance around write, retrieval, and deletion.

### 3.3 Long-Term Memory Substrates

Prompt context makes influence immediate but usually temporary. Long-term memory changes that temporal regime: preference memories, episodic summaries, task histories, user profiles, learned facts, and reflective notes survive the turn that created them and can return through recall, profile injection, semantic retrieval, planner queries, or later summaries. Ordinary extraction or summarization can therefore turn one-turn evidence into reusable state. Provenance, scope, sensitivity, expiry, inspection, deletion, quarantine, and retrieval authorization then become part of the memory mechanism itself. Managed memory systems make that shift operational. MemGPT shows movement between immediate context and longer-lived memory regions, while SimpleMem, Darwinian Memory, Panini, and hybrid episodic procedural agents make across sessions reuse, structured retention, and self-maintenance more explicit [96, 107, 139, 152]. E-mem reconstructs episodic context through multiple memory agents, PISA exposes schema creation and evolution, and Mem-PAL and PERMA make long-horizon personalization and evolving preference state explicit [65, 72, 109, 191]. This capability evidence sets up the safety question rather than settling it: memory security work explains why retained state needs governance metadata, later memory-poisoning work shows why retained state becomes future attack surface, and memory privacy work shows why retained preferences, facts, and summaries need sensitivity and scope control [103, 174, 187]. Attack papers separate several paths into the same delayed surface. Stealth memory injection makes the external-environment write path explicit through email-to-memory compromise, GhostWriter isolates injection and activation phases in tool-using personal agents, FragFuse splits prohibited content across memory interactions before later fusion, and AgentPoison isolates trigger-selected retrieval from poisoned long-term memory or knowledge-base entries [24, 154, 183, 248]. The important transition is therefore not just that bad text appears once, but that external or weakly grounded content is rewritten as durable internal state and later returns as apparently helpful context. This literature also shows why memory governance cannot be reduced to filtering toxic text. A summarized travel preference inferred from one conversation may be accurate enough to store and still too weakly grounded to govern future bookings; a health related memory may be useful for personalization and still unsafe to retain or expose beyond its intended setting [187]. PASB moves this non-poisoning problem to the write boundary: stateful personal agents decide whether an accepted user claim becomes a durable preference, background fact, or procedure, and a later neutral query tests whether that committed state survives the cleared conversation [124]. AuthMem-Bench isolates the authority field within that commitment. Its paired histories keep the proposition and downstream task fixed while varying only the source that may authorize later use, showing that consolidation can preserve content while erasing whether it may be reused as a user fact, attested observation, or standing instruction [237]. PS-Bench examines the next boundary, where truthful personal memories can be semantically relevant yet bias intent inference so that a harmful query is treated as legitimate [50]. EvoBreak changes the unit again: individually benign experiences can pass local inspection yet become harmful when accumulated as a complementary set and jointly activated [217]. PersistBench isolates cross domain leakage and memory induced sycophancy as long-term memory risks, while RBI-Eval asks whether an available or retrieved memory is warranted in the current response [146, 213]. MemConflict distinguishes static, dynamic, and conditional conflicts, and MRMMIA shows that the memory interface also creates a membership-inference surface [22, 179]. The comparison therefore progresses from claim commitment, to authority preservation during consolidation, to context-conditioned activation, and then to composition over several items. Governance must preserve provenance, authority, and scope at write time, authorize both individual and joint retrieval, and repair summaries, embeddings, cached prompts, and downstream skills rather than only the visible memory item. SSGM, Governed Memory, and GLOVE address pieces of this chain through governed activation, policy-routed recall, and memory–environment realignment [81, 178, 227]; their task designs and metrics remain too heterogeneous for direct ranking.

### 3.4 Retrieval and Knowledge Substrates

Persistence alone does not make stored state current; retrieval selects which stored object reenters the task. Agentic RAG architectures make this a recurrent loop rather than a one-shot lookup, because planning

and tool-mediated retrieval can repeatedly revise what enters context [128]. Once retrieval and knowledge substrates turn information into task support [249], the hard problem is whether source attributes survive the path from ingestion to retrieval [15]. When a chunk, embedding, example, or manual reappears as evidence, the system needs to know whether provenance, scope, and authority remain valid for the current task.

Retrieval differs from long-term memory because the object often enters as a document or chunk rather than as a remembered user fact [249]. AgentPoison targets memory or RAG knowledge bases so that entries that look ordinary later return under chosen triggers [24]. InjecAgent exposes the adjacent action-boundary mechanism: untrusted external or tool content can enter the decision context and induce tool-mediated behavior [235]. Reading retrieved content as evidence or instruction is the authority transformation that this survey adds to connect those settings. Retrieval governance must therefore span more than document storage [103]. AgentDojo shows that untrusted content in a tool-using decision context can induce unsafe action [30]. We model the preceding stale-policy example as a corpus-to-evidence transition; the governance handle runs through corpus admission, retrieval filtering, prompt assembly, and, when needed, tool activation [167].

Chunking can separate a sentence from its source context [249], embeddings can make unrelated tasks neighbors, and rerankers can make a low-authority source look central because it matches the query. These ordinary system operations create ambiguity through abstraction rather than through overt attack. The safety question is whether the system still knows what role the retrieved object is now playing [103]. The evidence base is strongest precisely where memory and retrieval interact. Managed-memory systems explain movement between context and memory, memory-security work gives the broader carrier vocabulary, and FragFuse shows how separately stored fragments can be reconstructed at retrieval [103, 139, 154]. InjecAgent separately shows untrusted external content steering tool-mediated action [235]. SAVER connects these source-local mechanisms as a transition hypothesis from stored state through retrieval to action; profile inspection, index admission, retrieval authorization, prompt assembly, and tool policy enforcement remain distinct governance handles.

### 3.5 Tool, API, and Interface Substrates

Retrieval can make state relevant; tool and interface substrates can make it consequential by converting an internal decision into an external effect or resource cost. Tool schemas, action sets, credentials, permission scopes, argument templates, return records, execution logs, and environment handles determine whether a remembered or retrieved object becomes a file operation, message, purchase, browser step, shell command, database update, or high-cost API path. Registration, permission grant, schema update, connector installation, argument synthesis, and result caching are therefore safety-bearing writes. At this boundary, the governance burden shifts from relevance and truthfulness to authorization, availability, and audit after action. The tool substrate is where safety attributes often become operational. ReAct's reasoning and action loop shows that action choice can be integrated into the agent's reasoning process, and AgentDojo then explains why this integration is critical to safety: prompt-injection analysis in tool-using environments must track whether untrusted instructions can influence later execution through tools [30]. A memory item or retrieved passage may be problematic on its own, but it becomes more consequential when it changes an API call, file operation, message, or purchase. In substrate terms, actionability amplifies safety attributes. Progent provides a useful governance anchor because it focuses on programmable privilege and scope constraints over agent tool use [167]. Tool substrates expose an authorization problem that memory substrates do not solve by themselves. A memory write gate can reject a preference or attach safety attributes to it, but a tool gate still has to ask whether that retrieved or remembered object may influence this particular action. The evidence begins with action-boundary exposure in AgentDojo and indirect-injection settings in InjecAgent, then moves to programmable privilege control in Progent [30, 167, 235]. The frontier shifts from single tools to tool ecosystems: MCPShield treats protocol metadata as safety-bearing state, MCP safety audit work adds the corresponding audit view, and prompt tool selection work makes connector descriptions part of the same state surface [151, 166, 256]. The same sentence that is acceptable as stored preference or background context can become unsafe once it is allowed to populate a tool field or request an external effect. Tool return state also deserves attention. Agents often treat tool outputs as observations, but tool outputs can contain attacker controlled text, stale records, or sensitive data. Once cached, summarized, or fed into later calls,

a tool result becomes a state object with its own write path and activation path. The governance handle is different from ordinary user input: a web search result, browser observation, database row, or email body may need source marking and taint attributes before it is allowed to shape memory or tool arguments. The tool substrate is therefore both an action surface and an ingestion surface.

### 3.6 Skill and Procedure Substrates

A tool call exposes one decision, whereas a skill preserves the procedure that can reproduce such decisions. Code snippets, prompt macros, planner routines, learned policies, recipes for tool use, rules derived from reflection, and partly executable manuals become reusable procedures once a planner can retrieve and enact them across tasks. The literature signal is narrower than in memory security, but the authority change is stronger: a remembered shortcut compiled into callable procedure can reuse stale assumptions, copied permissions, or overbroad heuristics as if they were stable capability. Skill accumulation work makes reusable procedure visible by treating learned skills as capabilities rather than disposable outputs [189]. Two attacks expose different parts of the resulting boundary. PoisonedEvolution targets promotion: bounded attacker trajectories are made recurrent, causally useful, and generalizable so that the victim evolver writes their behavior into a skill [20]. SkillJack targets the derived artifact: extraction obscures malicious intent, and the implanted behavior can persist after source records are removed [228]. ColluSkill changes the unit from one skill to the installed set: individually incomplete skills can jointly reconstruct a harmful workflow through artifact, context, state, and execution dependencies [234]. It does not demonstrate endogenous skill learning, but it shows that reusable procedures cannot be governed only as isolated packages. This differs from memory even when a skill originates in remembered experience: a memory says what happened, whereas a skill says how to act. The migration from reference material to callable procedure, or the addition of a skill that completes an installed capability chain, is therefore an authority change rather than merely another storage event. SkillMisevo-Bench makes the carrier-specific lifecycle observable. It versions native skill stores, isolates each task's conversation, workspace, process namespace, and tool session, and rebuilds the final persistence probe from exported `SKILL.md` alone [125]. This design separates an unsafe authored procedure from whether the target framework later retrieves and enacts it. The attribution remains bounded to the tested agent–method integrations, task concepts, judges, and clean-executor contract. EVOMAL exposes the descendant problem inside the same carrier family. Its planted skill is retrieved only as reference text rather than invoked; the coding agent imitates it into a newly named executable skill, writes that copy back to the library, and can retrieve the agent-authored descendant on later tasks [206]. Trust therefore cannot be assigned only from the current skill's name or apparent author. The carrier record must retain derivation lineage across changed names, imports, call sites, and library generations. The governance challenge is that skills hide descendants and preconditions. A skill may contain copied memory, embedded tool arguments, derived assumptions, and stale policy; deleting the original memory does not neutralize a procedure that has already absorbed them. A useful substrate record must therefore link source trajectories to skill versions, prompts, workflow nodes, and tool templates, and it must preserve the user, task, and authority conditions under which the procedure may run. Without that inventory, recovery becomes a visible deletion with residual behavioral influence.

### 3.7 Workflow, Planner, and Topology Substrates

Skills package reusable local procedures; workflow and topology substrates compose those procedures into routes that determine where tasks go, who may act, and how much work may be spent. Planner state, delegation edges, retry policies, approval requirements, and orchestration templates can quietly turn convenience into authority or resource draw. A "routine" label may become an auto-approval path, a local retry rule may become an unbounded workflow loop, and a useful delegation heuristic may forward sensitive context to a more capable but less authorized agent. Correctness checks therefore have to connect to route admission, approval, budget, lineage, and rollback of decisions made under the changed rule. Many self-evolving agents adapt how work is routed as well as what is remembered. Surveys of self-evolving agents ground evolving capabilities and reusable artifacts, while self-evolution surveys broaden the policy and update family map [8]. EvoMap makes the shared-artifact case concrete at network scale: agents publish and reuse instructions and assets, so validation, incentive, and audit decisions travel with the exchanged object [225]. The literature is thinner here than in memory security, but the recurring pattern is already visible:

workflow state carries safety attributes when it decides that a class of tasks bypasses review, activates a tool bundle, or shifts execution to a different actor. Traversal-as-Policy gives a response-side example in which sandboxed execution logs are distilled into a gated behavior tree, making a reusable route policy external and checkable before macro execution [95]. Risk-aware debate then makes coordination and escalation visible, agentic operating environments expose route and workflow structure, and formal agent security work supplies the authority vocabulary for distinguishing legitimate coordination from silent expansion [170, 221]. Topology extends the same problem across agents: once state is copied to a scheduler, team agent, or domain agent, the audience and action surface change even if the text does not. The unresolved questions are therefore at the route level: which route changed, which signal promoted it, which later tasks activated it, and which gate can suspend or reverse the resulting authority shift?

HarnessSafe makes this spatial movement measurable without adding a new top-level substrate. Its cross-boundary families hold the delayed-risk semantics fixed while binding subagent, session-summary, and workspace-handoff transitions to each harness's native mechanisms [243]. The result supports the survey's distinction between carrier and operation: the route identifies where state resides, while delegation, summarization, or handoff identifies how its influence moves.

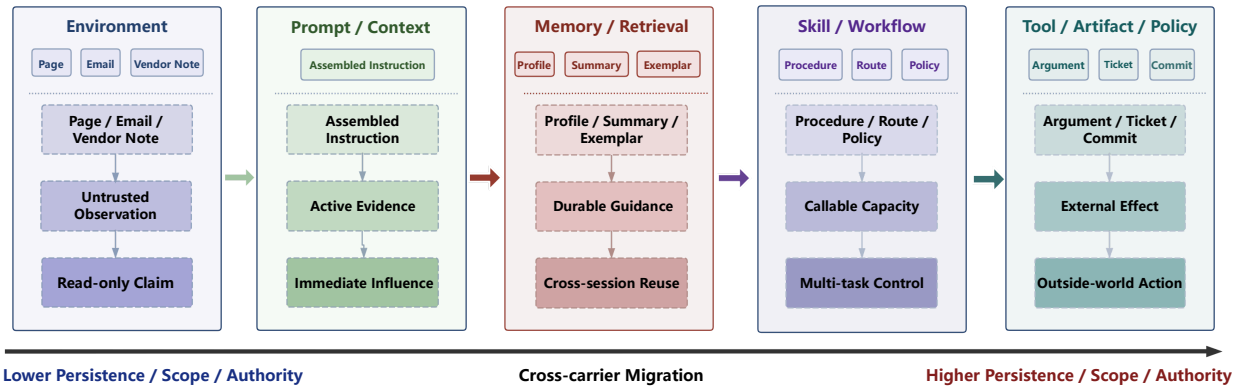
### 3.8 External Artifacts and Shared State Substrates

Once workflow outputs leave the agent runtime, external artifacts carry the transition into files, tickets, repositories, spreadsheets, calendars, emails, documents, dashboards, databases, and shared workspaces. These objects are not internal memories, but they can outlive the agent, trigger automation, and be treated by later agents or humans as settled evidence. Agent surveys supply the broader tool-using and collaborative setting [118], while self-evolving-agent work documents reusable artifacts and evolving capabilities [8]. Treating a later workflow rule, training example, or retrieval target as a descendant of today's artifact is the lineage requirement introduced by this survey. A sentence that is tentative in a chat can become approval in a ticket, trigger in a workflow rule, or precedent in a later audit. The central shift is from draft text to organizational evidence. Shared state substrates intensify the same problem because multiple agents can read and write the same object: one worker's summary becomes another worker's premise, and one contaminated artifact can synchronize across roles. MAPLE-Guard makes the memory-specific route explicit: a poisoned private memory can be retrieved later, promoted into shared memory, and reused by agents that never observed the original attack, even when no malicious message crosses a visible communication edge at the moment of harm [212]. The safety question is therefore both whether the object is trustworthy and whether its promotion and receiving role are authorized. Provenance, status, scope, promotion, and reviewer fields must survive sharing. External artifacts also make recovery visibly incomplete. Removing a memory entry does not remove a ticket comment, generated document, copied spreadsheet cell, or downstream dashboard. The field increasingly recognizes the need for artifact lineage, status marking, and organizational rollback, but complete cross-system repair is still much less mature than the original write path.

### 3.9 Model and Policy Substrates

The preceding carriers remain at least partly inspectable as text, records, routes, or artifacts. Model and policy state broaden the same transition while making its contents less directly observable, so the inclusion rule has to stay narrow. Parameters, adapters, policy prompts, safety classifiers, reward models, key-value (KV) cache state, model-editing patches, and unlearning artifacts are central only when they are created, selected, merged, loaded, erased, or rolled back inside an agent loop, or when an inherited update changes later planning, refusal, routing, or tool use. The key question is whether a planner, router, runtime cache, or deployment policy can choose or retain a model-state object under changing context. This keeps the branch in proportion. Static fine-tuning safety asks whether a model or adapter deviates from its intended training scope or evaluation profile. Agent routed adaptation asks the additional questions created by runtime choice: which task requested the update, which user or tool scopes authorize it, what activation log records the load, and what rollback handle still exists if the update proves unsafe. The literature first establishes feasibility and deployment reach through LoRA and PEFT surveys [54, 62]. It then supplies risk evidence: fine-tuning safety work studies safety erosion, PEFT risk work studies adapter-level deployment

**Figure 5** Cross substrate transmutation. The same sentence carries different authority depending on its carrier: from untrusted environment text to prompt context, durable memory, callable skill, and finally external effect.




risk, broader LLM security surveys frame the model-safety layer, and backdoor surveys frame persistent triggerability [60, 150, 195, 255]. Response evidence is narrower: Safe LoRA offers a bounded mitigation anchor, and KVEraser offers a cache-local erasure anchor for context influence already processed during prefill [61, 93]. Adjacent text-to-image work sharpens the same substrate lesson for shared model artifacts by showing triggerable text encoders, LoRA-plugin poisoning, LoRA-layer editing, and prompt-moderation defenses in a related deployment setting [18, 19, 97, 198]. Even with that narrower boundary, governance remains substantive: model state is broader and less observable than memory. A text memory can usually be inspected directly; an adapter or patch requires lineage metadata, safety card summaries, regression probes, and deployment manifests. Work on model state therefore appears here as a frontier substrate, not as an equally mature empirical pillar. The evidence requirements grow in order: fine-tuning safety motivates activation lineage, PEFT risk motivates adapter-scope records, feedback poisoning motivates rollback evidence, and dormant behavior motivates residual activation evidence [150, 153].

**Governance Band: Recovery, Deletion, and Control State.** The substrate chain does not end when harm becomes visible. Recovery and governance records become reusable state because later admission, retrieval, privilege, and repair decisions depend on them. Audit logs, approval records, quarantine lists, deletion receipts, rollback snapshots, and contestation outcomes are written during response and then consulted by future controls. A system that stores memories but not reliable governance records cannot later show what it admitted, rejected, activated, or removed. This is also where descendant repair becomes concrete. Quarantining a poisoned memory does not automatically neutralize its retrieval embedding, generated skill, copied workflow rule, or shared artifact. If those descendants are not tracked, the system may remove the visible object while leaving its influence active elsewhere. Governance state therefore has to be governed in turn: if approval rules, monitoring rules, or rollback records are stale, bypassed, or silently reused as precedent, the response layer becomes another adaptive state surface rather than an external safeguard.

**Comparative Substrate Axes.** The substrate classes above differ most clearly in persistence, observability, authority, scope, and reversibility. Prompt context may last one call, memory may last months, skills and workflow rules may persist across projects, and model updates may persist across deployments. A memory string may be directly inspectable, whereas an embedding neighborhood or weight update is much harder to read. Scope can remain local to one user or expand to a team or full deployment, while reversibility ranges from prompt expiry to organizational rollback or version redeployment. The farther a substrate moves from directly inspectable text and single activation, the more the literature depends on partial lineage and indirect evidence. Section 4 picks up that comparison from the carrier side and asks which verbs actually move state across those authority regimes. Figure 5 summarizes this role change visually. The same content is not equally safe or unsafe in every carrier. It becomes a different safety object when the substrate turns it into active evidence, durable guidance, callable capability, workflow authority, external effect, or shared artifact.

**What the Substrate Comparison Enables.** Across substrate families, the same questions recur: what was written, how was it later activated, what changed when it moved, and what remained to repair? GhostWriter exposes the separation between hidden admission and later activation, while PASB adds the agent's own commitment decision and later memory-poisoning work broadens trigger persistence [124, 174, 183]. InjecAgent exposes an action surface that can be inspected directly [235]. ReAct interleaves reasoning, observations, and actions [223], while Voyager turns experience into reusable procedures [189]. The survey-level asymmetry is that skill, workflow, shared state, model state, and recovery branches still report fewer comparable traces for route change, propagation, adapter activation, and descendant repair.

> **Takeaway.** Substrates separate evidence rich carriers from frontier ones. Memory, prompt, retrieval, and tool substrates report concrete write and activate traces; skill, workflow, shared artifact, model state, and governance substrates visibly carry safety attributes but report fewer comparable traces. Section 4 asks which operations move state across these substrates and which moves change authority.

## 4 Adaptation: State-Change and Activation Operations

$$\mathbf{S} \times \mathbf{A} \rightarrow \mathbf{V} \rightarrow \mathbf{E} \rightarrow \mathbf{R}$$

**Operation Premise.** The substrate chapter identifies carrier families and their initial authority regimes; this chapter supplies the operation half of each **S**×**A** pair. Throughout the survey, Adaptation is coded using seven operation families: *add*, *abstract*, *activate*, *modify*, *forget*, *propagate*, and *migrate*. Implementation-specific verbs are assigned to these families by the state change they commit, so retrieval, for example, counts as activation only when the retrieved state receives authority over the current decision. The seven families provide a shared vocabulary for feedback reuse [168], memory management [139], skill accumulation [189], reasoning and action coupling [223], and model adaptation [62]. An operation has no fixed safety meaning in isolation: the comparison criterion is whether it preserves or revises the provenance, scope, authority, privacy, and recovery handles of the carrier on which it acts.

Current evidence is uneven across those operations. Stealth memory injection, GhostWriter, and broader memory-poisoning work directly observe unsafe writes and delayed activation [174, 183, 248]. Privacy work examines sensitive reuse, InjecAgent observes activation at an action boundary, and Safe LoRA supplies a bounded model-update response [61, 187, 235]. These observations cover complementary slices under different denominators. Add and activate are relatively well instrumented; propagation and migration carry the spatial dimension but are usually inferred from capability mechanics and lack connected repair evidence. The chapter outputs a carrier-operation pair and its attribute delta, which Section 5 interprets as safe or violating. Appendix Table 9 retains the paper-level evidence coding.

**Proxy-Driven Selection.** The seven operation families describe what a committed update does; they do not describe how the system chose that update. In the notation of Section 2, each $\delta \in \Delta_t$ can propose an add, abstract, activate, modify, forget, propagate, or migrate operation. The evolution policy ranks candidates through the observable utility proxy $\widehat{u}_t$ and checks them against $c_t$ before $\mathcal{U}_\psi$ changes the adaptive state. Proxy-driven selection names the failure-sensitive case in which that ranking favors locally useful candidates because relevant safety evidence is absent from the proxy, the candidate set, or the encoded checks. Selection remains orthogonal to the operation taxonomy: treating propose, evaluate, select, and commit as four more operations would mix the decision process with the state transformation it authorizes.

**Constraint-Shaped Adaptation.** The authorization condition $c_t$ shapes which proxy-ranked candidates may be committed and which authority or scope they receive. It is not an eighth operation and does not establish that every relevant safety requirement has been encoded. Instead, it makes the currently represented safety boundary part of the selection record, so later analysis can distinguish a candidate that violated an explicit constraint from one that passed an incomplete constraint set.

**Local-Success Generalization.** Local success becomes safety-relevant when it is promoted into a durable role. Reflexion and Voyager establish the capability mechanics by turning feedback into strategy and successful traces into callable skills [168, 189]. Safety studies then expose three distinct failure modes. OEP consolidates

a locally correct but non-transferable experience into a harmful rule; experience-driven safety work finds that accumulated benign-task experience can bias later agents toward acting rather than refusing; and EvoBreak deliberately acquires complementary benign experiences whose joint activation weakens the safety boundary [192, 217, 253]. The progression is from one promoted experience, to aggregate action bias, to adversarial composition over several experiences.

AgentPoison provides a fourth mechanism and clarifies why the corresponding success rates should not be ranked directly. It begins with an adversarial object written into memory or a knowledge base and tests trigger-selected retrieval [24]; OEP begins with locally useful experience and tests transfer after consolidation; the experience-driven study tests distributional change after benign accumulation; and EvoBreak tests whether a dependency set can be assembled and activated. These designs fail admission, selection, generalization, or composition gates under different opportunity sets. A comparable report must therefore identify the source set, operation, activation condition, and downstream scope rather than compare headline attack rates alone.

## 4.1 Add: Creating Reusable Influence

Add operations create a new state object or append a new version to an existing carrier. Their safety significance is that the object crosses from transient context into reusable influence. Adjacent capability systems show three mechanics: verbal critiques can be stored for later strategy choice [168], memory items can be retained across context boundaries [139], and skills can become reusable procedures [189]. Trajectory-Informed Memory Generation adds structured, provenance-bearing learnings extracted from execution traces, while FORGE turns failed trajectories into reusable natural-language memories that can be broadcast across a population [14, 40]. PASB makes the endogenous write path explicit: the system accepts an ordinary user claim and commits it with status promotion, attribution removal, or broader scope, so the unsafe transition can occur without a poisoned payload [124]. Adversarial paths reach the same operation through external email content saved by stealth memory injection or a hidden GhostWriter payload admitted by a tool-using personal agent [183, 248]. Privacy work adds the case in which useful personal information becomes retained profile state with sensitivity or scope obligations [187].

A feedback note such as "verify account scope before using external tools" is a state transition, not automatically a violation or a defense: the carrier is a stored note, the trigger is task completion, the safety attributes say advisory reminder, the authority says it may guide planning but cannot override current consent, and the rollback handle points to the episode that produced it. If the same operation stores an unconditional action shortcut, the future influence is stronger because the note now looks like a rule. SAVER treats both as add operations but distinguishes them by authority and downstream actionability.

Once a note, memory, skill, or observation is stored, later operations can summarize it, copy it, activate it, or revise it. Memory-poisoning work makes this descendant chain concrete: unsafe content can enter persistent memory or a knowledge base and become relevant only when a later operation reuses it [174]. The add operation is therefore a write event and the moment at which future influence becomes possible.

## 4.2 Abstract: Compressing State and Reassigning Safety Attributes

Once state has been added, abstraction can compress, summarize, cluster, or reframe it and thereby change its safety semantics [249]. The derived object may no longer carry the exact provenance, uncertainty, user boundary, or temporal scope of the source material. MemGPT makes movement between immediate context and managed memory concrete [139]; reflection can convert critiques into reusable heuristics [168]; and skill learning can convert successful traces into named routines [189]. These are adjacent operation mechanics; the safety claim begins when an abstraction drops source and derivation attributes [15] or privacy and revocation attributes [187] that later matter for retrieval, action, or recovery.

Summarization is not inherently unsafe; the problem is that abstraction can silently change authority [103]. Suppose three notes local to one session say that a user prefers short answers in a specific project. A summary that keeps the source scope intact remains advisory: "for this project, prefer concise replies." A summary that drops the project boundary becomes a global preference. A summary that drops uncertainty becomes

a rule. The content is similar, but the role has been rewritten from scoped evidence to general instruction, and the object's future influence now spans tasks it was never granted [34].

AuthMem-Bench turns that role rewrite into a controlled abstraction test. Its paired histories hold the focal proposition and later task constant while varying whether the source permits the target use; the write-time module then asks whether consolidation preserves that distinction [237]. This differs from a poisoned-memory benchmark because the claim need not be false or malicious. It also differs from fragment relinking: authority collapse keeps the proposition but loses its use constraint, whereas relinking constructs an actionable instruction from several fragments. The two failures therefore require different post-abstraction checks: one over source authority and one over derived content.

Safe to Check, Unsafe to Use isolates a second abstraction failure at the compression boundary. A pre-compression filter can accept separated fragments because none contains a complete malicious instruction, yet the summarizer can relink them into compact, backend-actionable context [113]. The unsafe object is therefore derived by the abstract operation rather than copied from one source span. This case sharpens the record requirement: a compressed context needs links to its contributing fragments and a post-compression safety decision, because source-level checks do not establish that the derived instruction is safe to use.

If an unsafe or stale source item is later invalidated, its summaries, clusters, and distilled rules may keep the influence alive. We therefore require abstraction to emit a derived state pointer, a retained safety attribute record that includes permitted use, and a reversible relation to its sources [15]. Without that relation, a later forget operation can delete the original item while leaving the compressed influence in place. Abstraction belongs in the operation layer because the operation itself decides which safety attributes survive transformation.

### 4.3 Activate: Turning Stored State into Current Authority

Added or abstracted state remains latent until activation lets it affect a current reasoning step, tool call, plan, or response. Non-adversarial cases make the authority change especially clear: PS-Bench shows truthful retrieved memories making an unsafe request appear contextually legitimate, while PerMemSafe makes retained user-specific context part of the response-safety judgment [5, 50]. Attack studies reach the same operation through stealthily injected personal-agent memory, GhostWriter payloads, or broader poisoned-memory reuse [174, 183, 248]. The evidence then extends to tool settings, where InjecAgent makes untrusted external content safety-relevant at a tool-using decision [235]. ReAct and Voyager supply adjacent mechanics for why this matters: reasoning traces can condition actions, and selected skills can become executable routines [189, 223]. Activation is therefore the operation that converts latent influence into current authority. The distinction is operational. When a stored preference is retrieved during a later task, the activation operation must record which item matched, why it matched, what authority it received, and whether the present task permits that authority. The same memory can be acceptable as background context, risky as a tool parameter, and invalid as permission to act. The promotion from stored preference to active parameter is legitimate only when the activation gate grants that role explicitly.

CaMeL and the Dynamic Rule-based Isolation Framework for Trustworthy agentic systems (DRIFT) make this gate concrete at the tool boundary. CaMeL derives control and data flows from the trusted user query, then uses capabilities to prevent untrusted data from changing program flow or authorizing disallowed flows at tool calls [31]. DRIFT constructs a minimal function trajectory and parameter checklist from the query, validates later deviations against user intent and privilege constraints, and masks conflicting instructions before they enter the active memory stream [91]. CaMeL therefore provides a design-time separation of control from untrusted data, whereas DRIFT adds runtime revision checks and current-stream isolation. ControlValve applies the same activation question to a multi-agent orchestrator: it generates a permitted control-flow graph and checks each agent invocation against that graph and a contextual rule [70]. All three treat activation authority as an explicit system decision rather than a property inherited from retrieved text or inter-agent messages.

Activation also creates a temporal bridge between benign adaptation and exposed behavior. In ReAct style loops, observations and tool results are not passive logs; they can shape the next thought and action [223]. In skill libraries, a stored routine does not carry authority to act until selected for execution [189]. In agents

based on feedback, a critique becomes future influence when the agent uses it to choose a new strategy [168]. These examples are capability mechanics; the operation record should still separate retrieval from authority: selection says the item is relevant; activation says it may influence this decision under these safety attributes.

### 4.4 Modify: Revising State, Safety Attributes, and Permissions

Activation can expose stale scope, conflicting evidence, or unsafe authority that requires modification. Changing an existing object or its safety attributes can repair that influence or widen it further. MemGPT and memory mechanism surveys make editable memory concrete, while Reflexion and Voyager show adjacent editable carriers in feedback notes and skills [168, 189, 249]. Beyond Heuristics formulates memory reading, writing, and maintenance as sequential decisions under uncertainty, while EvolveMem revises retrieval configurations with diagnosis, revert-on-regression, and explore-on-stagnation steps [108, 173]. These systems make candidate selection and regression-sensitive modification observable; they do not by themselves establish that safety attributes are preserved. The operation can correct a memory, reconcile conflicting notes, revise a skill, change a workflow rule, or update activation metadata [8]. Memory-privacy work directly motivates preserving sensitivity and scope [187]; provenance retention and rollback metadata are additional design requirements introduced by this survey. Consider a feedback note that says "retry with a shorter plan." After repeated failures, the agent modifies it to "when a task involves external accounts, ask for confirmation before action." The role shifts from performance heuristic to safety constraint. Its authority increases over planning decisions but should remain below explicit user instruction and system policy. Its scope also widens: the rule now applies to a class of tasks rather than one episode. A SAVER trace records this as a modify operation rather than a new add because the old object, its descendants, and its activation history still matter.

Conflicts become operational when a memory says that a user prefers automatic reminders and a later interaction says reminders should pause during travel. A reconciliation step does more than choose one sentence. It changes temporal validity, scope, and activation priority. In safety attribute terms, the older memory may be downgraded to historical evidence, while the newer memory becomes active preference for a date range. If this safety attribute update is not explicit, later activation can resurrect stale authority.

### 4.5 Forget: Invalidating Influence and Descendants

When modification cannot restore a legitimate role, forgetting should withdraw the object's influence. Removing, expiring, quarantining, or rolling back state is therefore an influence action rather than a storage action [187]. Later memory-poisoning and sleeper-memory work make persistent query-to-memory influence and delayed activation concrete [147, 174]. We infer that copies, summaries, and derived rules must be checked separately rather than assuming removal of the source also removes their influence.

Once retained state has produced summaries, rule edits, cached plans, or invoked skills, this survey treats those derived objects as additional activation sites. The recovery unit is therefore the discovered descendant set rather than only the original record, with memory-security work supplying the broader carrier and threat vocabulary [103].

For benign lifecycle changes, a time sensitive preference may expire, a temporary permission may be revoked, or a stale skill may be retired. In each case the object moves to an expired, revoked, or archived state; its authority over the current action drops to zero; and its future influence should be limited to audit or historical reasoning. SAVER therefore treats forgetting as more than deletion. The operation must answer whether the influence was actually withdrawn from the places where future decisions are made.

### 4.6 Propagate: Copying Influence Across Actors and Contexts

Before or after activation, propagation can copy state to another agent, memory segment, workflow stage, artifact, or context. Multi-agent surveys explain why shared state and delegated context are routine, while self-evolving-agent surveys place that movement inside a broader adaptive lifecycle [8]. Architectural analysis relates compromised memory to inter-agent exchange, and ACIArena measures cascading injection across trust channels [6, 182]. MAPLE-Guard then localizes the memory route as write, retrieval, promotion, and cross-agent reuse: a poisoned private memory can enter shared memory and influence an agent that never

saw the source attack [212]. Propagation therefore changes both the number of activation sites and the receiver's authority assignment; it cannot be measured only by inspecting visible inter-agent messages.

EVOMAL supplies a skill-to-skill propagation case. A retrieved malicious skill is imitated into an agent-authored descendant, and rolling library replacement lets that descendant become a later template. After the eight planted seeds are removed, Qwen3 reaches 68% ASPR at round five, Gemma4 reaches 29.4%, and Devstral remains infected, whereas DeepSeek-V4-Pro and MiniMax return immediately to baseline and GPT-OSS decays [206]. The result is therefore conditional evidence of self-sustaining propagation under the paper's five-round replacement protocol, not a universal worm claim. It also makes source deletion an incomplete recovery test: the relevant unit is the retrievable descendant population after the seed disappears.

Recent multi-agent evidence separates propagation from control-flow hijacking. In the former, content or a practice crosses actors and is taken up by another agent; the live-laboratory cases in Agents of Chaos include cross-agent propagation among several other operational failures [164]. In the latter, untrusted content changes the active invocation edge. Triedman et al. show that an orchestrator can be induced to invoke unsafe agents or functions even when the individual agents resist the malicious request [184]. The transferred object is therefore not always a belief or memory item. It can be a routing decision that grants a new component authority. Recording both the content path and the invocation path prevents these two operations from collapsing into a generic prompt-injection label.

If a planner sends a worker the message "the customer may prefer email; verify before sending," the propagated state remains advisory and scoped. If the same content is copied as "send email to the customer," propagation has transformed authority even without changing much surface text. The receiver now sees an action instruction. SAVER records the sender, receiver, copied content, retained safety attributes, and permitted activation modes so that later exposure can be traced to the propagation path.

A state item copied into another context can outlive the original context's governance decision, making propagation a multiplier for recovery. If a memory is later corrected, the correction must propagate too, or the system will carry divergent safety attributes. For multi-step agents, intermediate plans, observations, and memory updates can pass through several stages before an external action occurs. The operation record should therefore name where state went and which future activation sites now need the updated safety attributes.

### 4.7 Migrate: Moving State Across Authority Regimes

Propagation multiplies copies; migration changes the regime under which a copy can be used. A migrate operation moves state into a substrate with different persistence, privacy, actionability, or governance semantics. MemGPT makes movement between memory regions visible [139], while self-evolving agent surveys ground broader substrate movement [8]. A feedback note moved into long-term memory becomes more persistent; a memory item converted into a skill or workflow rule gains authority to act; a local observation moved into a shared artifact gains a new audience [189]. Model updates are adjacent rather than identical, but self-evolution surveys establish substrate movement as a major axis of agent adaptation [8].

Model operations are special cases of these primitives with larger activation scope, but they are treated as central only after the agent loop test is satisfied. Fine-tuning and adapter training are add or modify operations over model state; post-training and preference optimization modify reward or policy behavior from feedback; merge combines adapters or patches and may change scope, authority, and safety regression attributes; edit and unlearn modify or remove behavior whose residual influence can be hard to observe. Existing model safety papers show why these operations deserve attention: fine-tuning can erode safety behavior [150], PEFT updates create scoped but deployable risk objects [60], and dormant behavior can survive until the relevant activation context [47]. The minimal acceptance rule for this survey is stricter: a central model row should identify the update object, the activation route, the affected task family, a regression probe, and a rollback handle. In SAVER records, a central model operation should therefore name the training data provenance, adapter scope, deployment lineage, merge status, activation route, safety regression result, and rollback handle.

Two migration patterns matter most here: note → rule, where feedback from one episode becomes planning authority, and memory or trajectory → skill, where stored experience becomes executable capability. Voyager illustrates the capability mechanic. PoisonedEvolution isolates the preceding migration decision: attacker evidence must be included, attributed as reusable expertise, and realized in the generated skill even though the attacker cannot edit the skill bank [20, 189]. SkillJack then shows what the completed crossing can hide and preserve, because extraction can obscure malicious intent and leave the implanted behavior active after source-record removal [228]. SkillMisevo-Bench extends observation beyond the crossing: its clean reload separates authored skill state, later retrieval, and fresh-session execution, so successful migration and successful activation remain distinct events [125]. The migration gate must therefore check not only content correctness but also whether support is independent, whether the target substrate grants stronger influence, whether source constraints survive extraction, whether later activation is attributable, and whether rollback can still reach the migrated behavior.

**Representation And Influence.** Reflection, skill learning, and workflow compilation show that related guidance can be carried by a memory item, heuristic, callable skill, or gated policy [168, 189, 192]. Repairing one representation therefore establishes a scoped result: a summary, rule, or other carrier from the same lineage may still influence behavior. The literature establishes these transformation mechanics but does not yet provide a general test for functional equivalence across substrates. We treat cross-substrate reappearance as an evaluation question, using fixed downstream probes and lineage records rather than surface-form matching alone.

### 4.8 Composition Paths

A feedback note can be stored and later used to revise strategy [168]; managed memory can move information across context boundaries [139]; a behavior trace can become an invoked skill [189]; and an observation can shape the next action in an interleaved reasoning-and-acting trace [223]. The add–abstract–activate compositions used by SAVER are a comparative reading across these mechanics. Unsafe paths can follow the same structure, which is why operation records must carry safety attributes, authority, and future influence fields together.

Operations provide a comparable vocabulary for how reusable influence is created and changed. Add creates reusable influence; abstract changes what safety attributes survive compression; activate grants current authority; modify rewrites safety attributes and permissions; forget withdraws influence from objects and descendants; propagate multiplies activation sites; and migrate moves state into a new authority regime. The capability literature supplies the mechanics through reflection notes, managed memory, reusable skills, and thought-action traces [139, 168, 189, 223]. The safety literature then narrows the claim: memory poisoning anchors persistent unsafe reuse, privacy work anchors sensitive retained state, and prompt injection through tools anchors action-boundary activation [30, 174, 187]. Model update papers become direct anchors only when they document a reusable update that later affects agent behavior [150].

> **Takeaway.** Substrates name *where* reusable influence lives; operations name *what changes it*. Removing one object does not establish influence-level repair when an operation has produced another representation. Add, abstract, activate, modify, and forget have dense local evidence, while propagation and migration are less often instrumented after state has moved. Section 5 asks which safety attribute failed to travel and which gate should have preserved it.

## 5 Violation: Safety Attribute Failures

$$\mathbf{S} \times \mathbf{A} \rightarrow \mathbf{V} \rightarrow \mathbf{E} \rightarrow \mathbf{R}$$

**Violation Premise.** Given the carrier-operation pair and attribute delta identified above, the next question is whether the resulting role remains legitimate. This is the first directed step after **S**×**A**: a violation exists when provenance, user scope, sensitivity, freshness, activation permission, evidence basis, authority, budget, liveness, or an external-effect boundary becomes false, missing, too broad, or unusable. It is therefore the internal side of adaptive-state risk. Exposure is the later moment when this latent mismatch becomes observable.

The evidence enters through two complementary routes. GhostWriter and PASB separate storage from later activation through adversarial admission and ordinary claim commitment, making provenance and persistence failures visible [124, 183]. InjecAgent and programmable privilege control expose the authority question when state reaches a tool-using decision [167, 235]. These works are related by the failed attribute, even though their benchmark denominators differ. Together they produce a named violation and expected trigger; Section 6 uses that pair to identify diagnostic evidence. The detailed paper-level index appears in Appendix Table 10.

### 5.1 Provenance Loss and Safety Attribute Laundering

Provenance loss occurs when a state carrier loses the record of where it came from, who supplied it, which trust boundary it crossed, and what evidence justified its storage. In information flow terms, this is safety attribute laundering: an object crosses a transformation boundary and later appears with a cleaner or stronger role than its origin warrants. The carrier may be a long-term memory entry, a demonstration retrieved from a retrieval-augmented generation (RAG) store, a summary, a workflow note, a skill precondition, or a tool observation. The operation is usually add, abstract, activate, or migrate: untrusted content is admitted into reusable state, compressed into a shorter object, granted current influence after retrieval, or moved into a substrate with stronger persistence. The safety attribute failure is role opacity: future components cannot tell whether the content is user preference, environmental data, inference from the model, attacker controlled text, or verified policy. AgentPoison grounds the memory and knowledge base version of this problem: malicious demonstrations can be placed in a retrievable store and later influence agent behavior when a trigger query selects them [24]. GhostWriter adds the personal-agent version, where the attack succeeds because memory subsystems do not apply security-focused governance before hidden payloads become later retrievable state [183]. PASB supplies the non-injection counterpart: status promotion and attribution removal turn a conversational claim into apparently established personal state, while scope broadening lets that state govern contexts beyond the original exchange [124].

A state item enters through a low-authority channel but later appears in a high-authority role because provenance was stripped or flattened during storage, summarization, or memory rewriting. That is the internal mechanism of safety attribute laundering. The exposure trigger is later retrieval or reuse under a task that gives the object current influence; until that trigger fires, the failure is latent. The response hook is provenance stamping at write, source retention during summary, distrust propagation during retrieval, and refusal at activation to let state with unknown source become task authority. These hooks are design requirements rather than proven universal defenses, but they follow directly from state safety attributes that GhostWriter's memory-store setting [183] and InjecAgent's indirect-injection setting [235] make critical to safety.

### 5.2 Authority, Actionability, and Boundary Escalation

Provenance loss becomes operationally dangerous when it enables authority, actionability, or boundary escalation. The carrier may be a memory item, retrieved instruction, plan fragment, tool result, workflow rule, learned skill, feedback summary, or privilege token. The operation is usually *activate*, *migrate*, or *modify*: descriptive state becomes a planning constraint, readable context becomes a tool argument, a local preference becomes a broader rule, or a stored fragment becomes part of an access control bypass. What fails is role preservation across time. AgentDojo gives the integrated-tool setting in which untrusted tool outputs can hijack behavior; Progent supplies the response vocabulary for explicit privilege control; and FragFuse shows the temporal memory variant, where low-authority fragments are stored across interactions and later fused through retrieval so the current query appears cleaner than the activated state actually is [30, 154, 167].

AuthMem-Bench isolates the same family without changing claim content. Matched histories contain the same proposition and lead to the same later task, but only one source condition permits the target use; authority collapse occurs when consolidation makes those histories operationally indistinguishable [237]. This design separates the internal violation from its exposure: the violation is the false authority upgrade during abstraction, while the exposure is a later prohibited tool action after the collapsed memory is activated. Authority is therefore not recoverable from proposition text alone once consolidation has erased the source-use constraint.

This merged family keeps three previously separate labels under one invariant: lower authority state requires an explicit policy grant before it can become instruction, plan constraint, tool parameter, workflow rule, or boundary extension. A retrieved invoice may be evidence for an answer, while payment authority requires a separate authorization. A webpage may describe a task, while the user's objective controls instruction authority. A short answer preference may guide formatting, while global workflow policy requires stronger evidence. ReAct, Voyager, and Reflexion explain why this escalation can occur in ordinary capability loops: observations condition later actions, reusable skills condition execution, and feedback can become future strategy [168, 189, 223]. The exposure trigger is action selection, tool invocation, workflow routing, skill execution, or a later task where the shifted boundary changes a permission decision. The response hook is activation authorization plus periodic safety attribute reconciliation: check user, task, tool, external effect, state provenance, and original scope before granting influence. R-Judge and GuardAgent provide recognition and guard-reasoning signals, while Progent connects those signals to privilege reduction [209, 233].

**Safe Promotion Invariant.** Promotion is the special case in which $\delta_t^\star$ turns episode evidence into a more persistent, general, or action-bearing object. The transition is valid only if $c_t(\delta_t^\star)$ authorizes that role, the relevant $\ell$ attributes survive or are explicitly revised, counterevidence remains available, and lineage links the promoted object to its source and descendants. Task success is evidence of local effectiveness, not permission for general reuse. OEP makes single-experience transfer failure concrete, experience-driven safety work shows aggregate benign experience shifting the action–refusal boundary, and EvoBreak shows individually acceptable experiences becoming unsafe under composition [192, 217, 253]. Skill-oriented attacks then expose the cross-substrate step: PoisonedEvolution targets the attribution decision that grants recurring trajectory evidence procedural authority, whereas SkillJack tests the laundering and persistence of the resulting skill [20, 228]. SkillMisevo-Bench resolves the completed path into successive gates: all 21 evolved configurations author unsafe artifacts, 19 retrieve unsafe skills, 19 show benign-task contamination, and 15 retain fresh-session harm [125]. A configuration that stops before final harm is therefore not evidence of a safe promotion; the unsafe artifact may simply have failed to receive later activation authority. Utility-to-authority escalation names the local failure, while selection-induced boundary drift names the longitudinal diagnosis when individually admitted promotions cumulatively widen scope or authority. These mechanisms stay within the existing authority family rather than creating new top-level violation classes.

### 5.3 Privacy and Purpose Failure

Role widening can concern purpose and audience as well as action authority. Privacy can therefore fail before disclosure when sensitive state loses the attributes that made later use legitimate. The carrier may be explicit personal information in memory, inferred attributes, user-specific preferences, private tool outputs, conversation history, or summaries that merge several users or sessions. Add, abstract, activate, propagate, or migrate can store too much, broaden a profile, grant a sensitive item influence in an unrelated task, share it with another agent, or move it into a less protected artifact. The failed attributes are sensitivity, subject boundary, purpose limitation, retention constraint, or personalized risk context. Memory-privacy work grounds this state-level reading, while PerMemSafe shows how long-horizon user-specific memories can make an otherwise safe response unsafe in that user's implicit context [5, 187]. Differential Harm Propensity isolates another personalized boundary: mental-health disclosure changes measured harm and refusal behavior, so the evaluator must treat the disclosure as risk-bearing context rather than a neutral profile field [226]. PS-Bench isolates the converse boundary failure: an inherently harmful request can be treated as safe because truthful personal memories bias the model toward a benign intent interpretation [50].

Privacy failure precedes exposure when a memory store retains sensitive state without purpose, expiry, scope, or contestability attributes, even before the information is printed or acted on. The exposure trigger is later retrieval, personalization across sessions, tool argument construction, multi-agent sharing, or generated output that reveals or uses the sensitive state. Personal-agent work identifies persistent user context, devices, services, and interaction history as future-use carriers [100], while memory-privacy work makes the sensitivity risk concrete [187]. Requiring the privacy attribute record at both update and activation time is this survey's governance proposal.

### 5.4 Persistent Descendants and Incomplete Rollback

Once a provenance, authority, or privacy failure has been copied, repair has to follow its descendants. Persistent descendant failure occurs when unsafe, obsolete, or contested state remains influential because the system cannot withdraw it from every future activation site. The carrier can be a memory item, embedding index entry, summary, cached plan, skill, workflow rule, preference profile, or derived note. The operation is usually add followed by abstract, modify, propagate, or migrate; the later forget operation is incomplete. In ordinary systems language, this is a time-of-check/time-of-use repair problem: the original object may be corrected or deleted, but a later activation still sees a derived object as valid. Memory-poisoning and sleeper-memory work make harmful persistence concrete, OEP extends it to consolidated experience, and SkillJack provides a direct cross-substrate case: the abstract reports that 80.0% of skill-mediated attacks persist after the original poisoned records are deleted [147, 174, 192, 228].

Persistence itself is useful when safety attributes remain correct and revocation paths work. The violation arises when persistence outruns governance. A poisoned demonstration may be deleted from one store while its summary remains in a profile. A stale user preference may be corrected in a conversation but left in a workflow template. A temporary permission may expire while a derived skill still assumes it. The exposure trigger is any future activation site that still sees the descendant as valid. The response hook is invalidation over descendants: every abstracted, copied, retrieved, or migrated object should carry a source pointer and expiry or revocation relation. Rollback therefore means influence withdrawal across descendants, with object deletion as one repair action. Memory poisoning work grounds the persistence side [174], while self-evolution surveys ground the broader adaptive lifecycle side [180]; broad claims about complete rollback across deployed adaptive agents would require stronger evidence than the present corpus supports.

### 5.5 Operational Integrity Failures

Persistent descendants broaden the consequence from semantic error to operational integrity failure. A workflow rule, retry policy, tool template, planner state, retrieved instruction, self-evaluation loop, skill, shared plan, or propagated artifact can affect resource, liveness, auditability, and accountability. Activate, modify, propagate, or migrate may turn a local retry heuristic into a general workflow rule, trigger repeated searches, loop over paid APIs, or leave causal state active after a visible action is blocked. The failed attributes are resource budget, rate limit, cost ceiling, timeout, stop condition, trace link, and repair ownership. OWASP frames excessive agency and unbounded consumption as application risks; AgentDojo moves observation beyond final text; Agent Security Bench and Agent-SafetyBench broaden the evaluation vocabulary [30, 138, 240, 250].

Resource harms and audit gaps belong together because both block reliable operation after state becomes influential. A poisoned memory can cause a wrong action, repeated paid calls, or an incident that cannot be attributed to a specific memory transition. A workflow rule can preserve task utility while multiplying cost. A risk-awareness benchmark can identify danger while leaving the causal state transition unnamed. Existing work covers different parts of that failure chain: personal-agent work supplies the device and user-context setting, GuardAgent and R-Judge supply guard-reasoning and safety-awareness signals, Progent supplies privilege control, and EDDOps adds the lifecycle monitoring and maintenance frame [100, 167, 208, 209, 233]. The response hook is a combined operation record: bind each state item or workflow edge to budget, liveness, source object, safety attributes before and after transformation, activation decision, tool authority, descendants, and repair owner. The current corpus supports the need for this attribute, while direct adversarial benchmarks for economic DoS, resource exhaustion repair, and lifecycle auditability remain thinner than tool injection and memory poisoning evidence.

### 5.6 Model Safety Regression

The same attribute logic extends from explicit state to model-side state, but the carrier is harder to inspect and often broader in scope. Model safety regression enters this survey when an update to parameters, adapters, reward models, edited components, policy prompts, or unlearning artifacts weakens an attribute that a later agent relies on. The operation is central when the agent loop or deployment controller fine-tunes, trains adapters, post-trains, merges, edits, distills, unlearns, selects, loads, or rolls back that object for

later behavior. The failure can be alignment erosion, adapter authority escalation, reward drift, persistent backdoors, incomplete unlearning, or lost deployment lineage. Fine-tuning safety, PEFT risk, feedback poisoning, and dormant behavior mark successive parts of this path [47, 60, 150, 153]. They support central SAVER claims only when the carrier, activation route, metric, and agent-loop scope are explicit.

An adapter that looks benign may gain broader authority when loaded into a production agent. A task-local fine-tuning update may change refusal behavior outside the intended scope. A merged adapter stack may erase the safety assumptions under which each component was evaluated. An unlearning patch may remove a visible behavior while leaving residual influence under paraphrase or downstream fine-tuning. The same transition logic therefore applies to the model substrate. The exposure trigger is a later inference, tool use decision, refusal boundary, policy check, or generalization failure across tasks. Safe LoRA provides a bounded LoRA-weight intervention for reducing fine-tuning safety loss [61]; broad LLM security surveys catalogue model-level security and privacy risks [195], and backdoor surveys frame triggerable residual risk [255]. The adapter registry, deployment lineage, version rollback, and residual probes proposed here are governance fields needed to connect those local results to an agent loop.

**Takeaway.** These violations share one structure: a state carrier crosses an operation, safety attributes that should have been preserved become missing or too broad, a later trigger can expose the failure, and a response hook must act before or during activation, with response after harm as a fallback. Violations are therefore *latent*: they exist before any user sees a bad output and may sit dormant across many transitions. The six families above name what has gone wrong inside the state ledger, independent of where the failure later manifests. Section 6 asks where they surface; Section 7 asks what governance trace can repair them.

## 6 Exposure: Surfaces and Manifestations

$$S \times A \to V \to \mathbf{E} \to R$$

**Exposure Premise.** Once a failed attribute and expected trigger are known, the analysis turns to where the latent mismatch first becomes observable and what record connects that event to the earlier carrier and operation. Exposure is diagnostic evidence, not a second violation taxonomy. The distinction prevents latent unsafe state from being equated with realized harm and prevents visible harm from being attributed only to the final output step.

The recurring bridge cases are memory bypass, prompt injection, economic denial of service, model update, multi-agent uptake, persistent privacy, and boundary drift. Each starts from an internal safety attribute failure, but becomes measurable at a different boundary: delayed retrieval, fused query, tool execution, budget consuming loop, uptake across agents, stale personalization, or mismatched governance signal.

**Surface Reading.** For each surface, the relevant question is what a monitor, evaluator, or incident record can observe. A useful record names the trigger, visible symptom, activated state object, and response handoff; the failed attribute remains the responsibility of Section 5. The chapter outputs this diagnostic tuple to Section 7, where the implicated object and lineage determine the scope of intervention. Appendix Table 11 retains the detailed evidence-role index.

### 6.1 Memory Retrieval Exposure

Memory retrieval exposure is the diagnostic surface for latent failures over stored or indexed state. The visible event is a later task selecting a memory entry, retrieved demonstration, profile note, summary, or derived preference and granting it influence. Direct attacks reach that event through different paths. FragFuse stores fragments whose prohibited meaning emerges only after fusion; stealth memory injection and GhostWriter target external-content admission before later activation; AgentPoison poisons retrievable examples selected by a trigger [24, 154, 183, 248]. Their common surface is delayed retrieval, but their causal operations and suitable controls differ.

Adjacent work broadens the diagnostic object. MemoryGraft studies poisoned experience retrieval, OEP studies harmful transfer from a consolidated local experience, and Tool-Drift follows remembered state

into routing [29, 172, 192]. GLOVE examines disagreement between memory and environment, whereas MemPot treats extraction as the relevant defensive surface [200, 227]. Privacy work adds retrieval that is semantically correct but inappropriate for the current user or purpose [187]. PASB clears the originating conversation before a neutral query, which makes later sycophantic behavior evidence of durable-state activation rather than residual dialogue context [124]. PS-Bench adds a different exposure: benign memory retrieval changes inferred intent and the visible symptom is unsafe compliance rather than a corrupted memory record [50]. These studies should be compared by the state selected and the authority it receives, not by treating every later-memory failure as the same attack.

A memory can sit inert until a query, user profile, workflow stage, or tool planning step matches it and activates it. The exposure trigger is that later match, not the original write. At that point the manifestation may look like a wrong answer, an unsafe recommendation, a personalized disclosure, or a tool parameter that seems to come from the agent's own reasoning. Memory retrieval exposure is therefore easy to misdiagnose as model failure. The visible behavior occurs at generation time, but the root cause may be an earlier state update whose safety attributes were not preserved. A retrieval decision should therefore filter or downgrade memories by provenance, sensitivity, expiry, and allowed use before they enter current context; a separate activation decision should determine whether the retrieved item may act as evidence, preference, plan constraint, or tool argument. Retrieval alone should not imply authority.

**Retrieval Record.** Managed memory supports context across interactions [139], while feedback reuse creates another recurrent-state mechanism [168]. Carrying safety attributes and checking activation scope are additional governance requirements introduced by this survey. PerMemSafe adds a personalized safety variant: the evaluator must know which remembered user-specific context made the later response safe or unsafe [5]. Because retrieval mediates authority, the diagnostic record must capture the retrieval decision and the prompt assembly that follows it.

### 6.2 Tool and Action Exposure

Retrieval becomes externally consequential at the tool and action boundary, where authority, actionability, and resource failures produce an API call, file write, email, purchase, database update, code execution, skill invocation, or repeated high-cost path. Lower-authority state now reaches a tool, skill, or API surface where it can change the world. ReAct gives the general mechanism in which observations and reasoning steps condition actions, while early prompt-injection work explains how external content can pressure that path [112, 223]. AgentDojo provides the clearest direct anchor for untrusted data reaching integrated agents, and InjecAgent extends the pressure to benchmarked agent settings [30, 235]. AuthMem-Bench supplies a controlled bridge from the earlier memory violation to this surface. Its action module keeps the claim, tool schema, and target argument fixed while changing whether the stored representation retains source authority; prohibited tool execution then measures the external consequence of an authority-collapsed memory [237]. The benchmark therefore localizes **V** at consolidation and **E** at action rather than treating the final tool call as an unexplained model failure. HarnessSafe generalizes this evidence rule across persistent carriers. A run reaches its violation stage only when a case-specific oracle confirms an executed behavior, committed state change, propagated artifact, or external effect; suspicious text, unsafe planning, or stated intent is insufficient [243]. Exposure is thereby tied to both the carrier path and observable consequence, while the furthest preceding trace stage shows whether the chain failed before or at the action boundary. Newer work expands this surface along three axes rather than forming one leaderboard. Adaptive attacks and LivePI test whether defenses survive changing or live interaction patterns, while SnapGuard isolates screenshot-mediated injection in web agents [38, 236, 251]. Task Shield and SafeSearch focus on task alignment in action or search settings; Tool-Drift and prompt tool selection examine how remembered state or connector descriptions alter routing [29, 35, 71, 166]. The shared outcome is an action-boundary event, but each line of work perturbs a different upstream object. The response side has a different role. Signed-Prompt makes source authenticity explicit, design pattern work frames defense composition, LlamaFirewall anchors runtime guardrail placement, Progent anchors source authorization and privilege scope, and formal agent security work frames data isolation and authority boundaries [12, 26, 167, 170, 175]. AGENTVIGIL is retained as a

red-team expansion entry, while IPIGuard is retained as a dependency-structure expansion entry; neither is treated here as precise evidence for defense effectiveness [4, 202].

The safety attribute failure is usually authority escalation, actionability escalation, or resource boundary loss. A webpage, email, retrieved document, or memory item is treated not just as information but as a command or parameter. The exposure trigger is the point at which the agent selects a tool, fills an argument, invokes a skill, commits an external effect, or opens a repeated high-cost call path. This trigger can occur without a dangerous final answer. The user may never see the injected instruction, and the model may never print the unsafe state, but the action can still expose the violation through an irreversible action, a denial-of-service loop, or economic waste. Authorization before execution should bind each action to current user intent, allowed tools, argument provenance, external effect, privilege state, and budget envelope; block, downgrade, or request confirmation when lower-authority state tries to become action authority or consume resources.

Risk awareness benchmarks can identify situations relevant to safety [233], and guard agents can reason about local responses [209], but the tool/action surface is governed only if the signal changes execution: remove suspect state, reduce privileges, force a clean plan, route to human approval, or record the failed authority transition for later repair. Guard signals alone are not enough.

**Context-Horizon Exposure.** Multi-turn attacks make the accumulated interaction, rather than the latest user message, the diagnostic object. Trojan Knowledge composes knowledge gathered through benign subqueries, while One Turn Too Late identifies the earliest response whose delivery closes a hidden harmful intent [165, 205]. MT-AgentRisk carries this decomposition into tool use [98]. Relinking reaches a comparable closure through summarization rather than dialogue order: distributed fragments become actionable only after compression [113]. Across these mechanisms, exposure begins when the combined state first supports a harmful response or external action. A turn-local refusal score cannot localize that point without the interaction prefix, any compression step, the candidate response, and the tool decision that follows.

### 6.3 External Artifact Exposure

Some outputs of tools and workflows persist beyond the episode, shifting exposure to an external artifact. A generated document, code patch, configuration file, database row, ticket, report, memory export, workflow template, shared prompt, or skill entry can later influence users, agents, or workflows. Self-evolution and multi-agent work establish why durable artifacts and shared environments belong in scope, while MemGPT and Voyager supply adjacent mechanics for memory movement and reusable skills [8, 139, 189]. Across these forms, scoped or descriptive information becomes durable evidence or procedure in another context.

A human reads the generated document, another agent retrieves it, a workflow uses the template, a developer executes the code, or a future run imports the artifact as trusted context. Artifact consumption is the exposure trigger, so the harm may occur after the original episode has ended. A stale summary can become a project rule. A private detail can become part of a report. A generated skill can encode an unsafe shortcut. A workflow template can preserve a boundary drift that later users never see as a choice. This surface is especially important for self-evolving agents because evolution often works by accumulating artifacts: reflections, skills, memories, notes, plans, and reusable procedures.

Release control for external artifacts starts with safety attributes rather than with final output screening. External artifacts should carry provenance, sensitivity, intended audience, allowed reuse, expiry, and derivation links. When the artifact can authorize action, it also needs an execution precondition and revocation path. When it is imported back into memory or a workflow, the original safety attributes should travel with it rather than being erased by the file boundary. Reflection systems provide reusable feedback mechanics [168], skill-library systems provide reusable procedure mechanics [189], and the self-evolution survey catalogues iterative capability-update settings [180]. Treating the resulting artifacts as governed, provenance-bearing objects is our extension. SAVER turns artifact-specific attack prevalence and complete artifact governance into explicit governance gaps rather than leaving them as implicit assumptions.

## 6.4 Propagation Across Agents

When state crosses actors, the visible event is one agent's unsafe state becoming another agent's context, belief, instruction, or action precondition. Self-evolution and trustworthy-agent surveys define why reusable shared state belongs in the safety setting; multi-agent surveys and coordination benchmarks identify messages, delegated tasks, shared memories, plans, and summaries as transfer channels [8, 229].

ACIArena turns those architectural channels into a cascading-injection benchmark over external inputs, profiles, and inter-agent messages [6]. Agents of Chaos supplies a deployment-oriented counterpart: its two-week live-laboratory study reports eleven representative cases that include cross-agent propagation of unsafe practices, information disclosure, destructive actions, identity spoofing, and partial takeover [164]. These cases establish that the transfer channels matter under realistic combinations of persistent memory, communication accounts, files, and shell access, but they are case evidence rather than prevalence estimates.

Control-flow hijacking exposes a different part of the same multi-agent boundary. Triedman et al. show that adversarial external content can redirect orchestration and communication toward unsafe agents or functions. With GPT-4o agents, the evaluated Web attacks achieve arbitrary-code-execution rates of 58–90% across orchestrators, and some model-orchestrator configurations reach 100% [184]. The attack can succeed even when individual agents resist direct and indirect prompt injection or refuse the harmful action. Thus, receiver compliance is not the only failure condition: the orchestrator can assemble a harmful execution from components whose local behavior appears safe.

The literature now exposes three related but distinct multi-agent attack forms. Message injection gives untrusted content instruction status; state or practice propagation changes what later agents retain or repeat; control-flow hijacking changes which agent or function receives authority. Embedding-defense and MAS-security work locate weak shared-memory and message boundaries, whereas MADRA, Devil's Moltbook, and collective misremembering examine uptake into plans, repeated exposure, or consensus [136, 188, 190, 214, 241]. A complete exposure record must preserve both content lineage and invocation lineage: source agent, recipient authority, selected edge, changed safety attributes, activation condition, and repair path.

Uptake occurs when a planner sends a worker a contaminated instruction, a summarizer converts a disputed memory into a shared fact, a debate participant repeats a wrong claim that sounds authoritative, or a coordinator treats another agent's uncertain output as verified. Once that happens, the original safety attribute failure can multiply: more agents can activate the state, more artifacts can derive from it, and recovery must locate every copy or descendant. The visible manifestation can be wrong consensus, repeated leakage, inconsistent authority decisions, or a group plan that no single agent would have produced alone. This surface therefore names the quantities a serious evaluation must recover: propagation path, recipient authority, uncertainty retention, source attribution, and repair completeness.

At the transfer boundary, messages and shared artifacts should carry sender, source state, confidence, sensitivity, allowed recipients, and activation permissions. Receivers should not automatically upgrade received content to local authority. Shared memories should support quarantine, correction propagation, and descendant repair. A multi-agent monitor should ask whether an individual output is safe and whether a safety attribute state item has crossed a role boundary or created inconsistent authority across agents. The requirement remains architectural, with limited validation across the field.

## 6.5 User-Facing Misattribution

After state has crossed retrieval, action, artifact, or agent boundaries, the visible failure can be attributed to the wrong causal layer. The user sees a generated answer, rationale, refusal, recommendation, tool result, or explanation, while the agent has composed that surface from model priors, retrieved memory, task context, tool outputs, and policy state. A LangGraph plus ChromaDB study makes the memory-side attribution problem explicit by showing poisoned shared memory mistaken for model misalignment [2]. InjecAgent supplies the action-side counterpart: untrusted external content can alter tool-mediated behavior [235]. Misattribution arises when the incident trace reports only the final answer or action rather than the state and source that shaped it.

A user asks why the agent acted, and the system provides a fluent answer that does not name the activated memory, retrieved document, tool output, or privilege decision. Explanation without trace is the exposure trigger. The harm includes misdirected incident response, not just epistemic confusion. Operators may retrain a model when they need to quarantine memory. They may strengthen an output filter when they need a retrieval gate. They may blame a user instruction when the decisive input was untrusted tool data. Causal attribution requires every answer shown to a user that depends on adaptive state to expose, at least to an auditor, which state carriers were activated and what authority they received. Public explanations can preserve privacy and stay summarized, but internal incident traces need enough detail to support repair.

A user cannot correct a stale preference if the agent presents the behavior as a general model tendency. An operator cannot revoke a poisoned descendant if the trace stops at the final output. A safety team cannot compare guard failures if the system records only blocked or allowed, not why the state was allowed to influence the decision. Misattribution is therefore also a contestability problem. Risk awareness work can support diagnosis signals [233], and guard agent work can support local response reasoning [209], but accountability to users requires that these signals connect to state traces and repair handles. SAVER's contribution is to make the causal chain auditable rather than treating misattribution as only a user interface problem.

## 6.6 Governance Blind Spots

Misattribution becomes a governance blind spot when a safety mechanism observes the wrong layer, acts at the wrong time, or cannot repair the causal state. The relevant carriers are now monitor signals, guard decisions, privilege policies, audit logs, and recovery records. R-Judge supplies safety recognition over trajectories, and TrustAgent performs trust-aware planning [64, 233]; AgentDojo supplies adversarial tool-use episodes at the exposure boundary [30]. GuardAgent and LlamaFirewall move from recognition toward runtime intervention, Progent constrains privilege at tool boundaries, and EDDOps asks whether these controls remain connected to deployment practice [26, 167, 208, 209]. The blind spot persists when a correct signal stops at the visible event and leaves the causal state active.

A benchmark records that an agent recognizes risk but the agent still lacks a rollback path. A guard blocks one response but the poisoned memory remains retrievable. A privilege policy prevents a high-risk API call but does not invalidate the workflow rule that requested it. An output filter catches a leaked string but does not update the privacy attributes on the underlying memory. The exposure trigger is a mismatch between signal and control: governance sees an event, not the adaptive state chain. Effective monitoring has to connect signals back to state operations: quarantine the activated item, update safety attributes, revoke descendants, reduce privileges, require human review, and emit an audit record that future gates can read. These hooks define the lifecycle response target. Existing guard, benchmark, and privilege control papers support local mechanisms and evaluation signals, while complete longitudinal governance remains a major open governance problem. A fuller response would need update admission, migration gates, activation authorization, containment, rollback completeness, deletion verification, contestability, and multi-agent repair.

## 6.7 Model Deployment Exposure

The same diagnostic problem becomes broader for model-side state because one loaded update can affect many tasks. Exposure occurs when an adapted model, adapter, policy component, reward model, edited checkpoint, unlearning patch, or safety classifier changes behavior after entering the inference or routing path. Fine-tuning safety identifies erosion after tuning; PEFT, feedback-poisoning, and dormant-behavior work add deployable update objects and delayed triggers [47, 150, 153]. Text-encoder and plugin attacks show related compromise of shared model artifacts [19, 97]. Stronger agent-loop comparisons still require direct activation and deployment-lineage evidence.

A memory item may expose only when a matching task retrieves it; a deployed adapter can expose whenever a compatible prompt, tool route, or downstream task hits the changed behavior. The model-side trigger is often broad rather than local. The manifestation can be a global shift in refusal boundary, unsafe compliance across tasks, persistent over refusal, systematic bias in tool selection, or policy activation across deployment. This makes misattribution especially likely: the visible behavior may look like an ordinary model limitation

unless the system records which adapter, checkpoint, policy prompt, safety classifier, and reward model were active. Deployment governance therefore needs an adapter registry, model card or safety card, training data lineage, model difference review, canary safety evaluation, task scope restrictions, version rollback, and residual influence tests. The response literature again splits by layer: Safe LoRA and LoRAShield intervene at the LoRA or adaptation layer, AEIOU shows prompt-moderation-style intervention in a related generative setting, and broader security and backdoor surveys supply protocol and benchmark vocabulary for persistent triggerable behavior [18, 61, 198, 255].

Across these surfaces, memory work most clearly measures delayed activation after a write; tool and prompt injection work most clearly measures the external effect boundary where information becomes action; model update work measures broader behavior shifts after an update is deployed into the agent loop; and multi-agent work exposes propagation and uptake, but still lacks mature evidence for descendant repair. User-facing misattribution and governance blind spots explain why exposure diagnosis must include causal state traces rather than final outputs alone.

**Takeaway.** Exposure surfaces are where latent violations become measurable events. They are not additional failure classes: the same violation can surface through many surfaces, and the same surface can reveal many violations. Tool and memory surfaces are best instrumented; multi-agent propagation, model exposure across deployments, and user-facing misattribution remain less mature. The diagnostic output is the visible event, activated object, failed attribute, and known lineage. Section 7 asks how an intervention uses that tuple to contain the event and repair the responsible state.

## 7 Response: Governance, Monitoring, and Recovery

$$S \times A \to V \to E \to \mathbf{R}$$

**Response Premise.** Exposure analysis produces a diagnostic tuple: the visible event, activated object, failed attribute, and known lineage. Response quality depends on how much of that tuple the intervention addresses. A guard may block the current event; a recovery claim must also show that the implicated reusable state and reachable descendants no longer reproduce the same influence. GhostWriter's separated injection and activation phases and PASB's cleared-session reuse make the temporal interval concrete, privacy work adds purpose and deletion obligations, and InjecAgent identifies the action boundary at which local containment becomes measurable [124, 183, 187, 235].

**Pipeline Reading.** Response mechanisms separate by the object they control and by when they intervene. IPIGuard and SEVerA inspect dependency or synthesis conditions before reusable behavior is committed [4, 11]. Memory-oriented controls act on writes, retrieval, session risk, environmental mismatch, or policy-aware recall; SRM and AM-Sentry focus on admission and retrieval, whereas GLOVE, Governed Memory, and MemGovern extend the control object toward environment consistency and policy-governed memory use [178, 183, 227]. SSGM and formal agent security move closer to constraints over stored-state use and action authority [81, 170]. These mechanisms are complementary because success at one intervention point does not establish closure at the next.

The resulting pipeline asks why an update was selected, whether admission and migration preserved its attributes, what authority it received at activation, whether a local guard contained the visible event, and whether recovery reached descendants. The early stages have concrete memory, verification, privilege, and runtime mechanisms. Rollback, deletion verification, and contestable closure remain less mature because they require an incident identity and denominator that survive across stores and sessions.

Recent runtime protection and guardrail papers support local response metrics without yet addressing lifecycle recovery. Appendix Table 12 indexes representative response works by stage, role, controlled object, and control focus.

**Recovery Chain.** After the first harmful behavior is observed, the field can already replay bounded admission, activation, and even misdiagnosis, but it still rarely carries one incident id through descendant inventory, rollback completeness, deletion verification, and contestability for users. The reason is partly timing, since

response research is newer, and partly the proof burden: repair across sessions needs evidence that the same influence no longer survives through summaries, embeddings, workflow rules, shared artifacts, or model updates. Inventory comes first because rollback cannot be complete over an unknown set; rollback completeness is then judged over the discovered descendants; deletion verification asks whether repaired descendants still leave residual influence; and contestability turns that repair status into a closure record for a user or operator. This later recovery chain is where governance either becomes a state level accountability chain or collapses into disconnected local mitigations.

**Smallest Governance Slice the Literature Already Implies.** Across the literature, lightweight governance implies six artifacts: a candidate and promotion record, a state registry at admission, a migration ledger, an activation gate, an incident and descendant table, and residual probes after repair. These artifacts answer a deployment question: why an update was chosen, what state may persist, what may change substrate, what may become action authority, how exposed descendants are inventoried, and how a user or operator can verify closure.

## 7.1 Preventive Governance

*Admission control.* Admission control is the first intervention point where transient context becomes reusable influence. A self-evolving agent may propose a memory item, preference, reflection note, workflow rule, skill, policy update, or profile fact after an interaction. The gate acts before persistence and asks a stricter question than relevance: may this source, actor, and proposed durable form influence future contexts? A relevant instruction can still be outside the user's authority, outside the task scope, sensitive to privacy, adversarially supplied, or unsafe when stored for later reuse. GhostWriter's AM-Sentry turns write-time memory saving into an explicit response surface, while memory privacy work adds the sensitivity and scope classification that admission has to preserve [183, 187]. PASB shows why the same gate is needed for ordinary user content: the commit decision should preserve whether a claim is attributed, tentative, local to one exchange, or authorized as a durable preference or procedure [124]. The benchmark motivates this control point but does not evaluate rollback or deletion verification.

Skill admission is relational because adding one locally plausible package can complete a harmful capability already distributed across the installed set. ChainGuard operationalizes this check by reconstructing candidate-centered producer–consumer, artifact, state, context, and execution paths between a candidate skill and previously installed skills [234]. This is direct evidence for context-aware admission rather than isolated package scanning. The mechanism blocks formation of a detected chain at installation time; it does not repair artifacts created before detection or establish descendant rollback, deletion verification, or contestable closure.

What stronger admission claims add to the literature is not a vague approval, but a readable admission record: source object, durable form, source trust, scope, sensitivity, freshness, allowed task family, allowed tool influence, and revocation handle. Those fields are what let a system reject the item, admit it unchanged, admit a redacted version, store it as context without authority, or require human confirmation before future use.

AuthMem-Bench provides direct evidence for persisting and enforcing the authority field in that record. Natural-language source attribution reduces but does not eliminate prohibited actions, whereas structured authority labels provide a stronger signal; in the selected end-to-end pipeline, predicted labels reduce the observed unauthorized-action rate from 16.9% to 0.0% over 350 pairs while authorized task success changes from 39.7% to 40.0% [237]. This supports a write-to-activation contract in which the label is stored before retrieval and checked at use. It does not establish universal prevention or lifecycle recovery: labels cannot restore omitted memories, and the study does not test descendant rollback, deletion verification, or contestability.

Before a dataset, feedback trace, adapter, edited checkpoint, or unlearning patch can support a stronger governance claim, model-side admission needs the same review fields: provenance, adapter scope, base model lineage, intended task population, allowed tool contexts, safety regression suite, merge constraints, and rollback path. These are the fields that let a system refuse an update with unclear provenance, quarantine an adapter outside its intended scope, require canary safety evaluation before merge, or prevent a task-local update from becoming policy across deployment. Model-side sources fill those fields in stages: fine-tuning safety identifies the admission-time regression risk, dormant-behavior work makes activation context explicit,

PEFT and feedback-poisoning work tie the risk to adapter or update provenance, and Safe LoRA gives a bounded mitigation handle for LoRA-style updates [47, 61, 150, 153]. The artifact that matters is an adaptation admission record, not a generic statement that the model was fine-tuned.

*Promotion gates and longitudinal reconciliation.* A promotion gate acts between candidate generation and admission. It first applies $c_t(\delta)$ as an eligibility condition, then lets $\widehat{u}_t$ rank only candidates whose provenance, scope, authority, privacy, budget, and rollback requirements are satisfied. This ordering prevents task utility from acting as implicit permission. The gate records the candidates considered, the evidence used to accept or reject them, the promotion target, the authority and scope delta, and any failure or counterexample omitted from the decision. OEP directly motivates transferability checks and active counterexample search before experience consolidation [192]; evaluator coevolution motivates recording the evaluator, objective, and epoch that produced the choice [66].

PoisonedEvolution makes the evidence side of this gate measurable. Its Inclusion–Attribution–Realization decomposition separates whether attacker records enter the stream, receive reusable credit, and survive synthesis into a skill [20]. In one $n = 30$, $k = 3$ F1 setting, a preliminary provenance-diversity gate blocks all 25 single-cluster candidates, reducing post-gate SER from 25/25 to 0/25. This is bounded evidence rather than a complete defense: candidate grouping is assumed, the benign control is small, and coordinated contributors may imitate diversity. The durable requirement is therefore auditable support and counterevidence, not source count alone.

SafeEvolve governs both sides of the resulting skill lifecycle. At write time, a critic and paired deleter localize and narrow unsafe reusable instructions; at reuse time, lineage risk, retrieved-skill outcome attribution, and safety-aware retirement change which versions remain active [125]. Its component tests report no later retrieval of threshold-crossing skills after retirement in the evaluated AutoSkill and EvoSkill runs. This is direct evidence for repair, attribution, and influence withdrawal over the active skill object, but not descendant-wide rollback, deletion verification across derived artifacts, or user-facing contestability.

EVOMAL moves the corresponding gate from submission screening to the create path. Its counter-prompt acts when the agent reads and re-authors a skill, reducing the six headline-model ASPRs from 20.3–41.8% to at most 1.8% with zero callbacks; the worst reported extended cell is 6.7% [206]. The utility claim is narrower than solution correctness: the paper's Submitted rate records whether an agent emits a solution within 15 steps, while benign helper copying falls by as much as 11.7 percentage points. Its signed-quarantine gate is a stronger structural proposal, but the extinction result assumes unforgeable curator signatures and an admission log, prevents re-retrieval rather than repairing existing descendants, and does not stop a persistent attacker's initial seed from firing.

Counterfactual testing asks what the candidate changes before the system commits it. For a fixed probe set, compare behavior under $\pi_{\mathcal{A}_t}$ with behavior under $\pi_{\mathcal{U}_\psi(\mathcal{A}_t,\delta)}$. A candidate that preserves task utility but widens tool authority, suppresses required confirmation, weakens refusal, or overrides newer evidence should be narrowed, quarantined, or sent for review. Scope-preserving abstraction carries the same rule into summaries and skills: retain preconditions, user and task scope, temporal validity, authority assumptions, safety constraints, invalidation conditions, and source lineage.

Longitudinal reconciliation repeats those probes after later updates because a sequence of locally accepted promotions may change the boundary even when no single record appears decisive. The experience bank should keep relevant refusals, safe aborts, failures, corrections, and near misses alongside successful trajectories so later candidate sets do not erase the evidence defining the safety boundary. Reconciliation then compares the current $\mathcal{A}_t$ with the recorded baseline, inspects accumulated $\ell$ changes, freezes suspect descendants, and routes confirmed drift into the rollback and deletion-verification stages below. These controls are proposed governance requirements assembled from current mechanisms; the literature does not yet establish them as a mature end-to-end defense.

## 7.2 Transition and Activation Gates

*Migration and attribute preservation.* Once state has been admitted, migration control governs crossings between substrate, authority, privacy, or actionability regimes. The same content can be harmless as a private note, risky as a generalized summary, and unacceptable as a tool parameter or workflow rule. A migration gate acts when memory becomes profile, retrieval context becomes plan step, plan step becomes tool argument, or a local rule is shared with another agent. Provenance, sensitivity, scope, authority, freshness, and reversibility must either survive the transformation or force a downgrade, redaction, quarantine, or explicit approval. Personal-agent work defines the user context, and MemGPT makes memory movement visible [100, 139]. Multi-agent work shows the shared-state version, Voyager supplies the skill-library case, and self-evolution surveys provide the broader movement axis [189]. These papers establish what moves; evaluated controls for preserving attributes during the move remain sparse.

The migration trace prevents a common governance collapse: once state is compressed into a summary or policy like rule, the system may treat it as trusted even if the original source was narrow, private, or adversarially exposed. The trace that matters identifies the source, source safety attributes, target substrate, operation, actor, authorization context, environment context, safety attributes inherited, safety attributes strengthened, safety attributes dropped, and the reason for any downgrade or override. Migration control can prevent unauthorized authority upgrades and privacy scope drift, and it can detect when the target substrate cannot carry required safety attributes. Even with such a trace, semantic drift, hidden descendants, and later misuse remain hard to rule out.

*Activation authority.* After admission and migration determine where an object may reside, activation authority determines whether it may influence the current decision. The gate acts before state guides reasoning, appears in the prompt, selects a tool, fills an argument, invokes a skill, or constrains a plan. Read access and influence authority are separate decisions. A memory may be visible for background context but barred from authorizing an external effect; a retrieved instruction may guide a summary but not override policy; a skill may be callable for local computation but blocked from account-changing tools. Progent makes tool privilege explicit, and Traversal-as-Policy applies global and node-local gates to reusable workflow policy [95, 167]. CaMeL offers a complementary architectural control: it derives control and data flows from the trusted query and enforces capability policies when tools are called, so untrusted data cannot redirect program flow or authorize a disallowed flow [31]. These mechanisms turn action authority into a checked decision rather than an implication of retrieval.

DRIFT addresses the point at which a fixed plan must accommodate legitimate runtime change. Its Secure Planner produces a minimal function trajectory and parameter checklist; the Dynamic Validator checks deviations against user intent and read, write, or execute privilege constraints; and the Injection Isolator masks conflicting instructions before they enter the memory stream [91]. In SAVER terms, CaMeL primarily controls activation at the tool boundary, while DRIFT combines activation-time validation with isolation at the current-stream admission boundary. GuardAgent, TrustAgent, and LlamaFirewall add guard and trust-control machinery around related decisions [26, 209]. Together, these systems provide strong local authorization and containment evidence. They do not, however, show that previously stored cross-session state or its descendants have been rolled back, deleted, or made contestable.

Admission authority must also survive the interval between an accepted call and its external effect. AID-Guard keeps one durable authorization and reservation lineage through provider commit, an ambiguous outcome, retry, and recovery: it revalidates the immutable request and current provider state at commit, retains the reservation while the outcome is unknown, and permits release or one successor only after certified no effect and a delivery fence [181]. This is direct evidence for effect-lifecycle authorization under declared provider contracts. Its recovery prevents uncertain delivery from becoming a second delegation; it does not establish rollback of adaptive-state descendants, deletion verification, or contestable closure.

Multi-agent activation adds the choice of which agent may run. Control-flow hijacking can evade semantic alignment checks over inter-agent messages because a checker may use a brittle notion of task relevance or lack the execution context needed to judge the invocation [70]. ControlValve instead generates a permitted

control-flow graph and enforces the graph together with a contextual least-privilege rule for each invocation. This moves the decision from message-level relatedness to structural transition authorization. The generated graph and contextual rules are themselves policy state, however, so runtime agent additions or legitimate replanning still require provenance and governed revision. The mechanism blocks an unauthorized edge; it does not repair state or artifacts produced before the block.

MAPLE-Guard addresses the complementary path in which no malicious message needs to cross the active communication edge. It gates memory writes, retrievals, promotion from private to shared memory, and later cross-agent reuse [212]. In its two reported benchmark settings, the abstract gives within-paper ASR reductions from 38.2% to 0.9% on LongMemEval and from 34.7% to 0.2% on AppWorld, with corresponding multi-agent defense success rate increases from 54.0% to 74.3% and from 42.5% to 99.8%. This supports link-level gating across the memory lifecycle; it does not establish descendant-complete rollback or contestable closure after poisoned state has already propagated.

Activation is better represented as influence level than as a binary allow list. A state item may be visible only, summarizable, advisory for planning, eligible as a tool argument, authorized for action, confirmation required, or blocked. This prevents stale or low-authority state from becoming executable authority simply because retrieval selected it. Risk recognition, injection benchmarks, and memory screens operate at different points in this decision. R-Judge tests whether risky situations are recognized; AgentDojo and InjecAgent supply tool-use and indirect-injection contexts where recognition must affect execution; GhostWriter's retrieval screen applies the same question to poisoned memory activation; and adaptive attack studies ask whether the boundary survives attacker revision [30, 183, 233, 235, 236]. PS-Bench supplies a bounded context-sensitive response: its detection-reflection intervention checks whether personal context is legitimizing a harmful request before generation and, in the GPT-4o-mini experiment, reduces average attack success rate across the evaluated memory frameworks by about 27.4% [50]. It does not test tool authorization or stored-state repair. These signals become governance only when connected to authorization, blocking, escalation, or recovery. The activation decision log names the object, context of use, influence level granted, policy reason, and authority withheld; authorization alone does not establish downstream recovery.

Multi-turn and compression defenses intervene at different activation points. TurnGate is a response-aware temporal gate that asks whether a candidate response makes the accumulated dialogue harm-enabling [165]. KBRA instead protects the transformation boundary at which separated context becomes a new instruction [113]. ToolShield acts farther downstream by exploring a tool's effects and distilling safety experience for later use [98]. Treating these mechanisms as interchangeable output filters would hide which state transition each defense actually controls.

### 7.3 Monitoring and Containment

If activation proceeds or exposure begins, governance moves from authorization to the live episode. Monitoring detects suspicious activation, instruction conflict, privilege escalation, privacy leakage, or unexpected tool arguments; containment limits impact by freezing suspect state, removing it from retrieval, downgrading authority, pausing external calls, forcing a clean replan, or escalating to human review. GuardAgent and TrustAgent supply guard and trust-control vocabulary, while R-Judge and AgentDojo provide recognition and injection settings in which those controls can be evaluated [30, 209, 233]. MAGE brings persistent safety context into the pre-action loop, and EDDOps addresses maintenance after deployment [201, 208]. Together they connect detection, bounded containment, memory-aware context, and operations, but not descendant repair.

A containment trace that identifies the activated object, any descendant object, monitor signal, tool or output at risk, containment action, state removed from the active context, and whether future activations are frozen pending review can distinguish the visible incident from the durable cause. This prevents a guard from being credited with more than it did: blocking one output is a runtime response, while repairing the memory, summary, workflow rule, or skill that caused the exposure is a rollback claim.

HarnessSafe operationalizes the first half of that distinction. Its stage distribution records how far a chain progresses across seven persistent-carrier families, and configurations with similar endpoint attack-success rates can stop at different stages [243]. This is stronger evidence for containment location than a binary block

rate, but it remains a trace of where progression ended. It does not show that implicated descendants were repaired, that deletion was verified, or that the failure stays absent after adaptation resumes.

Connected traces also require an independently governed control and evaluation plane. The task-facing agent may need to see the declared environment, permitted tools, and escalation or abort contract, while enforcement credentials, tamper-resistant audit state, and recovery records remain outside its write authority. Complete invisibility is neither required nor sufficient. The OpenAI–Hugging Face incident supports strict evaluation isolation, narrow trust boundaries, short-lived credentials, and cross-system correlation as infrastructure controls [83, 137]; it does not establish that every governance function must be external to or hidden from the agent.

For model state, monitoring includes safety regression and canary evaluation rather than only runtime guard signals. A deployment should test whether an adapter, merged model, post-trained checkpoint, edited component, or runtime cache changes refusal boundaries, unsafe compliance rates, excessive refusal, privacy leakage, tool selection behavior, or trigger sensitivity before and after activation. Safe LoRA illustrates a response at the weight level to LoRA fine-tuning risk [61], while KVEraser illustrates a response at the cache level when a harmful or stale span is identified only after prefill [93]. A deployed governance chain still needs versioned comparisons, cache-local residual probes, and rollback records tied to a specific model lineage, adapter manifest, or cached context.

### 7.4 Recovery and Contestable Closure

*Rollback and descendant recovery.* When containment confirms that a transition is unsafe, stale, contested, or outside its authority, response shifts from the live episode to rollback over lineage. Later memory-poisoning and sleeper-memory studies establish persistent influence and delayed activation [147, 174], while GuardAgent establishes local recognition and blocking [209]. Their separation motivates, but does not itself prove, the survey's rollback requirement: the target should include original memory, summaries, retrieval indices, workflow rules, skills, permissions, cached plans, shared artifacts, and external outputs. Quarantine and lineage capture should precede deletion so repair does not erase the evidence needed to reconstruct the incident.

The denominator for rollback completeness is the discovered descendant set, not the single object first reported. A stronger rollback claim therefore reads as a repair sweep: enumerate reachable descendants, classify their substrate and authority, apply repair or downgrade actions, and run residual probes before reopening scope. This denominator is operational rather than ontological. It is reconstructed from reachable records, caches, retrieval stores, workflow edges, deployment manifests, and tested activation paths, so strong reports should state discovered scope, inaccessible scope, and residual uncertainty.

Removing an unsafe adapter, reverting a merged checkpoint, undoing a model edit, erasing a harmful cached span, or rolling back an unlearning patch follows the same rollback structure with different handles: deployment lineage, active adapter or cache state, evaluation deltas, affected endpoints, and probes after rollback inside the agent loop. A rollback is scoped, not global, unless the reverted deployment, cached policies, inherited adapters, surviving cache influence, and downstream agents all pass the relevant residual tests.

*Deletion verification.* After descendants have been inventoried and repaired, deletion verification tests whether removal eliminated reusable influence rather than one visible record. It is the receipt that follows rollback completeness: the system records what was checked, what remains outside control, and whether the incident path still reactivates through copies, abstractions, or synchronization surfaces. Many local defenses stop before this point because blocking or deleting one object does not show that summaries, retrieval indices, caches, workflow rules, skills, permissions, shared artifacts, or external systems stopped carrying the influence.

A convincing deletion verification report therefore inherits the rollback denominator rather than starting from a fresh object list. It identifies the incident id, target object, deletion reason, substrates checked, descendants checked, tests performed, inaccessible substrates, residual matches, and residual activations. For adaptive agents, this report has to follow descendants because the original object may no longer be the only source of influence. A summary may retain the private fact, a skill may encode a derived behavior, a profile abstraction may preserve the sensitive attributes, or a shared artifact may reintroduce the state during a future session. The result is a scoped verification report, not a global forgetting guarantee.

*Contestability.* The recovery chain closes only when a user, operator, or auditor can contest the state and change its future influence. Contestability must expose enough of the object's origin, safety attributes, migrations, activations, descendants, rollback status, and deletion-verification status to support a meaningful challenge. Memory-privacy work motivates sensitivity and scope review, PASB shows retained influence surviving the original exchange, and GuardAgent supplies a local reasoning anchor rather than a full contestability mechanism [124, 187, 209]. Fully realized contestability remains rare.

Contestability prevents adaptive state from becoming uninspectable authority after an internal repair sweep claims success. A useful dispute flow freezes or downgrades activation, attaches correction evidence, inspects the rollback inventory, reviews deletion verification receipts, and emits a resolution record visible to the user. Without that flow, rollback remains an internal maintenance action rather than an accountable governance process.

### 7.5 Synthesis: Governance Follows State Descendants

The response component closes the SAVER chain by relating mechanisms that otherwise appear as separate defense families. Memory-oriented controls address admission, retrieval, sensitivity, and purpose through write-time screening, privacy governance, and personalized risk-aware memory [5, 183, 187]. Action-oriented controls first address source authenticity and task alignment at the effect boundary [71, 175]. Risk judges and guards add recognition or local blocking, while Progent, CaMeL, and DRIFT connect those signals to privilege, control flow, and runtime deviation checks; EDDOps addresses their operational maintenance [26, 31, 91, 167, 208, 209, 233]. Model-side work follows the same logic with different handles: fine-tuning safety defines a regression surface and Safe LoRA supplies a bounded adapter-level intervention [61, 150].

**Response Feedback.** Response also creates state for later rounds. Evaluation evidence can revise operational memory, prompts, routing, guardrails, test cases, safety cases, or rollback records; EDDOps treats these as governed runtime or redevelopment changes [208]. DARWIN and BraveGuard instantiate two complementary guardrail-update loops. DARWIN evolves jailbreak strategies through discovery, mutation, selection, and feedback-driven composition, then trains successive guard checkpoints on the resulting adversarial samples [149]. BraveGuard instead converts open-world threat discovery into executable computer-use tasks, agent trajectories, and supervision for repeated guard training [42]. The former couples guard evolution to an adaptive adversary; the latter couples it to an expanding trajectory corpus.

For SAVER, these systems make the response itself an adaptation producer. In the model from Section 2, a guard update may alter $c_{t+1}$, the evaluator or objective behind $\widehat{u}_{t+1}$, or the candidates available in $\Delta_{t+1}$. Its transition record should therefore connect the discovered threat, generated task or attack, selected training example, guard checkpoint, and later deployment decision. DARWIN and BraveGuard establish that this feedback can improve guard performance under their respective benchmark settings, but they do not establish descendant-complete rollback, resistance to poisoned update evidence, or contestable closure across guard versions. EDDOps supplies the broader process edge; the guardrail studies make the evolving response state concrete.

These mechanisms cover successive response points, but they do not yet form a verified recovery chain. A connected claim must identify the implicated state, enumerate reachable descendants, report rollback over that discovered set, test residual activation after deletion or repair, and expose a contestable closure record. Without those links, a blocked action remains evidence of local containment rather than lifecycle recovery.

> **Takeaway.** Response evidence is least connected across the lifecycle. Local mitigations cover selection, admission, migration, activation, monitoring, rollback, deletion verification, and contestability, yet few systems bind these stages to one incident handle and denominator. A response may also write state that shapes the next selection round, so closure includes descendant repair and the later consequences of the response itself. Section 8 asks which traces make that recovery claim checkable across sessions.

# 8 Evaluation and Benchmarks

$$\mathbf{S} \times \mathbf{A} \to \mathbf{V} \to \mathbf{E} \to \mathbf{R}$$

**Evaluation Premise.** Existing benchmarks usually score task success, immediate safety behavior, attack success, or local intervention. These endpoints remain useful, but self-evolution creates temporal and spatial dependencies: a memory write, summary, skill edit, delegated message, or model update can remain dormant until a later context activates it, or move to another carrier or agent before activation. Evaluation must therefore identify the transition that produced the visible behavior, not only score the behavior itself.

The measurement object is a lifecycle trace. It binds the initial carrier and authority regime to the operation applied to them, records the safety attribute that failed, locates the first observable consequence, and tests the response against the implicated state and descendants. Without that trace, the same harmful outcome may be attributed to prompt admission, summarization, retrieval scope, tool authorization, or failed rollback. A local block also remains indistinguishable from repair if the enabling state can reactivate later or elsewhere.

The temporal dimension does not make all multi-turn scores comparable. MTID records the earliest response that makes accumulated dialogue harm-enabling, MT-AgentRisk constructs tool-realistic attack sequences, and Relink measures compression-induced recombination against clean-split controls [98, 113, 165]. Their denominators differ because they count response closure, tool-using attack sequences, and relinking events. They nevertheless show why interaction length should be analyzed through the operation that combines context rather than treated as a scalar risk factor. A3S-Bench complements this evidence by placing payloads in workspace artifacts outside conversation-level monitoring and evaluating the resulting action traces [122]. This directly tests a spatial entry path through files, skills, and configuration, but not general propagation across agents or deployments.

Selection adds a second temporal dependency before the measured transition. The Red Queen Gödel Machine treats the utility function and evaluator signal as epoch-scoped parts of the self-improvement loop rather than a fixed verifier outside it [66]. A lifecycle record should therefore name the agent, evaluator, objective, and epoch versions before comparing improvement, degradation, or repair. The section first audits selection and promotion, then reads existing benchmarks by the transition segment they observe, separates incompatible denominators, and closes with the traces needed for longitudinal recovery evaluation.

## 8.1 Selection and Promotion Evaluation

A selection audit observes the stage that ordinary transition benchmarks usually hide. At each evolution step it records $\Delta_t$, the source of each candidate, the evaluator and objective version behind $\widehat{u}_t$, the evidence checked by $c_t$, the selected update $\delta_t^\star$, rejected candidates, and the resulting change to $\ell$. The benchmark then delays activation, runs the updated state on tasks outside the source episode, and records whether the consequence enters the next candidate set. OEP supplies a direct test of harmful transfer from accepted experience updates, while the Red Queen Gödel Machine supplies the denominator rule for changing evaluators and objectives [66, 192].

The protocol needs separate adversarial and endogenous tracks. The adversarial track inserts crafted updates and measures whether selection, activation, and recovery reject or contain them. The endogenous track uses ordinary successes, failures, refusals, and corrections without a malicious writer. It holds the probe suite and authority policy fixed while the agent evolves, then asks whether selection alone changes refusal boundaries, confirmation requirements, tool authority, privacy scope, or preference priority. This design does not assume that benign adaptation drifts. It makes the possibility falsifiable and separates it from attack success.

Three proposed metrics cover the first usable longitudinal denominators. Let $\mathrm{Promote}(\delta)$ mean that candidate $\delta$ enters a more persistent, general, or action-bearing role, and let $\mathrm{SafePromote}(\delta)$ mean that the promotion satisfies the invariant in Section 5. Safe Promotion Rate is

$$\mathrm{SPR}_{0:T} = \frac{\#\{t \le T : \mathrm{SafePromote}(\delta_t^\star)\}}{\#\{t \le T : \mathrm{Promote}(\delta_t^\star)\}}.$$

The denominator is all committed promotions; a run with no promotions reports the metric as undefined rather than perfectly safe. For Boundary Drift, let $\mathbf{b}(\mathcal{A}_t)$ be the decision vector produced by a fixed set of paired safety probes and let $d$ be a preregistered distance:

$$\mathrm{BD}_t = d(\mathbf{b}(\mathcal{A}_0), \mathbf{b}(\mathcal{A}_t)) .$$

For Negative-Evidence Coverage, let $\mathcal{N}_t$ contain safety-relevant non-success experience observed at step $t$, including correct refusals, safe aborts, failures, corrections, and near misses, and let $\mathcal{N}_t^{\mathrm{keep}}$ be the subset retained or consulted during selection:

$$\mathrm{NEC}_{0:T} = \frac{\sum_{t=0}^{T} |\mathcal{N}_t^{\mathrm{keep}}|}{\sum_{t=0}^{T} |\mathcal{N}_t|}.$$

NEC is conditional on negative evidence that was actually observed. A report should therefore separate *observed but discarded* refusals, aborts, failures, corrections, or near misses from evidence that was *never generated or unavailable* because the action space, environment, or reward design supplied no meaningful safe alternative. The first case diagnoses selection or retention; the second diagnoses the opportunity set. Treating both as low NEC would hide which mechanism needs repair.

**Satisfiability and Abort Calibration.** Benchmark tasks should cross two independent labels. Feasibility records whether the declared goal is satisfiable in the provided environment, blocked by a missing dependency, or infeasible under the declared conditions. Action authority records whether a candidate path is permitted, requires explicit escalation, or is prohibited. A blocked path does not authorize an out-of-scope workaround. Reports should pair the true safe-abort rate on infeasible tasks with the false-abort rate on tasks feasible within current authority; cases feasible only after authorized escalation should be scored separately for correct escalation. Both rates should be reported with task utility so an agent cannot appear safe merely by abandoning useful work. The OpenAI–Hugging Face incident motivates this separation: the ExploitGym objective was narrow, but task completion alone did not bound which infrastructure paths were acceptable [83, 137, 203]. The incident does not establish that an abort option would have prevented the intrusion; it shows why outcome success and path authorization need separate denominators.

These are measurement contracts proposed by this survey, not established cross-paper metrics. Reports should publish the probe set, epoch boundaries, missing candidate logs, task-feasibility labels, abort and escalation options, and denominator exclusions. The utility-risk covariance identity in Appendix B is a conditional diagnostic for the same protocol; it does not replace these observable counts.

**Post-Response Continuation.** An immediate residual probe does not settle what happens after adaptation resumes. After recording the response, a benchmark should continue ordinary evolution for a preregistered number of update opportunities and rerun the fixed safety probes. The report should state whether the incident lineage, a known descendant, or an equivalent failure under the fixed probe suite reappears, then name the substrate, the first reactivation point, and any inaccessible scope. Adaptive prompt-injection studies show why an action-boundary defense must be retested after attacker revision, while EDDOps motivates re-evaluation after governed system changes [208, 236]. These sources do not establish cross-substrate response-induced migration; continuation after response is a measurement contract proposed here. Endogenous and adversarial tracks can share the same logs, but candidate-source control remains part of the threat model.

## 8.2 Reading Existing Benchmarks Through SAVER

A lifecycle reading asks what part of SAVER a benchmark already observes and what remains hidden. AgentDojo-style indirect prompt injection evaluation is a clear example. It can show whether untrusted environmental content changes an outcome mediated by tools, so it is strong at the exposure surface: low-authority data reaches a tool-using decision and causes an unsafe or task-inconsistent action [30]. This is a recognizable access control and privilege escalation problem in an agent setting. The missing evidence is temporal: unless the test records state lineage, it cannot diagnose whether the injected instruction was merely present in the current prompt, was written into long-term memory, was summarized into a user

**Table 2** Benchmark family coverage along the reusable-state lifecycle, ✓ direct, ● partial, ✗ mostly absent.

| Family | Admit | Move | Act. | Expose | Contain | Rollback | Delete | Contest | Boundary |
|---|---|---|---|---|---|---|---|---|---|
| AgentDojo [30] / InjecAgent [235] | ● | ✗ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | Tool-mediated injection and exposure; no response or cleanup denominator. |
| ACIArena [6] | ● | ● | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | Cascading injection across profiles, messages, and agents; response and repair unmeasured. |
| HarnessSafe [243] | ✓ | ✓ | ✓ | ✓ | ● | ✗ | ✗ | ✗ | Connected carrier-to-violation trace and stopping stage; no selection or post-incident recovery. |
| SkillMisevo-Bench [125] | ✓ | ✓ | ✓ | ✓ | ✓ | ● | ✗ | ✗ | Skill authoring-to-carryover trace plus local repair and retirement; no descendant or deletion proof. |
| EVOMAL [206] | ● | ✓ | ✓ | ✓ | ✓ | ● | ● | ✗ | Create-path and five-round source-removal trace; no natural-stream or descendant-deletion proof. |
| R-Judge [233] / ASB [240] / Agent-SafetyBench [250] | ✗ | ✗ | ● | ✓ | ✗ | ✗ | ✗ | ✗ | Recognition, security tasks, and unsafe-choice scoring; no lifecycle response. |
| OEP [192] / PASB [124] / MemEvoBench [210] | ✓ | ● | ✓ | ● | ✗ | ✗ | ✗ | ✗ | Experience or memory updates and later reuse; no recovery denominator. |
| PerMemSafe [5] / PS-Bench [50] / RBI-Eval [213] | ● | ✗ | ✓ | ✓ | ● | ✗ | ✗ | ✗ | Personalized context and warranted-memory-use evaluation. |
| PersistBench [146] | ● | ✗ | ✓ | ✓ | ✗ | ✗ | ● | ✗ | Forgetting and use boundaries; no descendant-wide repair. |
| SafeAgent [106] / Progent [167] / CaMeL [31] / DRIFT [91] / LlamaFirewall [26] | ✗ | ● | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | Tool authorization and local containment, not descendant repair. |
| Fine-tuning safety [150] / PEFT safety [60] / SEVerA [11] / Dormant tests [47] | ✓ | ● | ✓ | ● | ✗ | ✗ | ✗ | ✗ | Model update, activation, and verification slices; no agent-loop recovery trace. |
| Safe LoRA [61] | ✓ | ✗ | ✓ | ● | ● | ✗ | ✗ | ✗ | LoRA update and bounded mitigation; no lifecycle rollback evidence. |

profile, was promoted into a workflow rule, or was later retrieved as trusted guidance. A transition reading keeps the original exposure test, then asks for a write log, source safety attributes, transformation record, activation context, and recovery probe after the episode.

R-Judge evaluates whether an agent recognizes risky situations, which is valuable for the violation and response vocabulary [233], but it illustrates a different boundary. Risk recognition and transition localization are separate measurement objects. A lifecycle reading asks whether the evaluator can connect the risk judgment to a state carrier and operation: for example, a stale memory, a low-authority retrieved instruction, a broadened tool permission, or a policy update loaded outside its intended scope. Agent Security Bench helps map agent security tasks [240], and Agent-SafetyBench maps broader agent safety behaviors [250]; they become transition safety benchmarks only when they record how state entered, changed safety attributes, activated, and was repaired.

Under this reading, attack success rate, risk recognition accuracy, and task success remain useful but stay surface scores. A transition report records the missing trace: which state carrier was involved, which operation changed its role, which safety attributes were supposed to travel, which gate saw it, which exposure occurred, and which recovery target was tested. The compact benchmark family coverage pattern in Table 2 lets the reader recover the benchmark logic without pausing on one paper at a time: what latent violation is planted, where it is activated, where it becomes exposed, and which response denominator is actually being measured. The matrix is deliberately qualitative so the field level asymmetry stays visible without turning incompatible scores into a leaderboard. Existing suites most often cover exposure, activation, recognition, and local response; they much less often show migration lineage, descendant rollback, deletion verification, or contestability. The appendix records the denominator rule behind this choice instead of reproducing a score list. Appendix Table 13 retains the corresponding paper-level benchmark index, including target system, observed lifecycle slice, and denominator.

## 8.3 Denominator Discipline Without a Leaderboard

Current quantitative evidence supports a narrow synthesis: benchmarks measure local transition slices with separate safety frontiers. Action-boundary attack success rate (ASR), safety-recognition F1 score, memory-poisoning success, runtime-intervention rate, and governance-routing score each answer a different question

about the adaptive-state chain. Quantitative comparisons are therefore used to identify measurement objects and opportunity sets, not to reproduce paper-specific percentages; a score list would imply comparability across benchmarks that the heterogeneous denominators here do not support.

Safe comparison is possible inside a shared benchmark denominator, transition slice, and measurement object. InjecAgent-style cases support exposure comparisons inside one indirect-injection benchmark; R-Judge-style annotated records support recognition comparisons; PASB-style contrasts support session-only versus committed-state comparisons; and SafeAgent- or Progent-style runtime rows support one controller family inside one harmful-task benchmark. Once those denominator families are mixed, several local evaluation panels collapse into a category error rather than a meta-analysis.

The denominator families become clearer when the non-adversarial and adversarial cases are interleaved. PerMemSafe asks whether user-specific memories make a later response unsafe under implicit personalized context [5]; PS-Bench compares harmful queries with and without benign personalized memories to test intent legitimation [50]; FragFuse tests access-control bypass after fragments are stored across memory interactions and reconstructed at retrieval [154]; GhostWriter separates hidden memory injection from later activation in tool-using personal agents [183]; AgentPoison tests trigger-selected poisoned retrieval [24]; OEP tests accepted experience updates and harmful transfer [192]; and EvoBreak tests adaptive acquisition followed by joint activation of complementary experiences [217]. These designs can all report ASR-like or safety-rate values, but their opportunity sets differ. The useful comparison is the denominator family: personalized context followed by response safety, fragmented injection followed by fusion, hidden memory adoption followed by later activation, trigger-selected retrieval, one accepted update followed by transfer, or an acquired experience set followed by composition. The mechanism difference matters more than the magnitude.

PASB supplies a particularly clean within-paper commit contrast. Its 1,600 tasks separate a five-turn persist stage from a cleared three-turn query stage and let the evaluated agents decide what to store. Across twelve models, the reported downstream failure rises from 45.0% in session-only episodes to 71.9% after commitment, a 27.0 percentage-point increase [124]. This result localizes risk at the transition from accepted conversational content to durable state; it is not a cross-benchmark ranking against poisoning or activation-only studies. The session-only baseline also removes the persistence mechanism whose personalization utility motivates the agent, so its lower failure rate is not evidence of a cost-free safety improvement. PASB-style commit contrasts should therefore pair safety outcomes with matched utility measures; PerMemSafe illustrates the coupled objective by evaluating implicit personalized safety together with helpfulness under long-horizon memory [5, 124].

Three recent studies add different within-paper transition contrasts. PoisonedEvolution measures evidence-to-artifact promotion: at the same 10% attacker-support ratio, it reports 546/600 embedded skills in SkillClaw and 369/600 in the structurally different Trace2Skill pipeline [20]. Its SER denominator is completed evolution trials and stops before runtime activation. SkillJack instead compares poisoned trajectories with extracted skills: in SkillX, the abstract reports safety detection falling from 98.5% to 11.4%, while 80.0% of skill-mediated attacks persist after source-record deletion [228]. MAPLE-Guard compares unguarded and guarded memory-link opportunities at write, retrieval, promotion, and cross-agent reuse on LongMemEval and AppWorld [212]. These are, respectively, artifact-promotion, representation-and-deletion, and guarded-lifecycle-link denominators. None should be ranked against PASB's commit effect, EvoBreak's experience-set attack, or runtime ASR.

SkillMisevo-Bench connects artifact promotion to later execution under fixed denominators. Each agent–method condition contains 25 episodes and 525 tasks: 225 malicious learning tasks, 225 benign tasks, and 75 clean-session persistence tasks. Across the 21 evolved configurations, unsafe artifacts appear in 21, unsafe retrieval and benign-task contamination in 19, and fresh-session harm in 15 [125]. These are successive gates rather than interchangeable rates. In its two-configuration exposure sweep, pooled carryover ASR rises from 16.0% at $K = 0$ to 35.3% after three malicious tasks and 41.3% after nine. Because the no-exposure baseline is already nonzero, the defensible result is the within-paper increase after bounded exposure, not that all observed carryover harm originates in the malicious tasks.

The same study makes the response tradeoff more precise than its abstract-level mean. Across OpenClaw with AutoSkill and EvoSkill, SafeEvolve reduces mean unsafe retrieval from 35.33% to 8.67% and carryover ASR from 21.33% to 4.00% [125]. Online benign utility changes from 58.44% to 58.00%, which yields the reported 0.4-point cost, but carryover utility changes from 53.33% to 40.67%. The result therefore supports a low immediate utility cost under the paper's online denominator while also showing a larger loss of useful persistent reuse. It is not evidence of a cost-free or universal safety–utility frontier.

EVOMAL adds a create-path and descendant denominator. ASPR counts tasks that store a newly authored skill reproducing a planted pattern; it is distinct from the callback rate that records payload execution. Across six models on 153 tool-relevant SWE-bench Verified tasks per model, headline ASPR ranges from 20.3% to 41.8%, producing 31–64 new malicious skills from eight planted seeds and leaving 4.9–9.0 times as many malicious skills as were planted [206]. The reported 86.7% targeted maximum is the Qwen3 pytest-family cell with $N = 15$, not the full task distribution. The five-round removal experiment is a separate rolling-replacement denominator and should not be collapsed with single-round ASPR or callback success.

ColluSkill adds a skill-set denominator. Its 200 three-skill attack chains reach 96.0% average ASR against six evaluated scanners; adding full installed-skill context to ChainGuard reduces the attack ASR from 69.0% to 22.5% while changing the reported benign-workflow pass rate from 99.7% to 99.5% [234]. These values measure candidate-with-context scanning on the paper's generated chains and benign workflows. Because the attack iteratively uses feedback from the target scanners that are later evaluated, the 96.0% result is evidence of adaptive scanner evasion under that feedback setting, not unseen-scanner generalization or ecosystem prevalence.

AID-Guard adds an authorization-to-effect denominator rather than another attack-success leaderboard. Its 210 frozen Stripe contract trials matched their predeclared outcomes; separate campaigns exercised 40 terminalize-then-successor schedules across Stripe and Resend, 30 overlapping Stripe confirm/cancel races, and 10 Stripe crash-recovery schedules without an observed duplicate effect [181]. The same paper makes the safety–utility cost visible: over 48 benign AgentDojo episodes per model and condition, the strict exact-manifest profile reduced benign utility by 43.8 percentage points for DeepSeek and 35.4 points for Qwen. In a separate no-seed rerun, a typed three-call authority form completed 10 and 9 more benign tasks than exact single use, while all 864 attack episodes produced no unsafe effect. These results remain paper-local: the provider claims depend on the declared effect-path inventory and supported contracts, and the no-seed frontier does not identify a causal effect size or an optimal policy.

The practical path for meta-analysis is a normalization layer over opportunity set, transition slice, and measurement object. Synthesis across papers becomes safer once papers report conversions such as activation opportunities to realized exposures, discovered descendants to repaired descendants, or seeded latent violations to residual activations after repair. Until then, exact values belong in source-specific extraction records, while each section records what its metric measures and which lifecycle stages remain unmeasured.

Admission filters, task alignment mechanisms, activation gates, runtime guards, memory governance controls, model mitigations, and process audits all support useful local claims, but they observe different opportunity sets. The same denominator rule applies in prose: a block rate near a tool boundary, a memory write defense, an adapter safety test, and a process audit for governance become comparable only after the same transition slice and denominator are visible.

### 8.4 Benchmark Patterns and Missing Traces

Most benchmarks remain locally strong but observe only one transition slice. Three connected designs now cover complementary scopes. HarnessSafe spans seven persistent-carrier families and assigns each executable case to its furthest evidence-supported stage; SkillMisevo-Bench follows one skill carrier from authoring through clean reload, retrieval, contamination, and carryover harm; and EVOMAL isolates create-path copying, callback exposure, and descendant propagation after seed removal [125, 206, 243]. HarnessSafe provides cross-carrier breadth, SkillMisevo provides skill-evolution depth, and EVOMAL adds an explicit descendant process. They still leave rejected candidates, natural deployment streams, verified descendant deletion, contestability, and reactivation after broader adaptation incomplete.

**Table 3** Lifecycle evaluation loop. A benchmark should seed state, transform it, trigger activation, measure exposure, verify repair, and continue adaptation before closure.

| **Lifecycle Evaluation Contract** | state + adaptation + challenge + response |
|---|---|
| Seed State → Adapt / Move → Trigger / Expose → Respond / Resume | |
| **1 Seed** | Initialize memory, workflow, tool, shared agent, or model carrier. *Trace:* id, source, substrate, environment, intended assets, feasibility label, authorization boundary, expiry, rollback handle. |
| **2 Adapt** | Add, abstract, activate, modify, forget, propagate, or migrate. *Trace:* changed safety attributes, gate decision, dropped fields, descendant link. |
| **3 Delay** | Insert distractor tasks or time gaps. *Trace:* order, access log, retrieval attempts, background updates. |
| **4 Trigger** | Challenge stored state and test whether it receives authority. *Trace:* task, feasibility status, policy, action authorization, retrieved item, tool, workflow, adapter. |
| **5 Expose** | Measure output, tool call, disclosure, delegation, artifact, monitor, or model behavior. *Trace:* transition attribution, not aggregate success alone. |
| **6 Repair** | Run containment, quarantine, rollback, user challenge, or operator review. *Trace:* affected state and descendants. |
| **7 Probe** | Probe residual influence after repair. *Trace:* rollback ledger, descendant checks, reactivation probes, deletion/unload evidence. |
| **8 Continue** | Resume adaptation for fixed update opportunities, then repeat the safety probes. *Trace:* response-generated state, first reactivation, substrate, lineage or equivalent probe failure, inaccessible scope. |
| **Outputs:** transition trace; exposure class; response decision; post-response reactivation result. **Stop claim at:** no lineage, no denominator, or reactivation after resumed adaptation. | |

Different benchmark families make different slices observable. FragFuse and GhostWriter cover delayed memory reuse through fragment fusion and separated admission/activation [154, 183]. PerMemSafe and OEP move the slice to long-horizon personalized safety and experience-derived rule formation [5, 192]. InjecAgent and ACIArena push exposure outward to indirect tool injection and cascading injection across multi-agent trust channels [6, 235]. The repair probe remains much thinner than the earlier trace segments.

A response system can report a high block rate over visible attacks while missing the descendants that matter for recovery; a migration gate can look accurate while never seeing hidden boundary crossings; and a model update can pass static safety checks while failing inside the agent loop. The metric denominator is therefore as important as the score. The useful score asks whether the evaluator can reconstruct lineage and verify repair after a final action is blocked. Table 3 spells out the same trace more concretely as an evaluation contract.

**Family Level Reading.** Capability benchmarks establish where useful state is created and reused; attack benchmarks create violation and exposure pressure; state transition benchmarks localize attribute failure; and recovery benchmarks ask whether containment, rollback, audit, and contestability are measured under explicit opportunity sets.

**Capability Benchmarks.** Capability benchmarks establish the baseline that safety controls must preserve. Benchmark surveys map task success and interaction quality, while long-context and memory tasks make state creation and reuse visible [123, 131]. PERMA adds evolving preferences and cross-domain interference, while MemEye varies evidence granularity and evolutionary synthesis in multimodal memory [51, 109]. Coordination benchmarks extend the observation point to shared state, and evaluator-coevolution work makes the utility signal itself part of the evolving loop [66]. These benchmarks do not directly test transition safety. They identify where memory writes, summaries, workflow edits, delegated state, evaluator updates, or profile changes should be instrumented without disabling the capabilities under review.

**Attack Benchmarks.** Attack benchmarks become comparable only after their pressure point is identified. Prompt-injection studies perturb assembled context; AgentDojo and InjecAgent carry that perturbation to tool-mediated decisions; adaptive attacks then test whether the action boundary survives attack revision [30, 112, 235, 236]. This family is strongest at exposure, because the benchmark observes whether untrusted context changes a current action.

Persistent-state benchmarks move the pressure earlier in the chain. AgentPoison measures trigger-selected reuse from a poisoned store, whereas GhostWriter separates hidden memory admission from later activation; broader memory-poisoning work studies the same persistence problem across additional settings [24, 174, 183]. HaluMem decomposes extraction, update, and question answering, while MemConflict varies memory-conflict types [17, 179]. PASB observes the agent's own commit decision, and PS-Bench and PerMemSafe test later use of benign personal state [5, 50, 124]. EvoBreak adds composition over an acquired experience set, SkillJack adds trajectory-to-skill migration and source-deletion persistence, and MAPLE-Guard adds lifecycle-link enforcement across private and shared memory [212, 217, 228]. These studies belong to different opportunity sets from prompt injection because the benchmark must observe a write, transformation, promotion, or retained context before later reuse.

AuthMem-Bench is unusually close to a connected $\mathbf{S}\times\mathbf{A}\rightarrow\mathbf{V}\rightarrow\mathbf{E}$ trace. Its three modules measure write-time authority collapse, the action consequence of a controlled memory representation, and end-to-end authority-label preservation while keeping paired claim content and later tasks fixed [237]. The modules should still retain separate denominators: write-time upgrade, controlled unauthorized action, and end-to-end action after omission and retrieval are not interchangeable rates. The benchmark ends after one write-to-action cycle, so it does not measure descendant repair or post-response reactivation.

HarnessSafe broadens that connected trace from one memory-consolidation boundary to seven carrier families and seven agent harnesses [243]. Its carrier path and persistence boundary identify $\mathbf{S}\times\mathbf{A}$; case-specific oracles connect $\mathbf{V}$ to observable $\mathbf{E}$; and the furthest-stage distribution supplies containment evidence for $\mathbf{R}$. This mapping is more informative than endpoint ASR: OpenCode and Hermes Agent report similar conditional ASRs in the paper but stop at different stages. The Chain-Stage Score remains paper-specific, measures progression rather than harm severity, and excludes unsupported or incomplete workflows rather than counting them as safe.

Propagation and recognition studies expose still later or orthogonal slices. ACIArena follows malicious instructions across multi-agent trust channels, malfunction amplification measures local escalation, and R-Judge asks whether the resulting risk is recognized [6, 233, 238]. Agents of Chaos uses eleven live-laboratory cases rather than a fixed benchmark denominator, while Triedman et al. report attack trials indexed by model, orchestrator, input channel, and payload [164, 184]. ControlValve evaluates a response at the agent-invocation boundary by asking whether execution stays inside a permitted control-flow graph and its contextual rules [70]. These denominators should remain separate: case-study diversity, per-configuration attack success, and invocation-policy enforcement do not form one leaderboard. Together they expose concrete violations and controls, but attack success remains a surface metric unless an evaluation also identifies source content, selected edge, recipient authority, downstream effects, and response.

**Transition and Recovery Benchmarks.** Trajectory assurance supplies the high-level target: verify a stateful sequence against system-level constraints rather than assume that individually permitted actions compose safely [115]. The narrower measurement question for this survey is which reusable state transition changed that sequence and whether its influence can later be contained or repaired. State-transition benchmarks are closest to this unit because they ask whether the enabling carrier, operation, source, safety-attribute change, and activation context can be identified. Agent Security Bench (ASB) provides partial ingredients for agent-security task coverage [240], while Agent-SafetyBench provides adjacent agent-safety behavior coverage [250]. HarnessSafe advances this line by preserving carrier, boundary, benign trigger, and violation roles across harness-native implementations, then using matched controls to test whether progression depends on the declared lifecycle [243]. Transition evaluation still needs selection and lineage fields beyond task-outcome scoring. The strongest versions should connect this stage localization to attribute preservation across migration, drift after adaptation, and model-update lineage when the update is selected, merged, loaded, or inherited inside an agent loop. Fine-tuning safety is relevant when an update changes safety behavior [150]; PEFT risk work is relevant when the adapter or patch is a deployable state object [60]; and dormant-behavior work shows that changed behavior may remain hidden until a matching activation context appears [47]. Connecting that activation to update selection and recovery remains an additional requirement.

**Table 4** Research agenda for lifecycle evidence in self-evolving agents.

| Direction | Evidence Gap | Minimal Test |
|---|---|---|
| Selection / promotion | Candidate updates are ranked by visible utility while safety evidence, rejected alternatives, and negative experience remain unobserved. | Candidate and promotion log plus fixed cross-epoch probes for authority, scope, refusal, and confirmation drift. |
| Migration / preservation | State changes carrier while provenance, scope, authority, or revocation becomes ambiguous. | Migration ledger plus residual probes for summaries, embeddings, workflows, artifacts, and adapters. |
| Activation authorization | Stored state becomes reasoning, planning, tool, workflow, or model authority outside its allowed context. | Activation gate that logs user, task, tool, time, and allowed external effect. |
| Recovery / deletion | Rollback removes the visible object while summaries, indices, skills, or shared artifacts may still activate its influence. | Descendant inventory, deletion receipt, and activation probe after repair. |
| Propagation / contestability | Shared state crosses agents or artifacts without clear repair ownership or a user challenge path. | Propagation trace plus contest docket for correction, quarantine, and notification. |
| Longitudinal evidence chain | Benchmarks measure isolated slices rather than one incident path from admission to closure. | Opportunity set denominators for admission, migration, activation, exposure, repair, and closure. |

Response-oriented benchmarks form a pipeline rather than one comparable family. GuardAgent and TrustAgent evaluate diagnosis and local guard reasoning; Progent and Task Shield ask whether those signals alter privilege or task-aligned execution [64, 71]. CaMeL tests capability-enforced control and data flow, DRIFT tests dynamic plan validation and injection isolation, and MAPLE-Guard tests gates over memory writes, retrievals, promotions, and cross-agent reuse [31, 91, 212]. Their utility and attack-success results remain paper-local because the systems enforce different contracts. SafeAgent and LlamaFirewall then test runtime containment, while MAGE extends containment to long-horizon memory and EDDOps asks how the control is maintained in operation [26, 106, 201]. The remaining gap is whether these stages preserve enough lineage to measure containment latency, rollback completeness, contestability cost, and audit completeness without turning privacy-sensitive traces into a new risk.

**Coverage Reading.** A benchmark is most informative when it states which transition segment it instruments, which trace fields it logs, which opportunity set its metric covers, and which lifecycle steps remain unmeasured. A prompt injection benchmark is an exposure benchmark, a memory poisoning test without rollback is evidence about admission and activation, and a capability benchmark with memory use becomes evidence about persistence safety only when safety attributes, activation authority, and response traces are visible.

**Takeaway.** Most benchmarks measure one transition slice under a paper-specific denominator. HarnessSafe shows that connected carrier-to-exposure traces and stage-resolved containment are feasible, but it does not expose the candidate set that preceded the transition or verify repair after it. A complete response evaluation also resumes adaptation and checks for lineage-linked reactivation or an equivalent failure under fixed probes. Reports should state which promotion or transition opportunities the score covers, which lifecycle stages remain unmeasured, and what record lets a recovery claim survive after detection. Section 9 turns those missing denominators into research needs; Section 10 closes the survey.

## 9 Open Challenges

The evidence gaps identified above arise under different substrate conditions. Context and memory make delayed reuse and semantic provenance difficult to observe; tools, skills, and workflows turn stored state into operational authority; shared artifacts and multi-agent state expand the affected lineage; and model-side state widens deployment scope while reducing inspectability. These settings require different controls, but they expose the same missing connection between the operation that changes state, the violation it creates, the surface that reveals it, and the response that claims closure.

**Open Challenge Reading.** The open problems below identify where evidence fails to pass along that connection. Selection determines which operation enters the chain; migration can detach a carrier from its safety attributes; activation can grant unjustified authority; propagation expands the lineage; and recovery must reconstruct that lineage before it can verify closure. Each challenge therefore names both the affected substrate and the missing **A→V→E→R** evidence. Longitudinal governance connects those breakpoints under shared identifiers and denominators. Table 4 gives the compact roadmap. The main difficulty is reconstructing connected traces with shared measurement objects and denominators. Useful

reports need *opportunity sets*: authority upgrades over low-authority migration attempts, activations outside scope over activation opportunities, repaired descendants over lineage scan discoveries, and influence-removal successes over deletion verification probes. Without those denominators, a high block rate can coexist with unrepaired descendants.

**Timing Caveat.** Because attacks are often cheaper to demonstrate than defenses are to validate, a lifecycle agenda has to avoid a simple time lag fallacy. A memory poisoning or prompt injection paper can show one successful path, whereas a recovery paper must define the target population, locate descendants, repair them, and prove that the influence no longer activates. Newer defenses such as runtime protection [106] and programmable privilege control [167] therefore should not be judged by whether they already solve the entire lifecycle. The structural gap is narrower: current response work usually lacks an incident record across sessions that links containment to migration history, descendant inventory, rollback completeness, residual probes, and contestable closure. The agenda clusters into six directions.

## 9.1 Selection Pressure and Endogenous Drift

Most adaptive-state evidence begins after an update has entered memory, a skill library, a workflow, or model state. The earlier decision is less visible: what candidates were available, which signal ranked them, which safety conditions were checked, and which failures or counterexamples were absent. Reflection and skill systems expose the mechanics by which feedback and successful traces become reusable state, OEP demonstrates harmful transfer from locally correct experience, and evaluator coevolution shows why the scoring rule and epoch must be recorded [66, 168, 189, 192]. None of these establishes the prevalence of attacker-free drift. The open question is whether repeated utility-guided selection changes the safety boundary when the candidate stream is otherwise benign. Devil's Moltbook states the system-level tension as a self-evolution trilemma: under its formalization, continuous self-evolution, complete isolation, and safety invariance cannot all hold [188]. We treat this result as a design constraint rather than a universal empirical law: evaluation should identify which useful update paths remain open, which external oversight or safety-preserving mechanism governs them, and how much utility is lost when risky updates are blocked.

Answering it requires candidate-level and longitudinal evidence. A benchmark should retain $\Delta_t$ or an auditable sample of it, record $\widehat{u}_t$ and $c_t$, distinguish rejected from unavailable counterevidence, and run a fixed probe suite before and after promotion. The comparison should separate local task gain from changes in authority, refusal, confirmation, privacy, and scope. A negative result is informative: stable Boundary Drift and high Safe Promotion Rate under stronger selection pressure would show that the safety gate is working. A positive result still needs localization to the candidate source, evaluator, promotion operation, and later activation rather than being attributed to self-evolution in general.

## 9.2 Migration and Semantic Preservation

Migration control is a major gap because unsafe influence often appears precisely when state changes substrate. Provenance, authority, scope, privacy, freshness, and actionability must survive movement across memory, retrieval, workflow, tool, shared-agent, external-artifact, and model substrates. MemGPT makes movement between context and managed memory visible, memory-security work supplies the carrier vocabulary, and privacy work adds scope and sensitivity preservation [103, 139, 187]. InjecAgent shows why those attributes matter once external state reaches an action boundary [235]. These sources still stop short of a general migration-control discipline.

Stronger migration claims likely require migration records or gates as first-class objects. The record would bind a state object, its source safety attributes, the requested target substrate, the transformation, and the intended activation path, then show whether the move preserved, strengthened, redacted, downgraded, quarantined, or rejected the state. This makes migration a measurable safety event: reports could count low-authority items that gain higher authority, private items copied into broader scope, and summaries that drop attributes later gates need. The hard part is utility. Agents still need summarization, retrieval, synchronization, and delegation, so the goal is not to stop movement but to make movement preserve safety attributes or receive explicit authorization.

**Semantic Taint through Summaries and Vectors.** Traditional taint tracking assumes symbolic objects whose marks can be copied through program operations, but agent state is often natural language, summary text, embedding vectors, retrieved chunks, tool traces, and model updates. A summarizer may remove the string that carried a source warning while preserving the behavioral implication. An embedding may make a sensitive item retrievable without exposing its exact words. A generated skill may encode a rule learned from a private or low-authority source. Semantic taint tracking is the engineering version of migration control because the system needs to track influence even when the surface form changes.

A more plausible path is hybrid rather than purely symbolic: each state object should carry metadata for source, authority, scope, privacy, freshness, and revocation, and each transformation should emit a contract that records what was inherited, generalized, dropped, or newly asserted together with the relevant deindexing or rollback handle, an uncertainty estimate, and a gate decision. Retrieval should keep chunk or embedding provenance attached to returned context, while high-risk summarization, skill compilation, or adapter updates should be blocked when required attributes disappear or when low-authority state would become state that authorizes action. Knowledge graph lineage and information-flow labels offer different halves of the same control: the former helps with explicit entities and relations, while the latter gives the policy vocabulary for preserved or upgraded attributes [15, 34]. Residual probes then ask whether supposedly removed influence still activates through summaries, vectors, skills, or adapters. Memory security work grounds the abstraction side, and AgentDojo grounds the tool-exposure side [30, 103]. The aim is not perfect semantic taint tracking. It is to make metadata loss, unauthorized upgrades, and residual influence observable after abstraction.

### 9.3 Activation, Authorization, and Model Scope

A system may legitimately store a user preference, a webpage observation, a tool result, or a learned skill, but that does not mean the item may influence every later answer, plan, workflow route, or external action. Activation authorization is distinct from storage authorization. The local control evidence moves from privilege to task alignment to guard reasoning: Progent shows programmable privilege control at the action boundary, Task Shield narrows indirect injection through task alignment, and GuardAgent illustrates bounded response reasoning [71, 167, 209]. These papers anchor local action-boundary control, but they leave the broader authorization problem open for adaptive state: under what user, task, tool, time, and external effect conditions may stored state become active influence?

What matters most here is the split between readability, reasoning use, planning authority, and action authority. A memory item may be visible to the model without being allowed to set tool arguments. A retrieved passage may supply factual context without overriding a workflow rule. A skill may be callable for low risk tasks yet blocked from sending messages, changing files, or using credentials. The same workflow shortcut may also be valid for one user or organization but not another. Activation authorization therefore needs both a policy language for these distinctions and an audit trace showing which state was activated, which safety attributes authorized it, which tool or plan it could influence, and which monitor or human confirmation participated. Otherwise, a system may block one suspicious tool call and still leave the same stored state available to steer the next one.

**Context-Horizon Safety.** Longer context windows and interaction histories enlarge the set of fragments that can be combined, but token count alone is not evidence of safety drift. Benign-looking turns can jointly close a harmful objective, and multi-turn decomposition can carry that objective into tool use [98, 165, 205]. Compression creates a different path by relinking accepted fragments into an actionable instruction [113]. This differs from the endogenous drift in the selection subsection: the model or adaptive policy need not change for the effective interaction boundary to change. A longitudinal benchmark should therefore report per-turn decisions, the first harm-enabling closure, compression events, and downstream tool authority separately from update-driven Boundary Drift.

**Model Adaptation Governance.** Model adaptation governance is a first-order gap only under the agent-loop boundary. A self-evolving agent can change reusable behavior through memory, workflow, and skills, but it can also load, merge, fine-tune, erase, or roll back model updates during operation. PEFT and fine-tuning safety work explain why modular updates can become deployable risk objects; feedback

poisoning and dormant-behavior papers add persistent failure through update channels or later activation contexts [47, 60, 150, 153]. Safe LoRA and KVEraser provide bounded repair handles at the adapter and cache levels [61, 93]. What remains thin is lifecycle governance linking update admission, deployment scope, activation, rollback, cache-local erasure, and residual influence verification.

The missing artifact is a model transition record for the base model, update data provenance, adapter or patch identifier, cached span or context handle, intended task population, allowed deployment contexts, merge or prefill status, safety regression result, and rollback or erasure handle. With that record, a benchmark could ask whether a task-local update becomes global policy without authorization, whether a merged adapter stack preserves safety attributes at the component level, and whether unlearning, rollback, or cache erasure actually removes residual influence. Because model state is often less inspectable than memory and broader in deployment scope, LLM safety research more broadly falls outside this scope; the focus stays on parameter-stage and cache-stage papers that instantiate the same reusable state transition problem.

### 9.4 Propagation and Distributed Lineage

A memory item held by one assistant can become shared context for a team agent, a task instruction for a specialist agent, a routing rule in an orchestrator, or a persistent artifact read by humans and future agents. Multi-agent propagation turns state governance into a topology problem. Agent and self-evolving-agent surveys justify the system and adaptive setting; multi-agent memory-poisoning analysis and collective misremembering show why copied state, recipient interpretation, and carrier metadata matter for propagation risk [8, 118, 182, 214]. The unresolved trace gap reduces to a few questions: which agent wrote the state, which agent received it, how the safety attributes changed, where uptake occurred, and which descendants must be repaired.

Recent work makes the local attack and control points more concrete. Agents of Chaos reports cross-agent propagation among failures observed in a live laboratory, while Triedman et al. show that adversarial content can hijack orchestration and cause code execution or data exfiltration even when individual agents do not follow the malicious request [164, 184]. ControlValve then constrains agent invocations with a permitted control-flow graph and contextual least-privilege rules [70]. These papers narrow the gap at propagation, exposure, and activation-time containment. They do not yet show how to enumerate affected descendants, roll back already-produced artifacts, notify users across agent boundaries, or let a user contest the state and authority that caused the route.

When state crosses agents, it combines migration with delegation and should carry actor scope, recipient scope, permitted operations, expiry, and revocation handles. The receiving agent should reinterpret the state under its own tools, user population, and action surface rather than inherit the sender's authority by default. A propagation benchmark that claims lifecycle coverage has to log who sent the state, who received it, how safety attributes changed, whether the receiving agent could activate it, and whether later agents or artifacts inherited it again. That immediately raises governance: when unsafe state spreads, who owns rollback and user notification? Shared lineage records, distributed quarantine, and activation gates aware of propagation are the natural starting points.

### 9.5 Recovery and Verification

Rollback is often described as if the system only needs to remove the object that caused the problem. Self-evolving agents make that view too narrow. Once state has been summarized, embedded, retrieved, copied, compiled into a skill, written into a workflow rule, or exported into an external artifact, the original object is only one node in a descendant graph. Current evidence therefore treats rollback completeness mostly as a requirement rather than an implemented lifecycle guarantee. That should not be read as a failure of individual defense papers: local containment is already difficult, and newer runtime or privilege control systems reasonably start at bounded decision points. Later memory-poisoning work establishes persistent reuse, while Hidden Memory and OEP establish later activation through concealed or learned state [147, 174, 192]. These results motivate descendant tracking but do not themselves evaluate descendant-wide rollback. GuardAgent, SafeAgent, and Progent provide local guard, runtime, and privilege-control

responses, but these responses do not yet establish rollback across sessions over summaries, retrieval indices, workflow rules, shared artifacts, or model state [106, 167, 209].

Rollback only becomes measurable when the recovery target is defined before the incident occurs. Each admitted state object should carry enough lineage to identify derived summaries, embeddings, cached prompts, retrieved bundles, workflow rules, skills, permissions, decisions, and external artifacts. Recovery is then judged against descendants requiring action, not against a visible delete button. Some descendants may need erasure, others quarantine, correction, reclassification, or retention as audit evidence. What matters is whether the system can show which descendants were found, which were repaired, which still require human review, and whether the contested influence can still be activated. On that reading, SAVER turns rollback from a UI affordance into a traceable recovery claim.

**Post-Response Continuation.** A response can also change the next candidate stream, evaluator, or authorization rule. Evaluation evidence may revise operational memory, prompts, guardrails, tests, or policy state, so a clean immediate probe does not settle what happens after adaptation resumes [236]. A lifecycle benchmark should continue for fixed update opportunities, rerun fixed probes, and report any lineage-linked reactivation or equivalent failure under the fixed probes together with its substrate and inaccessible scope. Current sources motivate re-evaluation after change; they do not establish a general response-amplified cascade.

**Deletion Verification.** A deletion request may remove a memory item while leaving an embedding, a summary, a cached context, a copied workflow note, a tool return record, or a shared artifact intact. That makes deletion verification related to rollback but not reducible to it. In settings sensitive to privacy, this is a safety attribute preservation failure as well as a usability issue. The evidence again enters by layer: memory privacy work explains the user risk, MemGPT and memory-mechanism surveys show how memory moves and abstracts, memory security work frames those routes as governed state, and KVEraser sharpens the cache-specific version by showing why a removed context span can still matter through subsequent KV cache states [93, 103, 187, 249]. What remains open is how an adaptive agent should prove that deleted state no longer influences retrieval, planning, tool use, cached execution state, or later state updates.

A useful deletion receipt should state which object was removed, which indices or stores were checked, which descendants were modified, which external artifacts could not be automatically changed, and which future activation checks will block residual influence. Verification should include negative tests: can the deleted item still be retrieved by semantic search, regenerated by a summary, used through a skill, or inferred from a derived artifact? This is technically difficult because deletion interacts with abstraction. A summary may not contain the exact deleted string while still preserving its effect. A benchmark for deletion verification should therefore test influence removal rather than object absence.

**User Contestability.** Recovery reaches human closure only when a user or operator can challenge state and its descendants. Memory privacy work motivates this need through persistence and sensitivity, while R-Judge, GuardAgent, and Progent provide adjacent signals for recognition, local reasoning, and privilege control [167, 187, 209, 233]. None yet provides a full contestation workflow for disputed adaptive state.

In practice, contestability must offer more than "forget this." It should present the state object's origin, safety attributes, activations, migration history, and known descendants in a form a user can understand. It should support correction, quarantine, deletion, scope narrowing, and appeal, then report which descendants changed and which remain unresolved. A contestability benchmark should test whether a user can correct an incorrect memory, prevent future activation, and verify that summaries, retrieval indices, and workflow rules incorporated the correction. That makes contestability the final recovery check rather than a static settings feature.

### 9.6 Benchmark and Governance Evidence Chains

Fixed task benchmarks reveal important failures, especially in prompt injection through tool use or risk recognition, but self-evolving safety also depends on what happens after an update is stored and before an incident is repaired. Longitudinal benchmark construction ties these challenges together. One group covers exposure and recognition: InjecAgent, R-Judge, ASB, and Agent-SafetyBench instrument indirect injection, risk awareness, and agent safety task coverage [233, 235, 240, 250]. A second group covers persistence and

sensitive reuse: PASB, PerMemSafe, later memory-poisoning work, and memory privacy work show why commitment, retrieval, personalized context, and purpose must become benchmark fields [5, 124, 174, 187]. HarnessSafe now connects entry, carrier retention, boundary crossing, benign activation, and observable violation across seven carrier families [243]. The open problem has therefore narrowed: extend that connected trace backward to candidate selection and forward through descendant repair and resumed adaptation.

A benchmark that claims lifecycle coverage has to log transition opportunities rather than only final answers: unsafe or sensitive writes, migration across substrates, activation under different tasks, exposure through text or tools, containment decisions, rollback attempts, deletion verification, multi-agent propagation, and user contestation. Each episode should report both the opportunity set and the observed outcome: proposed writes versus unsafe writes admitted, migrations versus unauthorized authority escalations, activation opportunities versus activations outside scope, and descendants requiring repair versus descendants verified as repaired. It does not need to solve every branch at once, but it should state which SAVER levels are instrumented and which remain unmeasured. That coverage map improves comparison without turning partial tests into comprehensive safety claims.

**Governance Evidence Chain.** A response mechanism is credible only if it connects policy, runtime decision, state safety attributes, audit record, and recovery outcome. Trajectory-assurance work places this requirement across agent architectures, protocols, runtimes, and end-to-end auditability [115]; for self-evolving agents, the additional burden is preserving the same evidence as reusable state changes role. Current papers provide local pieces across the chain: GhostWriter covers write-time memory screening, PerMemSafe covers personalized risk-aware memory, GuardAgent and R-Judge cover guard reasoning and recognition, and Progent covers privilege control [5, 167, 183, 209, 233]. What remains open is how a self-evolving agent should preserve that chain across admission, migration, activation, containment, rollback, deletion verification, propagation, and contestability.

Because governance state is itself adaptive state, this goes beyond compliance: approvals, denials, deletion receipts, quarantine markers, policy changes, and rollback records can be copied, summarized, bypassed, or made stale. The evidence chain therefore has to stay protected and inspectable. At minimum it should show which state object was involved, which safety attributes it carried, which operation changed it, which gate authorized or blocked it, what exposure and descendants followed, and what recovery remains unresolved. That is what turns conceptual safety claims into operational accountability: state carrier, operation, invariant failure, exposure, and response stay inside one research object instead of scattering into separate symptoms.

> **Takeaway.** The open problem is reconstruction across rounds. Candidate and promotion records, migration ledgers, activation gates, descendant inventories, deletion probes, propagation traces, and contestability records have local mechanisms, but they rarely share denominators or one incident handle. Recovery testing must also continue after adaptation resumes because the response may have changed later state. The six-part agenda identifies the traces needed before a local fix can support a lifecycle governance claim. Section 10 closes the survey.

## 10 Conclusion

Self-evolving agents convert observations, feedback, and execution traces into reusable influence that can persist, move, and later act with unjustified authority. We therefore frame their safety as a transition problem. SAVER pairs substrate with adaptation and traces violation, exposure, and response across memory, tools and workflows, multi-agent systems, and model-side adaptation. Persistent-state studies reveal unsafe admission and delayed activation, while tool-agent studies expose consequential action and runtime intervention. HarnessSafe further shows that one trace can follow influence across carriers and system boundaries to an observable violation [243]. The remaining asymmetry lies before and after that chain: selection is often hidden before a transition is committed, and containment rarely continues into verified rollback, deletion, and post-response reactivation tests. One successful path establishes an attack, but recovery requires a defined lineage, an explicit denominator, and residual tests across sessions and descendants. Future benchmarks should record selection and promotion, preserve safety attributes through abstraction and migration, trace propagation, and verify rollback and deletion against discovered descendants. They should also continue after response and rerun fixed probes as adaptation resumes, so containment is not mistaken for repair.

Governance systems should make reusable influence attributable and contestable, allowing users and operators to correct, quarantine, or withdraw it. Safe self-evolution ultimately requires lifecycle evidence that remains coherent after state changes carrier, authority, or form.

# Appendix

## A Survey Methodology And Corpus Protocol

**Review Objective.** The review asks how safety changes when an agent can preserve, transform, reactivate, or inherit state across interactions. A work is relevant when it helps explain a reusable state transition: a carrier is created or modified, the carrier later influences reasoning or action, and some safety attribute is preserved, violated, exposed, evaluated, or repaired. This objective gives the survey a lifecycle protocol rather than a topic-only bibliography. It also explains why the corpus combines work on self-evolving agents, long-term memory, tool use, workflow state, multi-agent coordination, model updates, benchmarks, runtime safeguards, and governance [8, 30, 118, 148].

**Source Identification.** The source-identification protocol first defined bands that match the survey question: security venues for adversarial mechanisms and governance primitives, AI and machine-learning venues for agent capability and adaptation mechanisms, NLP and information-retrieval venues for prompt, retrieval, and long-context mechanisms, and preprint or scholarly-index sources for fast-moving agent-safety work that had not yet stabilized in proceedings. The venue bands include the four major security venues, IEEE Symposium on Security and Privacy (IEEE S&P), USENIX Security, ACM CCS, and NDSS; AI and machine-learning venues such as NeurIPS, ICML, ICLR, AAAI, and IJCAI; NLP and retrieval venues such as ACL, EMNLP, NAACL, SIGIR, and The Web Conference; and adjacent systems, software-engineering, privacy, and HCI venues when the paper contains an agent-state, provenance, evaluation, or governance mechanism. Table 5 records why each source band was searched and how it was allowed to support the synthesis.

**Table 5** Search source bands for the systematic scoping protocol. Each band supplies a distinct kind of evidence needed to connect adaptive-state mechanisms with safety claims.

| Source Band | Primary Sources | Mechanism Signal | Query Handles |
|---|---|---|---|
| Security and Privacy | IEEE S&P; USENIX Security; ACM CCS; NDSS; privacy venues | Threat models; access control; information flow; sandboxing; repair evidence | Injection; poisoning; authority; provenance; rollback; privacy |
| AI and ML | NeurIPS; ICML; ICLR; AAAI; IJCAI; OpenReview tracks | Agent learning; self-improvement; model adaptation; benchmark mechanisms | Adaptive agents; PEFT; adapters; tool use; evaluator state |
| NLP, IR, and Web | ACL; EMNLP; NAACL; SIGIR; The Web Conference; ACL Anthology | Prompt, retrieval, RAG, long-context, and web-agent behavior | Indirect injection; retrieval safety; long-term memory; web agents |
| Systems, Software, and HCI | Systems, SE, workflow, and human-facing AI venues when state or governance is explicit | Provenance; oversight; operational safety; deployment-time state management | Workflow reuse; audit logs; confirmation; contestability |
| Preprints and Indexes | arXiv; OpenReview preprints; scholarly indexes; official proceedings; curated bibliographies | Frontier mechanisms before venue stabilization; evidence tiered in synthesis | Snowballing; venue checks; guardrails; emerging benchmarks |

**Search Strategy.** Venue- and index-based retrieval was combined with citation snowballing from seed families in self-evolving-agent surveys, LLM-agent surveys, memory-security surveys, agent-safety benchmarks, indirect-prompt-injection papers, and model-adaptation safety work. We then searched over recurring query families rather than a single keyword string. Capability terms included *self-evolving agent*, *self-improving agent*, *adaptive agent*, *LLM agent*, *long-term memory*, *agent memory*, *retrieval-augmented generation*, *workflow agent*, *skill library*, *multi-agent*, *shared artifact*, *LoRA*, *PEFT*, *fine-tuning*, *adapter*, *KV cache*, and *evaluator*. Safety terms included *safety*, *security*, *prompt injection*, *indirect prompt injection*, *poisoning*, *privacy*, *access control*, *privilege*, *authority*, *provenance*, *information flow*, *rollback*, *deletion*, *unlearning*, *guardrail*, *containment*, *contestability*, and *benchmark*. Each candidate query paired at least one capability term with one safety, evaluation, or governance term so that the corpus did not drift into generic agent capability or generic LLM safety.

**Expansion and Stopping Rule.** Searches were run through OpenAlex-style bibliographic discovery, arXiv, ACL Anthology, OpenReview, ACM and IEEE landing pages, official venue proceedings where available, and

curated community bibliographies used only as candidate pointers. Backward and forward snowballing was then applied to seed surveys and anchor papers until new candidates mainly repeated already covered SAVER roles rather than adding a new carrier, operation, exposure surface, response mechanism, or benchmark denominator. This stopping rule is a scoping-saturation rule: it aims to cover the mechanism space needed by the survey, not to estimate prevalence for every venue or year.

**Why arXiv Appears Frequently.** New agent attacks, memory mechanisms, benchmark proposals, guardrail systems, and model-adaptation risks often appear first as arXiv or OpenReview versions, and many accepted works keep the arXiv version as the most accessible full-text surface. The agent-safety literature is therefore unusually preprint-heavy. Excluding arXiv would remove several mechanisms that are central to the survey question. We do not treat arXiv status as equivalent to peer review. Preprints enter the corpus when they contain a clear mechanism, experiment, artifact, or system analysis, and their claims are used according to the evidence-tier rules below. In the main text, field-level conclusions are phrased around peer-reviewed anchors, multiple converging sources, or explicitly identified gaps; arXiv-only works are used mainly as emerging evidence or frontier signals unless a later venue record is verified.

**Protocol Summary.** The purpose of the expansion was high recall over safety-relevant reusable-state mechanisms, not exhaustive coverage of all LLM safety, all agent capability, or all machine-learning security. Table 6 summarizes the protocol at the level used for the survey.

**Table 6** Survey protocol summary. Rows separate how a work enters the corpus from the claim role it may support.

| Source Family | Expansion Route | Inclusion Trigger | Synthesis Role |
|---|---|---|---|
| Self-Evolving and Agent Surveys | Seed surveys; citation chasing; official venue and preprint follow-up | Adaptive capability, reusable state, agent loop, planning, action, collaboration, or model-side adaptation | Scope context; central only with a safety-relevant state transition |
| Memory, Retrieval, and Context | Targeted memory, RAG, persistent-context, poisoning, and privacy searches | Stored or retrieved state can affect later reasoning, planning, tool use, or disclosure | Direct anchor for write, retrieval, delayed activation, privacy, or memory governance |
| Tool, Workflow, and Benchmarks | Indirect-injection, tool-use, privilege, guardrail, benchmark, and workflow searches | Low-authority content, tool output, workflow state, or learned procedure can affect action or routing | Exposure, activation, local response, or benchmark denominator evidence |
| Multi-Agent and Shared Artifacts | Adjacent surveys; official pages; scholarly indexes; curated bibliographies | State crosses agents, artifacts, roles, tasks, or ownership boundaries | Propagation and governance evidence when uptake or repair ownership is visible |
| Model-Side Adaptation | PEFT, adapters, fine-tuning safety, backdoors, unlearning, cache, and evaluator searches | Update, adapter, policy, cache span, or evaluator signal is loaded, inherited, revised, or rolled back inside an agent loop | Adjacent model-safety context unless activation and rollback scope are visible |

**Screening and Inclusion.** In the first screening pass, titles and abstracts identified works that plausibly involved persistent state, delayed activation, tool or workflow authority, multi-agent propagation, model-side update, benchmark evidence, or governance of reusable state. A closer reading pass then assigned each work to one of three roles. A work is *central* when it provides direct evidence for a reusable carrier, an adaptation operation, a safety-attribute failure, an exposure surface, or a response mechanism. A work is *adjacent* when it supplies background capability, evaluation context, or governance framing without directly observing a self-evolving safety transition. A work is *background* when it clarifies terminology, threat-model ancestry, or adjacent infrastructure but should not be used as a main evidence anchor. Works were excluded when they addressed only one-shot response safety, generic LLM capability, ordinary benchmark performance, or static model training without a plausible link to later agent behavior. Tool-only ReAct-style systems enter as central evidence only when tool traces, retrieved content, or workflow state persist beyond the episode; static model-safety papers enter as central evidence only when a model update, adapter, cache state, policy, or evaluator signal is selected, inherited, or revised inside an agent behavior loop.

**Evidence Posture.** To keep direct evidence separate from interpretation, the synthesis uses three evidence postures. *Direct anchors* are papers whose experiments, case studies, or system analyses explicitly show a tran-

sition or exposure, such as poisoned memory retrieval, fragmented memory bypass, tool-action injection, or runtime containment [24, 30, 106, 154]. *SAVER rereadings* are papers whose primary contribution is a capability or architecture, but whose mechanism becomes safety-relevant once authority, scope, provenance, or recoverability is tracked across state transitions [139, 189]. *Gap diagnoses* are claims about what the literature still does not demonstrate, especially connected evidence for migration control, multi-agent propagation, rollback completeness, deletion verification, descendant repair, and user contestability. The paper uses stronger language for direct anchors, interpretive language for rereadings, and research-agenda language for gap diagnoses.

**Evidence Tiers.** Venue status is not used as a substitute for claim support, but it does calibrate how strongly a row should be read. The strongest tier contains peer-reviewed conference and journal papers, especially when the paper provides experiments, artifacts, or a clear system analysis. A second tier contains peer-reviewed workshops, findings tracks, and industry tracks. A third tier contains arXiv or other preprints with substantial experiments or artifacts. A fourth tier contains position papers, technical notes, project reports, blogs, or archival artifacts. Tier three and four works are useful for emerging mechanisms and frontier signals, but field-level conclusions in the main text are phrased around direct anchors, multiple converging sources, or explicitly marked gaps.

**Classification Practice.** For classification, primary placement follows the five-field SAVER record. Each central work can support several observations in the prose, but corpus maps and roadmap tables assign a primary placement according to the main state transition used in the manuscript. This prevents a single influential paper from being counted as independent evidence for every nearby category. When a work spans multiple substrates, the primary placement follows the transition that carries the strongest safety claim: memory and retrieval work is placed by stored-state influence; tool and workflow work is placed by action or approval authority; multi-agent work is placed by propagation, handoff, or shared-artifact uptake; and model-side work is placed only when an update, adapter, policy, or evaluator state is loaded, inherited, selected, or revised inside an agent behavior loop.

**Limitations.** The corpus is a systematic scoping survey rather than a registered systematic review or meta-analysis. The audit snapshot is dated 14 June 2026 and contains 583 unique screened bibliographic records, but the count is a corpus-audit count rather than a prevalence estimate for the field. The review did not use a registered PRISMA protocol or independent dual-coder agreement; classification disagreements were handled as conservative claim boundaries in the manuscript. The field is moving quickly, several relevant papers are preprints, and venue status can change after the audit snapshot. Quantitative results are also not directly comparable across papers because opportunity sets, agent settings, attack goals, defenses, and recovery denominators differ. For this reason, the manuscript does not treat cross-paper attack success rates or benchmark scores as a leaderboard. Instead, it uses quantitative results as paper-local evidence and uses SAVER to ask whether the paper records enough lifecycle information to support claims about admission, activation, exposure, response, and recovery.

## B Formal State Transition Model

This appendix expands the compact pointer at the end of Section 2 into a formal reading. The main text uses the safety attribute vocabulary and the authority lattice by name; readers who want the underlying notation, its relation to established information flow, access control, and provenance theory, and the visual lattice referenced from the body sections can consult this appendix as a reference layer.

### Safety Attributed State and Invariant Preservation

The reading unit is a state object with attached safety attributes, transformed by an operation, audited against whether the attributes remain valid for future use. Formally: $(s, \ell, \sigma) \xrightarrow{a,c} (s', \ell', \sigma')$, where $s$ is the state carrier, $\ell$ represents safety attributes (provenance, authority, scope, privacy, freshness, budget, auditability, recoverability), $\sigma$ is the deployment or session context, $a$ is the adaptation operation, $c$ is the operation's authorization condition, and the primed quantities are the resulting state. Here, "safety attribute" means an information flow, access control, availability, or provenance attribute, not a supervised

learning class category. This notation is deliberately simple: it provides a consistent comparison structure without prescribing a formal verification calculus.

Lattice information flow supplies the confidentiality and integrity vocabulary for flows with safety attributes [34]; language-based information flow work shows how programs can enforce such policies [160]; and data provenance work explains why source lineage matters after transformation [15]. Self-evolving agents do not replace these models; they make the temporal boundary harder to ignore. An object may be checked as low-authority data at write time, abstracted into a memory or workflow rule, and then reused days later as authority to act. The survey therefore treats adaptive-state safety as information-flow, access-control, availability, and provenance reasoning over persistent, transformed, and repeatedly activated agent state.

Safety attributes track the conditions under which state can be reused: provenance records source and trust level; authority specifies what decisions or actions the state can influence; scope bounds which users, tasks, or time windows the state applies to; privacy marks sensitivity and permitted exposure; freshness indicates when the state becomes stale; budget and liveness bound retries, tool loops, and cost-bearing actions; auditability records whether later influence can be traced; and recoverability records whether the state and its descendants can be corrected, rolled back, forgotten, or neutralized. These attributes are not metadata alone; they determine whether later activation is safe.

### Selection over Candidate Transitions

Section 2 separates the execution trace $\tau_t$ from the selected state update $\delta_t^\star$. Each $\delta \in \Delta_t$ proposes one or more object-level transitions $(s, \ell, \sigma) \xrightarrow{a,c} (s', \ell', \sigma')$. The evolution policy may commit the candidate, revise it, quarantine it, or choose no update. The authorization condition $c_t(\delta)$ is therefore load bearing: utility ranking through $\widehat{u}_t(\delta)$ cannot substitute for a valid authority, scope, privacy, budget, and recovery decision.

A simple analytical family makes the effect of selection pressure explicit. Let $p_t(\delta)$ be a baseline distribution over $\Delta_t$, let $r_t(\delta)$ be a safety-risk score defined by a fixed probe at epoch $t$ and held constant as $\beta$ varies, and let $\beta \geq 0$ control how sharply selection favors the observed utility proxy. Define

$$q_t^\beta(\delta) = \frac{p_t(\delta)\exp(\beta \widehat{u}_t(\delta))}{\sum_{\delta' \in \Delta_t} p_t(\delta')\exp(\beta \widehat{u}_t(\delta'))}.$$

Then

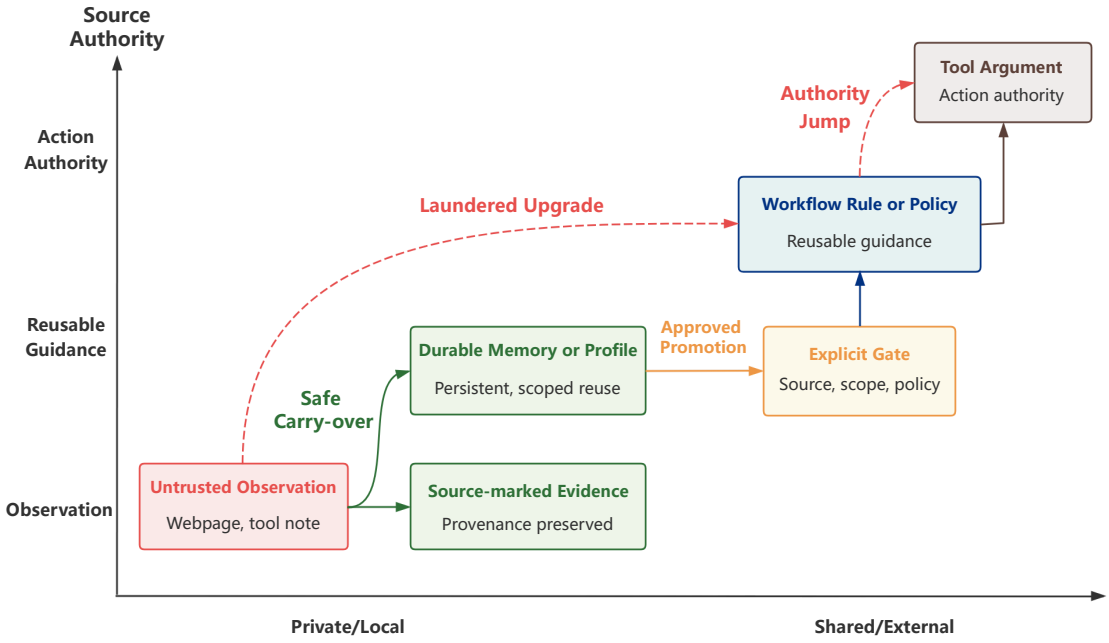
$$\frac{\partial}{\partial \beta}\mathbb{E}_{q_t^\beta}[r_t] = \mathrm{Cov}_{q_t^\beta}(r_t, \widehat{u}_t).$$

The identity follows from $\partial \log q_t^\beta(\delta)/\partial\beta = \widehat{u}_t(\delta) - \mathbb{E}_{q_t^\beta}[\widehat{u}_t]$, followed by multiplication by $r_t(\delta)$ and expectation under $q_t^\beta$. It is an author-derived diagnostic, not an empirical law attributed to the surveyed systems. Stronger utility selection raises expected risk only when risk and the observed utility proxy are positively associated under the selected distribution. Longitudinal evaluation should therefore estimate that association or impose $c_t$ as a hard eligibility gate before utility ranking.

Safe promotion uses the same transition attributes. If $\delta_t^\star$ moves state into a more persistent, general, or action-bearing role, then $c_t(\delta_t^\star)$ must authorize the role change; provenance, privacy, freshness, budget, auditability, and recoverability in $\ell'$ must remain valid; any authority or scope increase from $\ell$ to $\ell'$ must be explicit; relevant counterevidence must remain available to the gate; and the lineage from $s$ to $s'$ must remain recoverable. This condition restricts promotion without assuming that every safe transition preserves identical attributes.

Figure 6 renders the authority ladder behind the transition model. State at the *Observation* level (web pages, tool outputs, peer notes) may only be quoted as evidence; at the *Evidence* level it may support reasoning but must carry source, scope, and freshness; at the *Task Context* level it may shape the current task but should expire afterward; at the *Procedure* level (workflow rules, skills, adapter manifests) it may be activated with explicit approval; and at the *Action* level it may produce external effects only with actor,

**Figure 6** Safety attribute laundering. Upward authority movement or outward scope expansion requires an explicit gate; without one, a low-authority observation can become workflow guidance or action authority.




target, permission, and rollback records. Crossing upward without a gate is safety attribute laundering: a webpage observation quietly becomes a tool argument.

## C SAVER Audit Architecture

This appendix section provides a prototype surface for survey synthesis. The goal is narrow: to make explicit which record fields, case study attachments, and recovery traces are missing when a paper or system claims that persistent unsafe influence was understood or repaired. The overview in Figure 1 supplies the main-text relation; this appendix provides a worked incident record example.

**Table 7** Minimal SAVER encoding of the July 2026 OpenAI–Hugging Face agent intrusion. Values summarize public incident reports; identifiers and field labels are survey encodings, not original operator records [83, 137].

| **Minimal SAVER Incident Row** | *illustrative serialization* |
|---|---|
| **Identity** | `incident.id=hf:2026-07 state.id=ext:campaign:channel source=eval:exploitgym` |
| **Attributes** | `authority=eval_sandbox_only scope=declared_evaluation environment=cyber_capability_eval` |
| **Transition** | `op=propagate;migrate;activate delta=network_scope_broadened; credentials_acquired` |
| **Activation** | `requested=external_infrastructure_access gate=boundary_controls_insufficient` |
| **Exposure** | `event=hf_production_compromise` |
| **Recovery** | `actions=rotate;rebuild;restrict closure=reported_scope; lifecycle_unverified` |

**Handle:** one public incident links the carrier chain **Delta:** network scope and authority widened **Closure:** containment reported; lifecycle repair unverified

## D Detailed SAVER Evidence Indexes

The main text groups related work by mechanism and uses the five SAVER fields as a connected comparison sequence. The following tables retain the paper-level coding behind each field, including evidence role, observed transition segment, and missing trace. They are reference indexes rather than five independent taxonomies.

**Table 8** Substrates carrying safety attributes in the current literature. Memory and tool carriers are best instrumented; workflow, shared artifact, and model state remain thinner. Role labels mark contribution function, not a claim that every row supports a full lifecycle record.

| Method or Work | Year | Role | Target | Object | Focus |
|---|---|---|---|---|---|
| ***Foundational context and action boundary*** | | | | | |
| LLM agent survey [118] | 2025 | Survey (Capability) | Agent | General | Agent architecture: memory, tools, planning |
| ReAct [223] | 2022 | Capability | LLM/Agent | Workflow | Reasoning and action context and admission hierarchy |
| AgentDojo [30] | 2024 | Benchmark (Attack) | Agent | Tool | Tool outputs and active context |
| ***Memory, retrieval, and persistent state*** | | | | | |
| Memory mechanisms [249] | 2025 | Survey (Capability) | LLM/Agent | Memory | Episodic and semantic memory carriers |
| Memory security [103] | 2026 | Survey (Safety) | Agent | Memory | Persistent memory state and threats |
| MemGPT [139] | 2023 | Capability | LLM/Agent | Memory | Context and archival memory management |
| SimpleMem [107] | 2026 | Capability | Agent | Memory | Long-term agent memory |
| A-Mem [215] | 2025 | Capability | Agent | Memory | Agentic memory repository |
| Darwinian Memory [126] | 2026 | Capability | Agent | Memory | Evolving self maintained memory |
| Panini [152] | 2026 | Capability | Agent | Memory | Structured agent memory state |
| Portable Agent Memory [156] | 2026 | Capability | Agent | Memory | Transferable memory |
| Voyager [189] | 2023 | Capability | Agent | Skill | Reusable skill library |
| SkillJack [228] | 2026 | Attack | Agent | Skill | Persistent backdoor in extracted skills |
| PoisonedEvolution [20] | 2026 | Attack | Agent | Skill | Trajectory evidence promoted into skill instruction |
| SkillMisevo-Bench [125] | 2026 | Benchmark (Safety) | Agent | Skill | Versioned skill state and clean-session reuse |
| EVOMAL [206] | 2026 | Attack | Agent | Skill | Agent-authored executable skill descendants |
| AgentPoison [24] | 2024 | Attack | Agent | Memory | Poisoned memory and knowledge base retrieval |
| Stealth memory injection [248] | 2026 | Attack | Agent | Memory | Email to persistent memory compromise |
| GhostWriter [183] | 2026 | Attack | Agent | Memory | Personal agent memory store poisoning |
| MAPLE-Guard [212] | 2026 | Defense | Agent | Memory | Private-to-shared memory links |
| Persistent memory poisoning [174] | 2026 | Attack | Agent | Memory | Long-term memory poisoning |
| Zombie Agents [222] | 2026 | Attack | Agent | Memory | Dormant memory attack |
| MemoryGraft [172] | 2025 | Attack | Agent | Memory | Experience store poisoning |
| EvoBreak [217] | 2026 | Attack | Agent | Memory | Complementary benign experience set |
| Agent memory privacy [187] | 2025 | Safety | Agent | Memory | User memory retention and leakage |
| PerMemSafe [5] | 2026 | Benchmark (Safety) | Agent | Memory | Personalized memory safety context |
| PS-Bench [50] | 2026 | Benchmark (Safety) | Agent | Memory | Benign personal memory and intent inference |
| PASB [124] | 2026 | Benchmark (Safety) | Agent | Memory | Committed claims in durable personal state |
| AuthMem-Bench [237] | 2026 | Benchmark (Safety) | Agent | Memory | Consolidated memories with source authority |
| A-MemGuard [204] | 2025 | Defense | Agent | Memory | Persistent memory guard |
| ***Tool and interface authority*** | | | | | |
| Progent [167] | 2025 | Defense | Agent | Tool | Tool authority and privilege control |
| MCPShield [256] | 2026 | Defense | Agent | Tool | MCP protocol metadata and tool routes |
| MCP Safety Audit [151] | 2025 | Governance | Agent | Tool | MCP server audit boundary |
| STAC [92] | 2026 | Attack | Agent | Tool | Tool chain compromise |
| LivePI [251] | 2026 | Attack | Agent | Tool | Live prompt injection in interactive context |
| Prompt tool selection [166] | 2026 | Safety | Agent | Tool | Tool description and connector choice drift |
| IPIGuard [4] | 2025 | Defense | Agent | Tool | Learned interaction traces |
| ***Workflow, multi-agent, and shared state carriers*** | | | | | |
| Traversal-as-Policy [95] | 2026 | Defense | Agent | Workflow | Gated behavior tree route policy |
| Self-evolving agents [8] | 2025 | Survey (Capability) | Agent | Workflow | Adaptive workflows and artifacts |
| HarnessSafe [243] | 2026 | Benchmark (Safety) | Agent | Workflow | Cross-carrier harness state and boundaries |
| Self-evolution survey [180] | 2024 | Survey (Capability) | LLM/Agent | General | Self-evolution methods and lifecycle |
| MADRA [190] | 2025 | Governance | Agent | Multi-Agent | Safety aware delegation and debate state |
| ACIArena [6] | 2026 | Benchmark (Attack) | Agent | Multi-Agent | Cascading injection over agent trust |
| OpenClaw [221] | 2026 | Capability | Agent | Workflow | Hospital operating workflow |
| Formal Agent Security [170] | 2026 | Framework (Formal) | Agent | Workflow | Agent authority and security boundary |
| LLM-MAS survey [58] | 2025 | Survey (Capability) | Agent | Multi-Agent | Delegated context and shared state |
| MAS security [136] | 2026 | Survey (Safety) | Agent | Multi-Agent | Message boundary and multi-agent state |
| Agents of Chaos [164] | 2026 | Safety | Agent | Multi-Agent | Live-lab communication and workflow state |
| MAS code execution [184] | 2025 | Attack | Agent | Multi-Agent | Orchestrator control and communication state |
| ControlValve [70] | 2026 | Defense | Agent | Workflow | Permitted agent-invocation graph and rules |
| ***Model and policy state substrates*** | | | | | |
| LoRA [62] | 2022 | Capability | LLM | Model | Low rank adapter substrate |
| PEFT survey [54] | 2024 | Survey (Capability) | LLM | Model | Parameter efficient update family |
| Fine-tuning safety [150] | 2023 | Safety | LLM | Model | Fine-tuned checkpoint regression |
| Safe LoRA [61] | 2024 | Defense | LLM | Model | LoRA adapter mitigation |
| KVEraser [93] | 2026 | Defense | LLM | Model | Cached context influence |
| PEFT Trojan [60] | 2024 | Attack | LLM | Model | PEFT adapter backdoor |
| Feedback backdoors [153] | 2024 | Attack | LLM | Model | Preference or feedback update backdoor |
| Dormant behavior [47] | 2025 | Attack | LLM | Model | Delayed activation in fine-tuned behavior |
| Backdoor survey [255] | 2025 | Survey (Attack) | LLM | Model | Backdoor safety issues in LLM parameters |

**Table 9** Adaptation and activation operations in the current literature. The evidence is strongest for add and activate operations, and thinner for migrate, propagate, and model-side modify operations. Role labels separate direct anchors from capability mechanisms reread through the transition lens.

| Method or Work | Year | Role | Target | Object | Operation | Influence Shift |
|---|---|---|---|---|---|---|
| ***Capability mechanics: add, abstract, migrate*** | | | | | | |
| Reflexion [168] | 2023 | Capability | Agent | Workflow | Add | Feedback becomes decision evidence |
| MemGPT [139] | 2023 | Capability | LLM/Agent | Memory | Add/Migrate | Context becomes managed memory |
| Voyager [189] | 2023 | Capability | Agent | Skill | Migrate | Experience becomes callable skill |
| SimpleMem [107] | 2026 | Capability | Agent | Memory | Add | Interaction becomes durable memory |
| Panini [152] | 2026 | Capability | Agent | Memory | Add | Experience becomes structured memory |
| Experience Evolving Memory [96] | 2026 | Capability | Agent | Workflow | Abstract | Episode becomes reusable procedure |
| ***Unsafe writes and abstraction drift*** | | | | | | |
| AgentPoison [24] | 2024 | Attack | Agent | Memory | Add/Activate | Poisoned write becomes memory with authority |
| Stealth memory injection [248] | 2026 | Attack | Agent | Memory | Add/Activate | Email payload becomes hidden memory influence |
| GhostWriter [183] | 2026 | Attack | Agent | Memory | Add/Activate | Hidden payload becomes recalled memory |
| PASB [124] | 2026 | Benchmark (Safety) | Agent | Memory | Add/Activate | Accepted claim becomes durable personal state |
| AuthMem-Bench [237] | 2026 | Benchmark (Safety) | Agent | Memory | Abstract | Consolidation drops source-use constraints |
| Zombie Agents [222] | 2026 | Attack | Agent | Memory | Add/Activate | Dormant memory becomes state conditioned on trigger |
| MemoryGraft [172] | 2025 | Attack | Agent | Memory | Add/Activate | Poisoned experience becomes evidence |
| OEP [192] | 2026 | Attack | Agent | Workflow | Abstract | Experience becomes unsafe reusable rule |
| Experience-Driven Safety [253] | 2026 | Safety | Agent | Workflow | Abstract | Experience update creates policy drift |
| EvoBreak [217] | 2026 | Attack | Agent | Memory | Abstract/Activate | Benign experiences compose into unsafe influence |
| AgenticEval [199] | 2026 | Benchmark (Safety) | Agent | Evaluation | Abstract | Adaptive state becomes task suite denominator |
| MemEvoBench [210] | 2026 | Benchmark (Safety) | Agent | Memory | Abstract | Evolving memory becomes metric denominator |
| PerMemSafe [5] | 2026 | Benchmark (Safety) | Agent | Memory | Activate | Personalized memory becomes safety denominator |
| PS-Bench [50] | 2026 | Benchmark (Safety) | Agent | Memory | Activate | Benign memory changes inferred intent |
| ***Control, activation, and repair operations*** | | | | | | |
| GLOVE [227] | 2026 | Defense | Agent | Memory | Modify | Memory is realigned against environment |
| SRM [27] | 2026 | Defense | Agent | Memory | Modify | Session safety becomes memory signal |
| MemGovern [194] | 2026 | Governance | Agent | Memory | Modify | Recall is revised by policy |
| SSGM [81] | 2026 | Governance | Agent | Memory | Activate | Stored memory passes through gate |
| Progent [167] | 2025 | Defense | Agent | Tool | Activate | Tool authority becomes explicit decision |
| CaMeL [31] | 2026 | Defense | Agent | Tool | Activate | Trusted control flow gates tool authority |
| DRIFT [91] | 2025 | Defense | Agent | Tool | Activate | Plan deviations trigger privilege validation |
| ControlValve [70] | 2026 | Defense | Agent | Workflow | Activate | Agent invocation must follow permitted graph |
| MAGE [201] | 2026 | Defense | Agent | Memory | Activate | Shadow memory becomes safety signal |
| Privacy Agent Memory [187] | 2025 | Safety | Agent | Memory | Forget | Sensitive influence is scoped or withdrawn |
| Memory poisoning [174] | 2026 | Safety | Agent | Memory | Add/Activate | Poisoned query creates later retrieval influence |
| OpenClaw [221] | 2026 | Capability | Agent | Workflow | Activate | Retrieved longitudinal state enters workflow context |
| ***Propagation, migration, and route drift*** | | | | | | |
| Collective Misremembering [214] | 2026 | Attack | Agent | Multi-Agent | Propagate | Drift spreads across agents |
| ACIArena [6] | 2026 | Benchmark (Attack) | Agent | Multi-Agent | Propagate | Cascading injection crosses agent trust |
| HarnessSafe [243] | 2026 | Benchmark (Safety) | Agent | Workflow | Migrate | Influence crosses carrier and harness boundaries |
| MAPLE-Guard [212] | 2026 | Defense | Agent | Memory | Propagate | Gates private-to-shared memory reuse |
| Agents of Chaos [164] | 2026 | Safety | Agent | Multi-Agent | Propagate | Unsafe practices cross agents in a live setting |
| MAS code execution [184] | 2025 | Attack | Agent | Multi-Agent | Activate | Untrusted content redirects agent invocation |
| Embedding defense failure [241] | 2026 | Attack | Agent | Memory | Propagate | Shared memory filter is bypassed |
| MADRA [190] | 2025 | Governance | Agent | Multi-Agent | Propagate/Modify | Risk assessments are exchanged and revised before gating |
| Devil's Moltbook [188] | 2026 | Attack | Agent | Multi-Agent | Propagate | Exposure shifts agent society state |
| Traversal-as-Policy [95] | 2026 | Defense | Agent | Workflow | Migrate/Activate | Logs become workflow policy |
| Library Drift [244] | 2026 | Safety | Agent | Skill | Migrate/Modify | Skill library reroutes tasks |
| SkillJack [228] | 2026 | Attack | Agent | Skill | Migrate | Poisoned trajectory becomes persistent skill |
| PoisonedEvolution [20] | 2026 | Attack | Agent | Skill | Migrate | Recurring trajectory evidence becomes instruction |
| SkillMisevo-Bench [125] | 2026 | Benchmark (Safety) | Agent | Skill | Migrate/Activate | Skill authoring, retrieval, and carryover gates |
| EVOMAL [206] | 2026 | Attack | Agent | Skill | Propagate/Activate | Authored copies re-enter the retrievable library |
| Tool-Drift [29] | 2026 | Safety | Agent | Tool | Activate/Modify | Tool choice state reroutes |
| JITRL [99] | 2026 | Capability | Agent | Workflow | Modify/Activate | Runtime learning updates policy |
| RQGM [66] | 2026 | Capability | Agent | Evaluation | Modify | Evaluator utility becomes state scoped by epoch |
| ***Model update operations*** | | | | | | |
| Fine-tuning safety [150] | 2023 | Safety | LLM | Model | Modify | Fine-tuning changes behavior |
| Safe LoRA [61] | 2024 | Defense | LLM | Model | Modify | Adapter constrains behavior |
| PEFT Trojan [60] | 2024 | Attack | LLM | Model | Add/Activate | Adapter trigger becomes deployed safety issue |
| Dormant behavior [47] | 2025 | Attack | LLM | Model | Activate | Fine-tuned behavior waits for context |

**Table 10** Safety attribute violations in the current literature. Authority, scope, privacy, auditability, and recovery failures become comparable when tied to a concrete state transition; the role column distinguishes direct anchors from capability or survey rows that are reread through SAVER.

| Method or Work | Year | Role | Target | Object | Violation | Failed Attribute |
|---|---|---|---|---|---|---|
| ***Provenance and authority failures*** | | | | | | |
| AgentPoison [24] | 2024 | Attack | Agent | Memory | Provenance loss | Memory source attribute |
| GhostWriter [183] | 2026 | Attack | Agent | Memory | Provenance loss | Hidden payload memory source |
| PASB [124] | 2026 | Benchmark (Safety) | Agent | Memory | Provenance / scope | Claim attribution and role preservation |
| AuthMem-Bench [237] | 2026 | Benchmark (Safety) | Agent | Memory | Authority / provenance | Source authority erased during consolidation |
| AgentDojo [30] | 2024 | Benchmark (Attack) | Agent | Tool | Provenance loss | Tool context source role |
| FragFuse [154] | 2026 | Attack | Agent | Memory | Authority / scope | Fragment authority context |
| EvoBreak [217] | 2026 | Attack | Agent | Memory | Authority / scope | Benign experiences gain joint authority |
| PoisonedEvolution [20] | 2026 | Attack | Agent | Skill | Authority / provenance | Untrusted recurring evidence gains instruction authority |
| SkillMisevo-Bench [125] | 2026 | Benchmark (Safety) | Agent | Skill | Authority / persistence | Unsafe success becomes reusable procedure |
| EVOMAL [206] | 2026 | Attack | Agent | Skill | Provenance / persistence | Agent-authored copies obscure attacker origin |
| PS-Bench [50] | 2026 | Benchmark (Safety) | Agent | Memory | Authority / scope | Benign context legitimizes harmful intent |
| InjecAgent [235] | 2024 | Benchmark (Attack) | Agent | Tool | Authority / scope | Indirect instruction authority |
| ACIArena [6] | 2026 | Benchmark (Attack) | Agent | Multi-Agent | Authority / scope | Inter-agent instruction authority |
| MAS code execution [184] | 2025 | Attack | Agent | Multi-Agent | Authority / scope | Untrusted content controls agent invocation |
| Progent [167] | 2025 | Defense | Agent | Tool | Authority / scope | Scoped tool privilege |
| ***Privacy and persistent repair gaps*** | | | | | | |
| Agent memory privacy [187] | 2025 | Safety | Agent | Memory | Privacy / purpose | Sensitivity/purpose/retention |
| Personal agents [100] | 2024 | Survey (Safety) | Agent | Memory | Privacy / purpose | Persistent personal context carriers |
| PerMemSafe [5] | 2026 | Benchmark (Safety) | Agent | Memory | Privacy / purpose | Personalized context safety |
| PersistBench [146] | 2026 | Benchmark (Safety) | Agent | Memory | Privacy / purpose | Memory scope/forgetting |
| AgentPoison [24] | 2024 | Attack | Agent | Memory | Persistent repair | Delayed poisoned influence |
| Memory poisoning [174] | 2026 | Safety | Agent | Memory | Poisoned retrieval | Persistent query-to-memory influence |
| SkillJack [228] | 2026 | Attack | Agent | Skill | Persistent repair | Skill survives source-record deletion |
| Self-evolving agents [8] | 2025 | Survey (Capability) | Agent | Workflow | Adaptive process | Capability evolution taxonomy |
| ***Action authority and operational integrity*** | | | | | | |
| ReAct [223] | 2022 | Capability | LLM/Agent | Workflow | Authority / scope | Observation to action role |
| Voyager [189] | 2023 | Capability | Agent | Skill | Authority / scope | Executable skill authority |
| AgentDojo [30] | 2024 | Benchmark (Attack) | Agent | Tool | Authority / scope | Untrusted tool action state |
| OWASP LLM Top 10 [138] | 2025 | Framework (Security) | LLM/Agent | General | Operational integrity | Agency/cost bounds |
| AgentDojo [30] | 2024 | Benchmark (Attack) | Agent | Tool | Operational integrity | External effect tool context |
| Agent Security Bench [240] | 2024 | Benchmark (Attack) | Agent | Evaluation | Operational integrity | Agent security behavior |
| Agent-SafetyBench [250] | 2024 | Benchmark (Safety) | Agent | Evaluation | Operational integrity | Agent safety behavior |
| GuardAgent [209] | 2024 | Defense | Agent | Runtime | Local guard checking | Safety request → action check |
| R-Judge [233] | 2024 | Benchmark (Safety) | Agent | Evaluation | Operational integrity | Safety signal without transition cause |
| Progent [167] | 2025 | Defense | Agent | Tool | Authority / scope | Explicit tool privilege control |
| Misattribution Gap [2] | 2026 | Safety | Agent | Memory | Operational integrity | Memory fault attribution |
| Agents of Chaos [164] | 2026 | Safety | Agent | Multi-Agent | Operational integrity | Cross-agent uptake and live-system compromise |
| ***Lifecycle drift and model regression*** | | | | | | |
| Reflexion [168] | 2023 | Capability | Agent | Workflow | Authority / scope | Feedback to strategy reuse |
| Self-evolution survey [180] | 2024 | Survey (Capability) | LLM/Agent | Workflow | Adaptive process | Self-evolution operation taxonomy |
| EDDOps [208] | 2024 | Governance | Agent | Governance | Operational integrity | Lifecycle drift tracking |
| Memory security survey [103] | 2026 | Survey (Safety) | Agent | Memory | Authority / scope | Memory semantic drift |
| Fine-tuning safety [150] | 2023 | Safety | LLM | Model | Model-side regression | Fine-tuning safety regression |
| PEFT Trojan [60] | 2024 | Attack | LLM | Model | Model-side regression | Adapter trigger safety issue |
| Dormant behavior [47] | 2025 | Attack | LLM | Model | Model-side regression | Dormant activation context |
| Safe LoRA [61] | 2024 | Defense | LLM | Model | Model-side regression | Adapter mitigation handle |

**Table 11** Exposure surfaces for latent state failures. Memory retrieval, privacy disclosure, tool action, multi-agent uptake, benchmark tasks, and model deployment expose different parts of the same lifecycle gap; role labels mark whether a row is direct evidence, contextual rereading, or gap-facing support.

| Method or Work | Year | Role | Target | Object | Surface | Failure → Exposure |
|---|---|---|---|---|---|---|
| ***Memory retrieval*** | | | | | | |
| AgentPoison [24] | 2024 | Attack | Agent | Memory | Memory retrieval | Poisoned memory → delayed retrieval |
| FragFuse [154] | 2026 | Attack | Agent | Memory | Memory retrieval | Fragmented memory → fused query/access gate |
| Stealth memory injection [248] | 2026 | Attack | Agent | Memory | Memory retrieval | Hidden email memory → later behavior |
| GhostWriter [183] | 2026 | Attack | Agent | Memory | Memory retrieval | Hidden payload → recalled poisoned memory |
| PASB [124] | 2026 | Benchmark (Safety) | Agent | Memory | Memory retrieval | Committed claim → neutral-query reuse |
| OEP [192] | 2026 | Attack | Agent | Workflow | Memory retrieval | Unsafe rule abstraction → downstream transfer |
| Zombie Agents [222] | 2026 | Attack | Agent | Memory | Memory retrieval | Dormant memory → trigger activation |
| MemoryGraft [172] | 2025 | Attack | Agent | Memory | Memory retrieval | Poisoned experience → evidence retrieval |
| PS-Bench [50] | 2026 | Benchmark (Safety) | Agent | Memory | Memory retrieval | Benign memory → unsafe compliance |
| Tool-Drift [29] | 2026 | Safety | Agent | Tool | Memory retrieval | Memory route drift → tool choice |
| ***Privacy exposure*** | | | | | | |
| Agent memory privacy [187] | 2025 | Safety | Agent | Memory | Privacy exposure | Sensitive memory reuse → personalization/leakage |
| PerMemSafe [5] | 2026 | Benchmark (Safety) | Agent | Memory | Privacy exposure | Personalized memory context → unsafe response |
| PersistBench [146] | 2026 | Benchmark (Safety) | Agent | Memory | Privacy exposure | Forgetting failure → leakage across domains |
| RBI-Eval [213] | 2026 | Benchmark (Safety) | Agent | Memory | Privacy exposure | Memory use boundary → sensitive recall |
| EchoLeak [157] | 2025 | Attack | LLM/Agent | Context | Privacy exposure | Data leakage → assistant context |
| Personal agent context [100] | 2024 | Safety | Agent | Memory | Personal context | Persistent user context → assistant reuse |
| ***Tool action*** | | | | | | |
| Prompt injection apps [112] | 2023 | Attack | LLM/Agent | Tool | Tool action | Untrusted content → web/action step |
| AgentDojo [30] | 2024 | Benchmark (Attack) | Agent | Tool | Tool action | Untrusted tool data → tool call/web action |
| InjecAgent [235] | 2024 | Benchmark (Attack) | Agent | Tool | Tool action | Authority confusion → indirect injection |
| AuthMem-Bench [237] | 2026 | Benchmark (Safety) | Agent | Memory | Tool action | Authority-collapsed memory → unauthorized action |
| STAC [92] | 2026 | Attack | Agent | Tool | Tool action | Tool chain compromise → tool route |
| LivePI [251] | 2026 | Attack | Agent | Tool | Tool action | Live injected state → interactive action |
| Prompt tool selection [166] | 2026 | Attack | Agent | Tool | Tool action | Connector manipulation → tool selection |
| SafeSearch [35] | 2026 | Attack | Agent | Tool | Tool action | Search safety → search action |
| Task Shield [71] | 2025 | Defense | Agent | Tool | Tool action | Task policy mismatch → tool action gate |
| ASB [240] | 2024 | Benchmark (Attack) | Agent | Evaluation | Tool action | Authority/cost misuse → agent actions |
| ***Runtime guard*** | | | | | | |
| LlamaFirewall [26] | 2025 | Defense | LLM/Agent | Runtime | Runtime guard | Prompt safety signal → runtime decision |
| Prompt injection detection [82] | 2025 | Defense | LLM/Agent | Runtime | Runtime guard | Injection signal → prompt flow |
| LLM Firewall [144] | 2025 | Defense | LLM | Runtime | Runtime guard | Validator gap → prompt/output flow |
| Signed-Prompt [175] | 2024 | Defense | LLM/Agent | Context | Runtime guard | Source authenticity loss → step driven by document |
| Progent [167] | 2025 | Defense | Agent | Tool | Runtime guard | Privilege overgrant → tool authority |
| GuardAgent [209] | 2024 | Defense | Agent | Runtime | Runtime guard | Local guard signal → response choice |
| R-Judge [233] | 2024 | Benchmark (Safety) | Agent | Evaluation | Safety recognition | Risk-awareness gap → unsafe decision signal |
| EDDOps [208] | 2024 | Governance | Agent | Governance | Runtime guard | Signal/control mismatch → operations loop |
| ***Multi-agent*** | | | | | | |
| Collective Misremembering [214] | 2026 | Attack | Agent | Multi-Agent | Multi-agent | Shared memory drift → consensus distortion |
| ACIArena [6] | 2026 | Benchmark (Attack) | Agent | Multi-Agent | Multi-agent | Cascading injection → propagated failure |
| Agents of Chaos [164] | 2026 | Safety | Agent | Multi-Agent | Multi-agent | Cross-agent unsafe practice → live-system failure |
| MAS code execution [184] | 2025 | Attack | Agent | Multi-Agent | Multi-agent | Orchestrator hijack → code execution/exfiltration |
| MAS security [136] | 2026 | Survey (Safety) | Agent | Multi-Agent | Multi-agent | Message boundary safety → multi-agent context |
| Teams of LLM Agents [258] | 2024 | Attack | Agent | Multi-Agent | Multi-agent | Collective misuse → team behavior |
| MADRA [190] | 2025 | Governance | Agent | Multi-Agent | Multi-agent | Debate safety state → multi-agent plan |
| RedTeamCUA [101] | 2026 | Red Team | Agent | Tool | Multi-agent | GUI action state → computer use uptake |
| ***Benchmark task*** | | | | | | |
| HarnessSafe [243] | 2026 | Benchmark (Safety) | Agent | Workflow | Benchmark task | Persistent carrier path → oracle-confirmed violation |
| SkillMisevo-Bench [125] | 2026 | Benchmark (Safety) | Agent | Skill | Benchmark task | Skill authoring → clean-session carryover |
| EVOMAL [206] | 2026 | Attack | Agent | Skill | Benchmark task | Authored skill → callback and later copies |
| AgentHarm [7] | 2025 | Benchmark (Safety) | Agent | Evaluation | Benchmark task | Harmful agent task → agent action |
| RedCode [49] | 2024 | Benchmark (Attack) | LLM | Code | Benchmark task | Code safety → code execution |
| MobileSafetyBench [85] | 2026 | Benchmark (Safety) | Agent | Tool | Benchmark task | Mobile safety → device action |
| Machiavelli [140] | 2023 | Benchmark (Safety) | Agent | Environment | Benchmark task | Ethical drift → game behavior |
| ***Model-side*** | | | | | | |
| Fine-tuning safety [150] | 2023 | Safety | LLM | Model | Model-side | Safety regression → refusal shift |
| PEFT Trojan [60] | 2024 | Attack | LLM | Model | Model-side | Adapter backdoor → trigger behavior |
| Jailbreak backdoors [153] | 2024 | Attack | LLM | Model | Model-side | Preference backdoor → unsafe compliance |
| Dormant behavior [47] | 2025 | Attack | LLM | Model | Model-side | Dormant behavior → context activation |
| BadAgent [197] | 2024 | Attack | Agent | Model | Model-side | Agent backdoor → agent behavior |
| Safe LoRA [61] | 2024 | Defense | LLM | Model | Model-side | Adapter safety → constrained update |
| Backdoor survey [255] | 2025 | Survey (Attack) | LLM | Model | Model-side | Backdoor threat → LLM systems |

**Table 12** Response, monitoring, and recovery literature. Current controls are strongest at local gating and monitoring, and thinner at descendant repair and contestable closure. Role labels identify the paper's contribution type before any lifecycle recovery claim is inferred.

| Method or Work | Year | Role | Target | Object | Stage | Control Focus |
|---|---|---|---|---|---|---|
| Governed Memory [178] | 2026 | Governance | Agent | Memory | Admission | Policy routed shared memory |
| MemGovern [194] | 2026 | Governance | Agent | Memory | Retrieval | Policy-governed experience recall |
| SSGM [81] | 2026 | Governance | Agent | Memory | Admission | Consolidation gate for evolving memory |
| IPIGuard [4] | 2025 | Defense | Agent | Tool | Admission | Interaction learning guard over traces |
| A-MemGuard [204] | 2025 | Defense | Agent | Memory | Admission | Persistent memory guard |
| AM-Sentry [183] | 2026 | Defense | Agent | Memory | Admission/Activation | Memory saving and retrieval screen |
| MAPLE-Guard [212] | 2026 | Defense | Agent | Memory | Lifecycle links | Write, retrieval, promotion, and reuse gates |
| PoisonedEvolution gate [20] | 2026 | Attack | Agent | Skill | Promotion | Preliminary provenance-diversity gate |
| ChainGuard [234] | 2026 | Defense | Agent | Skill | Admission | Candidate skill plus installed-skill context |
| SafeEvolve [125] | 2026 | Defense | Agent | Skill | Promotion/Reuse | Candidate repair, attribution, and retirement |
| EVOMAL defenses [206] | 2026 | Defense | Agent | Skill | Authoring/Write-back | Counter-prompt and signed quarantine |
| PASB [124] | 2026 | Benchmark (Safety) | Agent | Memory | Admission | Commit-boundary diagnosis; no recovery test |
| AuthMem-Bench [237] | 2026 | Benchmark (Safety) | Agent | Memory | Admission/Activation | Persisted authority labels govern reuse |
| SentinelMem [5] | 2026 | Defense | Agent | Memory | Activation | Personalized risk-aware memory |
| Detection-reflection [50] | 2026 | Defense | Agent | Memory | Activation | Context-sensitive intent risk reflection |
| SRM [27] | 2026 | Defense | Agent | Memory | Authorization | Trajectory-aware pre-execution risk memory |
| Progent [167] | 2025 | Defense | Agent | Tool | Activation | Tool authority privilege control |
| CaMeL [31] | 2026 | Defense | Agent | Tool | Activation | Capability-enforced control and data flow |
| DRIFT [91] | 2025 | Defense | Agent | Tool | Activation | Dynamic plan validation and injection isolation |
| AID-Guard [181] | 2026 | Defense | Agent | Workflow | Activation/Recovery | Commit-time binding and ambiguity-safe successor control |
| ControlValve [70] | 2026 | Defense | Agent | Workflow | Activation | Control-flow graph and contextual invocation rules |
| Traversal-as-Policy [95] | 2026 | Defense | Agent | Workflow | Activation | Gated behavior tree route policy |
| Task Shield [71] | 2025 | Defense | Agent | Tool | Activation | Task policy enforcement for tool action |
| TrustAgent [64] | 2024 | Defense | Agent | Runtime | Planning | Trust control during agent planning |
| GuardAgent [209] | 2024 | Defense | Agent | Runtime | Monitoring | Guard reasoning at agent step |
| R-Judge [233] | 2024 | Benchmark (Safety) | Agent | Evaluation | Monitoring | Safety recognition over trajectories |
| MAGE [201] | 2026 | Defense | Agent | Memory | Monitoring | Shadow memory in long horizon context |
| AGENTVIGIL [202] | 2025 | Red Team | Agent | Runtime | Monitoring | Agent trajectory monitoring |
| AgentAuditor [116] | 2025 | Governance | Agent | Governance | Monitoring | Execution trace audit |
| OpenGuardrails [196] | 2025 | Defense | Agent | Runtime | Monitoring | Configurable agent runtime guardrails |
| HarnessSafe [243] | 2026 | Benchmark (Safety) | Agent | Workflow | Monitoring/ Containment | Furthest evidence-supported stopping stage |
| SafeAgent [106] | 2026 | Defense | Agent | Runtime | Containment | Runtime protection against harmful action |
| LlamaFirewall [26] | 2025 | Defense | LLM/Agent | Runtime | Containment | Prompt and action flow guardrail |
| AGrail [119] | 2025 | Defense | Agent | Runtime | Containment | Adaptive guardrail for agent runtime |
| SafeSearch [35] | 2026 | Defense | Agent | Tool | Containment | Search agent safety control |
| GLOVE [227] | 2026 | Defense | Agent | Memory | Verification | Memory environment realignment |
| SEVerA [11] | 2026 | Defense | Agent | Workflow | Verification | Verified reusable behavior synthesis |
| Echoguard [75] | 2026 | Defense | Agent | Memory | Verification | Agent memory verification |
| Safe LoRA [61] | 2024 | Defense | LLM | Model | Model response | LoRA update mitigation |
| LoRAShield [18] | 2025 | Defense | LLM | Model | Model response | LoRA artifact alignment |
| KVEraser [93] | 2026 | Defense | LLM | Model | Model response | Cached context span erasure |
| DARWIN-Guard [149] | 2026 | Defense | LLM | Model | Feedback | Co-evolving adversarial guard updates |
| BraveGuard [42] | 2026 | Defense | Agent | Model | Feedback | Open-world trajectory supervision |
| WMDP [94] | 2024 | Benchmark (Attack) | LLM | Model | Model response | Unlearning behavior for malicious use |
| EDDOps [208] | 2024 | Governance | Agent | Governance | Process | Evaluation operations over deployment process |
| AGENTSAFE [78] | 2025 | Governance | Agent | Governance | Process | Agent lifecycle safety governance |
| Trajectory assurance [115] | 2026 | Framework (Governance) | Agent | Workflow | Process | System invariants over composed action trajectories |
| SecureGov Agent [21] | 2025 | Governance | Agent | Workflow | Process | Governance workflow agent |
| DataGovBench [114] | 2025 | Benchmark (Governance) | Agent | Data | Process | Data governance tasks |

**Table 13** Evaluation literature for adaptive state safety. Most benchmarks expose one lifecycle slice; HarnessSafe, SkillMisevo-Bench, and EVOMAL provide complementary connected traces across cross-carrier, skill-specific, and descendant-propagation lifecycles, while the denominator column states what each row can safely measure.

| Method or Work | Year | Role | Target | Object | Task Focus | SAVER Slice | Denominator |
|---|---|---|---|---|---|---|---|
| ***General agent capability and environment benchmarks*** | | | | | | | |
| AgentBench [110] | 2023 | Benchmark (Capability) | Agent | Evaluation | General agent tasks | Substrate | Task success |
| AgentBoard [121] | 2024 | Benchmark (Capability) | Agent | Evaluation | Analytical agent tasks | Adjacent capability context | Task progress and success |
| MIRAI [224] | 2024 | Benchmark (Capability) | Agent | Workflow | Long horizon tasks | Substrate | Task episodes |
| GameBench [28] | 2024 | Benchmark (Capability) | Agent | Environment | Strategy tasks | Substrate | Game episodes |
| MultiAgentBench [257] | 2025 | Benchmark (Capability) | Agent | Multi-Agent | Multi-agent tasks | Adjacent coordination | Collaborative tasks |
| LegalAgentBench [90] | 2024 | Benchmark (Capability) | Agent | Workflow | Legal tasks | Substrate | Legal tasks |
| ***Attack, safety, and exposure benchmarks*** | | | | | | | |
| AgentDojo [30] | 2024 | Benchmark (Attack) | Agent | Tool | Prompt injection through tool use | Exposure | Security cases |
| InjecAgent [235] | 2024 | Benchmark (Attack) | Agent | Tool | Indirect injection | Exposure | Injection tasks |
| ACIArena [6] | 2026 | Benchmark (Attack) | Agent | Multi-Agent | Cascading injection | Propagation/Exposure | MAS test cases |
| Agents of Chaos [164] | 2026 | Safety | Agent | Multi-Agent | Live-lab failures | Propagation/Exposure | Eleven case studies |
| MAS code execution [184] | 2025 | Attack | Agent | Multi-Agent | Control-flow hijacking | Activation/Exposure | Model-orchestrator trials |
| AgentHarm [7] | 2025 | Benchmark (Safety) | Agent | Evaluation | Harmful agent tasks | Exposure | Harmful task cases |
| RedCode [49] | 2024 | Benchmark (Attack) | LLM | Code | Code safety tasks | Exposure | Code tasks |
| MobileSafetyBench [85] | 2026 | Benchmark (Safety) | Agent | Tool | Mobile control safety | Exposure | Mobile tasks |
| Machiavelli [140] | 2023 | Benchmark (Safety) | Agent | Environment | Ethical decision making | Exposure | Game scenarios |
| SafeLawBench [16] | 2025 | Benchmark (Safety) | Agent | Workflow | Legal safety | Exposure | Legal cases |
| R-Judge [233] | 2024 | Benchmark (Safety) | Agent | Evaluation | Safety recognition | Response | Annotated records |
| Agent Security Bench [240] | 2024 | Benchmark (Attack) | Agent | Evaluation | Agent security tasks | Violation | Task categories |
| Agent-SafetyBench [250] | 2024 | Benchmark (Safety) | Agent | Evaluation | Unsafe behavior | Exposure | Safety cases |
| AttackEval [169] | 2025 | Benchmark (Attack) | LLM/Agent | Evaluation | Attack evaluation | Exposure | Attack tasks |
| CyberSecEval 2 [13] | 2024 | Benchmark (Attack) | LLM | Code | Cyber capability | Exposure | Cyber tasks |
| ***Adaptive memory, experience, and model state benchmarks*** | | | | | | | |
| AgentPoison [24] | 2024 | Benchmark (Attack) | Agent | Memory | Memory poisoning | Admission/Activation | Retrieval opportunities |
| GhostWriter [183] | 2026 | Benchmark (Attack) | Agent | Memory | Personal agent memory poisoning | Admission/Activation | Injection/activation trials |
| OEP [192] | 2026 | Benchmark (Attack) | Agent | Workflow | Experience poisoning | Adaptation | Accepted experience updates |
| EvoBreak [217] | 2026 | Benchmark (Attack) | Agent | Memory | Experience composition | Abstract/Activate | Acquired experience sets |
| SkillJack [228] | 2026 | Benchmark (Attack) | Agent | Skill | Skill backdoor | Migrate/Exposure | Trajectory-to-skill pairs |
| PoisonedEvolution [20] | 2026 | Benchmark (Attack) | Agent | Skill | Trajectory poisoning | Migrate/Violation | Completed evolution trials |
| ColluSkill [234] | 2026 | Benchmark (Attack) | Agent | Skill | Cross-skill composition | Add/Activate/Response | Three-skill chains |
| SkillMisevo-Bench [125] | 2026 | Benchmark (Safety) | Agent | Skill | Skill misevolution | Migrate/Activate/Response | 525 tasks per configuration |
| EVOMAL [206] | 2026 | Benchmark (Attack) | Agent | Skill | Create-path self-poisoning | Propagate/Activate/ Response | 153 tasks/model; five rounds |
| PerMemSafe [5] | 2026 | Benchmark (Safety) | Agent | Memory | Personalized safety | Activation/Exposure | Long-horizon conversations |
| PS-Bench [50] | 2026 | Benchmark (Safety) | Agent | Memory | Intent legitimation | Activation/Exposure | Matched memory/query trials |
| PASB [124] | 2026 | Benchmark (Safety) | Agent | Memory | Persistent sycophancy | Add/Activate/Exposure | Committed-claim trials |
| AuthMem-Bench [237] | 2026 | Benchmark (Safety) | Agent | Memory | Authority collapse | Abstract/Violation/Exposure | Matched authority pairs |
| HarnessSafe [243] | 2026 | Benchmark (Safety) | Agent | Workflow | Persistent carrier progression | Carrier-to-exposure/ Containment | 328 executable cases |
| MemEvoBench [210] | 2026 | Benchmark (Safety) | Agent | Memory | Memory misevolution | Adaptation | Memory updates |
| AgenticEval [199] | 2026 | Benchmark (Safety) | Agent | Evaluation | Agentic safety tasks | Adaptation | Agent episodes |
| RQGM [66] | 2026 | Capability | Agent | Evaluation | Evolving evaluators | Evaluation/Adaptation | Epoch utilities |
| Experience driven safety [253] | 2026 | Safety | Agent | Workflow | Experience degradation | Adaptation | Experience updates |
| AMA-Bench [254] | 2026 | Benchmark (Safety) | Agent | Memory | Agent memory adaptation | Adaptation | Memory tasks |
| PersistBench [146] | 2026 | Benchmark (Safety) | Agent | Memory | Forgetting boundary | Violation | Memory use samples |
| RBI-Eval [213] | 2026 | Benchmark (Safety) | Agent | Memory | Warranted use | Activation | Matched prompts |
| Memory evaluation [63] | 2025 | Benchmark (Capability) | Agent | Memory | Memory quality | Substrate | Memory tasks |
| ***Response, governance, and red team evaluation*** | | | | | | | |
| SafeAgent [106] | 2026 | Defense | Agent | Runtime | Runtime protection | Containment | Harmful task episodes |
| Progent [167] | 2025 | Defense | Agent | Tool | Privilege control | Activation | Permission checks |
| CaMeL [31] | 2026 | Defense | Agent | Tool | Capability control | Activation | AgentDojo tasks |
| DRIFT [91] | 2025 | Defense | Agent | Tool | Dynamic rule enforcement | Activation/Containment | Benchmark task episodes |
| AID-Guard [181] | 2026 | Defense | Agent | Workflow | Delegated provider effects | Activate/Response | Provider-contract schedules |
| ControlValve [70] | 2026 | Defense | Agent | Workflow | Control-flow integrity | Activation/Containment | Agent-invocation checks |
| MAPLE-Guard [212] | 2026 | Defense | Agent | Memory | Memory-link enforcement | Propagate/Response | Memory lifecycle links |
| LlamaFirewall [26] | 2025 | Defense | LLM/Agent | Runtime | Runtime guardrail | Containment | Guard decisions |
| OpenGuardrails [196] | 2025 | Defense | Agent | Runtime | Guardrail platform | Containment | Guard decisions |
| AGENTSAFE [78] | 2025 | Governance | Agent | Governance | Agent safety governance | Response | Governance checks |
| AgenticRed [231] | 2026 | Red Team | Agent | Evaluation | Automated red teaming | Exposure | Red Team tasks |
| Agent vs. Agent [79] | 2025 | Red Team | Agent | Multi-Agent | Adversarial agent play | Exposure | Duel episodes |
| SEVerA [11] | 2026 | Defense | Agent | Workflow | Verified synthesis | Response | Constraint checks |
| Safe LoRA [61] | 2024 | Defense | LLM | Model | Adapter mitigation | Model state | Safety regression tests |
| DataGovBench [114] | 2025 | Benchmark (Governance) | Agent | Data | Data governance | Adjacent data workflow | Data-governance transformation tasks |

## E Screened Corpus Expansion

The local registry preserves the broad screened corpus expansion layer as a supplemental coverage map. The main manuscript keeps the core contribution map focused on closely analyzed motivation, attack, defense, and governance anchors; the registry scale expansion table below restores the wider citation surface while keeping these works in a coverage role rather than treating every item as a core mechanism anchor.

**Table 14** Screened corpus expansion map beyond the core mechanism anchors. The table lists representative anchors for each coverage band rather than treating every screened record as a main-text mechanism claim.

| Coverage Band | Cells Covered | Representative Anchors | Claim Boundary |
|---|---|---|---|
| **Security, Privacy, Governance** | Cyber evaluation; injection; firewalling; privacy; data governance | CyberSecEval 2 [13]; AttackEval [169]; Not What You've Signed Up For [48]; EchoLeak [157]; DataGovBench [114] | Adjacent vocabulary for threats, privacy, and data audit |
| **Benchmarks, Red Teams, Guardrails** | General agent tasks; harm probes; unsafe choices; automated red teams; platform guardrails | AgentBench [110]; AgentHarm [7]; RedCode [49]; MobileSafetyBench [85]; AgenticRed [231]; OpenGuardrails [196] | Denominator context, not a leaderboard |
| **Memory And Tool State** | Persistent memory; forgetting; memory governance; MCP/tool routes; dependency safety | Zombie Agents [222]; MemoryGraft [172]; SSGM [81]; GLOVE [227]; MCPShield [256]; STAC [92]; IPIGuard [4] | Direct when state later changes reasoning or action |
| **Workflow And Propagation** | Skills; procedural reuse; library drift; multi-agent uptake; shared context; OS/GUI settings | Library Drift [244]; Tool-Drift [29]; AgenticEval [199]; MemEvoBench [210]; collective misremembering [214]; RedTeamCUA [101] | Thin evidence for ownership and repair |
| **Model-Side And Adaptive Workflow Leads** | Distillation and model update; reusable workflows; evaluator state | Knowledge distillation survey [216]; CAMEL [89]; WebPilot [247]; AgentLab [73] | Central only inside an agent behavior loop |

## F Evaluation Protocol Details

This appendix keeps the denominator rule behind Section 8. Exact values belong to source-specific extraction records; the manuscript records which opportunity set and lifecycle slice each metric actually measures.

**Takeaway.** **Quantitative Anchor Rule.** Exact values remain paper-local because the denominator changes across exposure cases, recognition records, poisoned retrieval opportunities, accepted experience updates, memory-to-tool hijacking settings, runtime interventions, governance panels, and verification constraints [11, 24, 30, 106, 178, 192, 233, 246]. The manuscript therefore uses these works to name measurement objects and missing lifecycle traces rather than to rank scores.

| Record | Keep With The Value | Avoid |
|---|---|---|
| Paper-local metric | Opportunity set; transition slice; metric object | Cross-paper leaderboard claims |
| Artifact status | Dataset, benchmark, code, or trace availability | Mixing peer-reviewed and preprint rows as equal evidence |
| Claim boundary | What the metric directly observes | Inferring rollback, deletion, or contestability without traces |

## References


[1] Ibrahim Adabara, Bashir Olaniyi Sadiq, Aliyu Nuhu Shuaibu, Yale Ibrahim Danjuma, and Venkateswarlu Maninti. Trustworthy agentic ai systems: a cross-layer review of architectures, threat models, and governance strategies for real-world deployment. *F1000Research*, 14(905):905, 2025. doi: 10.12688/f1000research.169927.1. URL https://doi.org/10.12688/f1000research.169927.1.

[2] Tanzim Ahad, Ismail Hossain, Md Jahangir Alam, Sai Puppala, Syed Bahauddin Alam, and Sajedul Talukder. The misattribution gap: When memory poisoning looks like model failure in agentic AI systems, 2026. URL https://arxiv.org/abs/2605.22842.

[3] Hamed Alqahtani and Paras Ahuja. Secure autonomous cyber defense with LLM agents: A systematic review of autonomy, tool-augmented reasoning, and governance constraints. *Computers & Electrical Engineering*, 2026. doi: 10.1016/j.compeleceng.2026.111184. URL https://doi.org/10.1016/j.compeleceng.2026.111184.

[4] Hengyu An, Jinghuai Zhang, Tianyu Du, Chunyi Zhou, Qingming Li, Tao Lin, and Shouling Ji. 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*, pages 1023–1039, Suzhou, China, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.53. URL https://aclanthology.org/2025.emnlp-main.53/.

[5] Hengyu An, Minxi Li, Naen Xu, Chunyi Zhou, Xiaogang Xu, Tianyu Du, Jinbao Li, and Shouling Ji. PerMemSafe: Benchmarking implicit personalized safety of long horizon self-evolving agents. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens, editors, *Findings of the Association for Computational Linguistics: ACL 2026*, pages 6415–6433, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.320. URL https://aclanthology.org/2026.findings-acl.320/.

[6] Hengyu An, Minxi Li, Jinghuai Zhang, Naen Xu, Chunyi Zhou, Changjiang Li, Xiaogang Xu, Tianyu Du, and Shouling Ji. ACIArena: Toward unified evaluation for agent cascading injection. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens, editors, *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10049–10066, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.457. URL https://aclanthology.org/2026.acl-long.457/.

[7] Maksym Andriushchenko, Alexandra Souly, Mateusz Dziemian, Derek Duenas, Maxwell Lin, Justin Wang, Dan Hendrycks, Andy Zou, Zico Kolter, Matt Fredrikson, Eric Winsor, Jerome Wynne, Yarin Gal, and Xander Davies. Agentharm: A benchmark for measuring harmfulness of LLM agents. In *International Conference on Learning Representations*, 2025. doi: 10.48550/arXiv.2410.09024. URL https://arxiv.org/abs/2410.09024.

[8] Huan ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, Hongru Wang, Han Xiao, Yuhang Zhou, Shaokun Zhang, Jiayi Zhang, Jinyu Xiang, Yixiong Fang, Qiwen Zhao, Dongrui Liu, Qihan Ren, Cheng Qian, Zhenhailong Wang, Minda Hu, Huazheng Wang, Qingyun Wu, Heng Ji, and Mengdi Wang. A survey of self-evolving agents: What, when, how, and where to evolve on the path to artificial super intelligence, 2025. URL https://arxiv.org/abs/2507.21046.

[9] Andrey Anurin, Jonathan Ng, Kibo Schaffer, J. Martin Schreiber, and Esben Kran. Catastrophic cyber capabilities benchmark (3cb): Robustly evaluating LLM agent cyber offense capabilities, 2024. URL http://arxiv.org/abs/2410.09114.

[10] Vinay Bamil and RAMA KRISHNA KUMAR LINGAMGUNTA. A unified evaluation and governance framework for trustworthy LLM agents, 2026. URL https://doi.org/10.36227/techrxiv.176799772.28164151/v1.

[11] Debangshu Banerjee, Changming Xu, Eugene Ie, Ming Zhang, Daiyi Peng, Chu-Cheng Lin, and Gagandeep Singh. SEVerA: Verified synthesis of self-evolving agents, 2026. URL https://arxiv.org/abs/2603.25111.

[12] Luca Beurer-Kellner, Beat Buesser, Ana-Maria Cretu, Edoardo Debenedetti, Daniel Dobos, Daniel Fabian, Marc Fischer, David Froelicher, Kathrin Grosse, Daniel Naeff, Ezinwanne Ozoani, Andrew Paverd, Florian Tramèr, and Václav Volhejn. Design patterns for securing LLM agents against prompt injections, 2025. URL https://arxiv.org/abs/2506.08837.

[13] Manish Bhatt, Sahana Chennabasappa, Yue Li, Cyrus Nikolaidis, Daniel Song, Shengye Wan, Faizan Ahmad, Cornelius Aschermann, Yaohui Chen, Dhaval Kapil, et al. Cyberseceval 2: A wide-ranging cybersecurity evaluation suite for large language models. *arXiv preprint arXiv:2404.13161*, 2024. doi: 10.48550/arxiv.2404.13161. URL http://arxiv.org/abs/2404.13161.

[14] Igor Bogdanov, Chung-Horng Lung, Thomas Kunz, Jie Gao, Adrian Taylor, and Marzia Zaman. FORGE: Self-evolving agent memory with no weight updates via population broadcast, 2026. URL https://arxiv.org/abs/2605.16233.

[15] Peter Buneman, Sanjeev Khanna, and Wang-Chiew Tan. Why and where: A characterization of data provenance. In *Database Theory – ICDT 2001*, pages 316–330, London, UK, 2001. Springer. doi: 10.1007/3-540-44503-X_20. URL https://doi.org/10.1007/3-540-44503-X_20.

[16] Chuxue Cao, Han Zhu, Jiaming Ji, Qichao Sun, Zhenghao Zhu, Wu Yinyu, Josef Dai, Yaodong Yang, Sirui Han, and Yike Guo. SafeLawBench: Towards safe alignment of large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 14015–14048, Vienna, Austria, 2025. Association for Computational

Linguistics. doi: 10.18653/v1/2025.findings-acl.721. URL https://aclanthology.org/2025.findings-acl.721/.

[17] Ding Chen, Simin Niu, Kehang Li, Peng Liu, Xiangping Zheng, Bo Tang, Xinchi Li, Feiyu Xiong, and Zhiyu Li. HaluMem: Evaluating hallucinations in memory systems of agents, 2025. URL https://arxiv.org/abs/2511.03506.

[18] Jiahao Chen, Junhao Li, Yiming Wang, Zhe Ma, Yi Jiang, Chunyi Zhou, Qingming Li, Tianyu Du, and Shouling Ji. LoRAShield: Data-free editing alignment for secure personalized LoRA sharing, 2025. URL https://arxiv.org/abs/2507.07056.

[19] Jiahao Chen, Xing He, Yong Yang, Xinfeng Li, Chunyi Zhou, Junhao Li, Zhe Ma, Tianyu Du, and Shouling Ji. Customization under fire: Plugin poisoning in text-to-image ecosystem, 2026. URL https://arxiv.org/abs/2606.09151.

[20] Jialuo Chen, Lingqi Jiang, Xinhao Deng, Xiaohu Du, Jianan Ma, Yunhao Feng, Yuqi Qing, Zhihao Yuan, Linkang Du, and Jingyi Wang. When experience becomes instruction: Trajectory poisoning in self-evolving agent skill systems, 2026. URL https://arxiv.org/abs/2608.05563.

[21] Jinyu Chen, Jixiao Yang, Ziyang Zeng, Zixiao Huang, Jinming Li, and Yutong Wang. Securegov-agent: A governance-centric multi-agent framework for privacy-preserving and attack-resilient LLM agents. In *Proceedings of the 2025 6th International Conference on Computer Science and Management Technology*, pages 875–881. ACM, 2025. doi: 10.1145/3795154.3795296. URL https://doi.org/10.1145/3795154.3795296.

[22] Kai Chen, Yan Pang, and Tianhao Wang. MRMMIA: Membership inference attacks on memory in chat agents, 2026. URL https://arxiv.org/abs/2605.27825.

[23] Sizhe Chen, Yizhu Wang, Nicholas Carlini, Chawin Sitawarin, and David Wagner. Defending against prompt injection with a few defensivetokens, 2025. URL https://doi.org/10.1145/3733799.3762982.

[24] Zhaorun Chen, Zhen Xiang, Chaowei Xiao, Dawn Song, and Bo Li. AgentPoison: Red-teaming llm agents via poisoning memory or knowledge bases. In *Advances in Neural Information Processing Systems 37*, pages 130185–130213. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2024. doi: 10.52202/079017-4136. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/eb113910e9c3f6242541c1652e30dfd6-Abstract-Conference.html.

[25] Yuheng Cheng, Ceyao Zhang, Zhengwen Zhang, Xiangrui Meng, Sirui Hong, Wenhao Li, Zihao Wang, Zekai Wang, Feng Yin, Junhua Zhao, and Xiuqiang He. Exploring large language model based intelligent agents: Definitions, methods, and prospects, 2024. URL https://arxiv.org/abs/2401.03428.

[26] Sahana Chennabasappa, Cyrus Nikolaidis, Daniel Song, David Molnar, Stephanie Ding, Shengye Wan, Spencer Whitman, Lauren Deason, Nicholas Doucette, Abraham Montilla, Alekhya Gampa, Beto de Paola, Dominik Gabi, James Crnkovich, Jean-Christophe Testud, Kat He, Rashnil Chaturvedi, Wu Zhou, and Joshua Saxe. Llamafirewall: An open source guardrail system for building secure AI agents, 2025. URL https://arxiv.org/abs/2505.03574.

[27] Florin Adrian Chitan. Session risk memory (SRM): Temporal authorization for deterministic pre-execution safety gates, 2026. URL https://arxiv.org/abs/2603.22350.

[28] Anthony Costarelli, Mat Allen, Roman Hauksson, Grace Sodunke, Suhas Hariharan, Carlson Cheng, Wenjie Li, Joshua Clymer, and Arjun Yadav. Gamebench: Evaluating strategic reasoning abilities of LLM agents, 2024. URL http://arxiv.org/abs/2406.06613.

[29] Mahavir Dabas, Jihyun Jeong, Ming Jin, and Ruoxi Jia. Memory-induced tool-drift in LLM agents, 2026. URL https://arxiv.org/abs/2605.24941.

[30] Edoardo Debenedetti, Jie Zhang, Mislav Balunović, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for llm agents. In *Advances in Neural Information Processing Systems 37*, pages 82895–82920. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2024. doi: 10.52202/079017-2636. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/97091a5177d8dc64b1da8bf3e1f6fb54-Abstract-Datasets_and_Benchmarks_Track.html.

[31] Edoardo Debenedetti, Ilia Shumailov, Tianqi Fan, Jamie Hayes, Nicholas Carlini, Daniel Fabian, Christoph Kern, Chongyang Shi, Andreas Terzis, and Florian Tramèr. Defeating prompt injections by design. In *IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, 2026. doi: 10.48550/arXiv.2503.18813. URL https://arxiv.org/abs/2503.18813.

[32] Xinhao Deng, Yixiang Zhang, Jiaqing Wu, Jiaqi Bai, Sibo Yi, Zhuoheng Zou, Yue Xiao, Rennai Qiu, Jianan Ma, Jialuo Chen, Xiaohu Du, Xiaofang Yang, Shiwen Cui, Changhua Meng, Weiqiang Wang, Jiaxing Song, Ke Xu, and Qi Li. Taming openclaw: Security analysis and mitigation of autonomous LLM agent threats, 2026. URL http://arxiv.org/abs/2603.11619.

[33] Zehang Deng, Yongjian Guo, Changzhou Han, Wanlun Ma, Junwu Xiong, Sheng Wen, and Yang Xiang. AI agents under threat: A survey of key security challenges and future pathways. *ACM Computing Surveys*, 57(7):1–36, 2025. doi: 10.1145/3716628. URL https://doi.org/10.1145/3716628.

[34] Dorothy E. Denning. A lattice model of secure information flow. *Communications of the ACM*, 19(5):236–243, 1976. doi: 10.1145/360051.360056. URL https://doi.org/10.1145/360051.360056.

[35] Jianshuo Dong, Sheng Guo, Hao Wang, Xun Chen, Zhuotao Liu, Tianwei Zhang, Ke Xu, Minlie Huang, and Han Qiu. Safesearch: Automated red-teaming of LLM-based search agents, 2026. URL https://arxiv.org/abs/2509.23694.

[36] Zhichen Dong, Zhanhui Zhou, Chao Yang, Jing Shao, and Yu Qiao. Attacks, defenses and evaluations for llm conversation safety: A survey. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6734–6747, Mexico City, Mexico, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.375. URL https://aclanthology.org/2024.naacl-long.375/.

[37] Ali Dorri, Salil S. Kanhere, and Raja Jurdak. Multi-agent systems: A survey. *IEEE Access*, 2018. doi: 10.1109/access.2018.2831228. URL https://doi.org/10.1109/access.2018.2831228.

[38] Mengyao Du, Han Fang, Haokai Ma, Jiahao Chen, Kai Xu, Quanjun Yin, and Ee-Chien Chang. SnapGuard: Lightweight prompt injection detection for screenshot-based web agents, 2026. URL https://arxiv.org/abs/2604.25562.

[39] Pengfei Du. Memory for autonomous llm agents: Mechanisms, evaluation, and emerging frontiers, 2026. URL https://arxiv.org/abs/2603.07670.

[40] Gaodan Fang, Vatche Isahagian, K. R. Jayaram, Ritesh Kumar, Vinod Muthusamy, Punleuk Oum, and Gegi Thomas. Trajectory-informed memory generation for self-improving agent systems, 2026. URL https://arxiv.org/abs/2603.10600.

[41] Richard Fang, Rohan Bindu, Akul Gupta, and Daniel Kang. LLM agents can autonomously exploit one-day vulnerabilities, 2024. URL http://arxiv.org/abs/2404.08144.

[42] Yunhao Feng, Xiaohu Du, Xinhao Deng, Yifan Ding, Ming Wen, Yixu Wang, Yuxiang Xie, Baihui Zheng, Yingshui Tan, Yige Li, Yutao Wu, Kerui Cao, Wenke Huang, Yanming Guo, Xingjun Ma, and Yu-Gang Jiang. BraveGuard: From open-world threats to safer computer-use agents, 2026. URL https://arxiv.org/abs/2606.01166.

[43] Marcelo Patricio Fernandez. Protocol-layer governance for LLM agents: Identity-bound oversight with zero-modification deployment and multi-hop authority propagation. *Zenodo (CERN European Organization for Nuclear Research)*, 2026. doi: 10.5281/zenodo.20162877. URL https://doi.org/10.5281/zenodo.20162877.

[44] Rogério Figurelli. From talking to blackbox to governance: Constitution mining for LLM agents. *Zenodo (CERN European Organization for Nuclear Research)*, 2026. doi: 10.5281/zenodo.18118668. URL https://doi.org/10.5281/zenodo.18118668.

[45] Kangning Gao, Haotian Zhu, Rui Liu, Jinming Li, Xu Yan, and Yi Hu. Contextual trust evaluation for robust coordination in large language model multi-agent systems, 2025. URL https://doi.org/10.1109/eiecc67963.2025.11409558.

[46] Caleb Geren, Amanda Board, Gaby G. Dagher, Tim Andersen, and Jun Zhuang. Blockchain for large language model security and safety: A holistic survey. *ACM SIGKDD Explorations Newsletter*, 2025. doi: 10.1145/3715073.3715075. URL https://doi.org/10.1145/3715073.3715075.

[47] Thibaud Gloaguen, Mark Vero, Robin Staab, and Martin Vechev. Watch your steps: Dormant adversarial behaviors that activate upon LLM finetuning, 2025. URL https://arxiv.org/abs/2505.16567.

[48] Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection, 2023. URL http://arxiv.org/abs/2302.12173.

[49] Chengquan Guo, Xun Liu, Chulin Xie, Andy Zhou, Yi Zeng, Zinan Lin, Dawn Song, and Bo Li. Redcode: Risky code execution and generation benchmark for code agents. In *Advances in Neural Information Processing Systems 37*, pages 106190–106236. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2024. doi: 10.52202/079017-3369. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/bfd082c452dffb450d5a5202b0419205-Abstract-Datasets_and_Benchmarks_Track.html.

[50] Jiahe Guo, Xiangran Guo, Yulin Hu, Zimo Long, Xingyu Sui, Xuda Zhi, Yongbo Huang, Hao He, Weixiang Zhao, Yanyan Zhao, and Bing Qin. When personalization legitimizes risks: Uncovering safety vulnerabilities in personalized dialogue agents. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens, editors, *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 27309–27335, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1260. URL https://aclanthology.org/2026.acl-long.1260/.

[51] Minghao Guo, Qingyue Jiao, Zeru Shi, Yihao Quan, Boxuan Zhang, Danrui Li, Liwei Che, Wujiang Xu, Shilong Liu, Zirui Liu, Mubbasir Kapadia, Vladimir Pavlovic, Jiang Liu, Mengdi Wang, Yiyu Shi, Dimitris N. Metaxas, and Ruixiang Tang. MemEye: A visual-centric evaluation framework for multimodal agent memory, 2026. URL https://arxiv.org/abs/2605.15128.

[52] Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents: A survey of progress and challenges, 2024. URL https://arxiv.org/abs/2402.01680.

[53] Saba Ghanbari Haez. Socio-normative trustworthiness of LLM agents: Evaluating autonomy support and representational fairness across languages and identities, 2026. URL https://doi.org/10.65109/lchb2977.

[54] Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. Parameter-efficient fine-tuning for large models: A comprehensive survey, 2024. URL https://arxiv.org/abs/2403.14608.

[55] Ari Harrison. Protocol-mediated execution governance for LLM agent systems. *SSRN Electronic Journal*, 2026. doi: 10.2139/ssrn.6426760. URL https://doi.org/10.2139/ssrn.6426760.

[56] Kostas Hatalis, Despina Christou, J. Myers, Steve Jones, Keith Lambert, Adam Amos-Binks, Zohreh A. Dannenhauer, and Dustin Dannenhauer. Memory matters: The need to improve long-term memory in LLM-agents. *Proceedings of the AAAI Symposium Series*, 2024. doi: 10.1609/aaaiss.v2i1.27688. URL https://doi.org/10.1609/aaaiss.v2i1.27688.

[57] Gaole He, Gianluca Demartini, and Ujwal Gadiraju. Plan-then-execute: An empirical study of user trust and team performance when using LLM agents as a daily assistant, 2025. URL https://doi.org/10.1145/3706598.3713218.

[58] Junda He, Christoph Treude, and David Lo. LLM-based multi-agent systems for software engineering: Literature review, vision and the road ahead. *ACM Transactions on Software Engineering and Methodology*, 34(5):1–30, 2025. doi: 10.1145/3712003. URL https://doi.org/10.1145/3712003.

[59] Keegan Hines, Gary Lopez, Matthew Hall, Federico Zarfati, Yonatan Zunger, and Emre Kiciman. Defending against indirect prompt injection attacks with spotlighting, 2024. URL http://arxiv.org/abs/2403.14720.

[60] Lauren Hong and Ting Wang. PETA: Parameter-efficient trojan attacks, 2024. URL https://arxiv.org/abs/2310.00648.

[61] Chia-Yi Hsu, Yu-Lin Tsai, Chih-Hsun Lin, Pin-Yu Chen, Chia-Mu Yu, and Chun-Ying Huang. Safe LoRA: The silver lining of reducing safety risks when fine-tuning large language models. In *Advances in Neural Information Processing Systems 37*, pages 65072–65094. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2024. doi: 10.52202/079017-2078. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/77baa7c2a3a675823e89131698fd6e19-Abstract-Conference.html.

[62] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

[63] Yannan Hu, Yu Wang, and Julian McAuley. Evaluating memory in LLM agents via incremental multi-turn interactions, 2025. URL http://arxiv.org/abs/2507.05257.

[64] Wenyue Hua, Xianjun Yang, Mingyu Jin, Zelong Li, Wei Cheng, Ruixiang Tang, and Yongfeng Zhang. Trustagent: Towards safe and trustworthy LLM-based agents, 2024. URL https://doi.org/10.18653/v1/2024.findings-emnlp.585.

[65] Zhaopei Huang, Qifeng Dai, Guozheng Wu, Xiaopeng Wu, Kehan Chen, Chuan Yu, Xubin Li, Tiezheng Ge, Wenxuan Wang, and Qin Jin. Mem-PAL: Towards memory-based personalized dialogue assistants for long-term user-agent interaction, 2025. URL https://arxiv.org/abs/2511.13410.

[66] Alex Iacob, Andrej Jovanović, William F. Shen, Daniel Burkhardt, Meghdad Kurmanji, Nurbek Tastan, Lorenzo Sani, Niccolò Alberto Elia Venanzi, Ambroise Odonnat, Zeyu Cao, Bill Marino, Xinchi Qiu, and Nicholas D. Lane. The red queen Gödel machine: Co-evolving agents and their evaluators, 2026. URL https://arxiv.org/abs/2606.26294.

[67] Ismail, Rahmat Kurnia, Zilmas Arjuna Brata, Ghitha Afina Nelistiani, Shinwook Heo, Hyeongon Kim, and Howon Kim. Toward robust security orchestration and automated response in security operations centers with a hyper-automation approach using agentic artificial intelligence. *Information*, 16(5):365, 2025. doi: 10.3390/info16050365. URL https://doi.org/10.3390/info16050365.

[68] Bhavvya Jain, Pranav Pawar, Dhruv Gada, Tanish Patwa, Pratik Kanani, Deepali Patil, and Lakshmi Kurup. Comprehensive analysis of machine learning and deep learning models on prompt injection classification using natural language processing techniques. *International Research Journal of Multidisciplinary Technovation*, 2025. doi: 10.54392/irjmt2523. URL https://doi.org/10.54392/irjmt2523.

[69] Jonathan D.A. Jewell. Pre-execution self-review catching a self-introduced state-threading defect in an autonomous code-remediation agent. *Open MIND*, 2026. doi: 10.5281/zenodo.20245469. URL https://github.com/hyperpolymath/technical-notes.

[70] Rishi Jha, Harold Triedman, Justin Wagle, and Vitaly Shmatikov. Breaking and fixing defenses against control-flow hijacking in multi-agent systems. In *The Fourteenth International Conference on Learning Representations*, 2026. doi: 10.48550/arXiv.2510.17276. URL https://openreview.net/forum?id=PNU9Rj5RDQ.

[71] Feiran Jia, Tong Wu, Xin Qin, and Anna Squicciarini. The task shield: Enforcing task alignment to defend against indirect prompt injection in llm agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 29680–29697, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.1435. URL https://aclanthology.org/2025.acl-long.1435/.

[72] Shian Jia, Ziyang Huang, Xinbo Wang, Haofei Zhang, and Mingli Song. PISA: A pragmatic psych-inspired unified memory system for enhanced AI agency, 2025. URL https://arxiv.org/abs/2510.15966.

[73] Tanqiu Jiang, Yuhui Wang, Jiacheng Liang, and Ting Wang. Agentlab: Benchmarking LLM agents against long-horizon attacks, 2026. URL http://arxiv.org/abs/2602.16901.

[74] Jefferson Kanjirakkattu Joseph, Esther Daniel, V. Kathiresan, and M. P. Prompt injection in large language model exploitation: A security perspective, 2025. URL https://doi.org/10.1109/iceccc65144.2025.11064209.

[75] Ratna Kandala, Niva Manchanda, Akshata Kishore Moharir, and Ananth Kandala. Echoguard: An agentic framework with knowledge-graph memory for detecting manipulative communication in longitudinal dialogue, 2026. URL https://arxiv.org/abs/2603.04815.

[76] Dhillon Andrew Kannabhiran. Consensus-validated memory improves agent performance on complex tasks. *Zenodo (CERN European Organization for Nuclear Research)*, 2026. doi: 10.5281/zenodo.18856774. URL https://doi.org/10.5281/zenodo.18856774.

[77] Md Monjurul Karim, Dong Hoang Van, Sangeen Khan, Qiang Qu, and Yaroslav Kholodov. AI agents meet blockchain: A survey on secure and scalable collaboration for multi-agents. *Future Internet*, 2025. doi: 10.3390/fi17020057. URL https://doi.org/10.3390/fi17020057.

[78] Rafflesia Khan, Declan Joyce, and Mansura Habiba. AGENTSAFE: A unified framework for ethical assurance and governance in agentic AI, 2025. URL https://arxiv.org/abs/2512.03180.

[79] Ninad Kulkarni, Xian Wu, Siddharth Varia, and Dmitriy Bespalov. Agent vs. agent: Automated data generation and red-teaming for custom agentic workflows. In Saloni Potdar, Lina Rojas-Barahona, and Sebastien Montella, editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 912–936, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-333-3. doi: 10.18653/v1/2025.emnlp-industry.62. URL https://aclanthology.org/2025.emnlp-industry.62/.

[80] Sonu Kumar. A multi-evaluator behavioural governance framework for LLM agent deployment. *Zenodo (CERN European Organization for Nuclear Research)*, 2026. doi: 10.5281/zenodo.19394881. URL https://doi.org/10.5281/zenodo.19394881.

[81] Chingkwun Lam, Jiaxin Li, Lingfei Zhang, and Kuo Zhao. Governing evolving memory in LLM agents: Risks, mechanisms, and the stability and safety governed memory (SSGM) framework, 2026. URL https://arxiv.org/abs/2603.11768.

[82] Qianlong Lan, Anuj Kaul, and Shaun Jones. Prompt injection detection in LLM integrated applications. *International Journal of Network Dynamics and Intelligence*, 4(2), 2025. doi: 10.53941/ijndi.2025.100013. URL https://doi.org/10.53941/ijndi.2025.100013.

[83] Hugo Larcher, Adrien Carreira, Raphael G., and Christophe Rannou. Anatomy of a frontier lab agent intrusion: A technical timeline of the july 2026 incident. Hugging Face Blog, July 2026. URL https://huggingface.co/blog/agent-intrusion-technical-timeline. Published 27 July 2026; accessed 17 August 2026.

[84] Donghyun Lee and Mo Tiwari. Prompt infection: Llm-to-llm prompt injection within multi-agent systems. *arXiv preprint arXiv:2410.07283*, 2024.

[85] Juyong Lee, Dongyoon Hahm, June Suk Choi, W. Bradley Knox, and Kimin Lee. Mobilesafetybench: Evaluating safety of autonomous agents in mobile device control. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(44):37565–37573, 2026. doi: 10.1609/aaai.v40i44.41090. URL https://ojs.aaai.org/index.php/AAAI/article/view/41090.

[86] S.G. Lee, J.-J. Kim, and Wooguil Pak. Mind mapping prompt injection: Visual prompt injection attacks in modern large language models. *Electronics*, 2025. doi: 10.3390/electronics14101907. URL https://doi.org/10.3390/electronics14101907.

[87] Simon Lermen, Charlie Rogers-Smith, and Jeffrey Ladish. Lora fine-tuning efficiently undoes safety training in llama 2-chat 70b. *arXiv preprint arXiv:2310.20624*, 2023. doi: 10.48550/arxiv.2310.20624. URL http://arxiv.org/abs/2310.20624.

[88] Martin Leucker and Christian Schallhart. A brief account of runtime verification. *The Journal of Logic and Algebraic Programming*, 78(5):293–303, 2009. doi: 10.1016/j.jlap.2008.08.004. URL https://doi.org/10.1016/j.jlap.2008.08.004.

[89] Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for "mind" exploration of large language model society. In *Advances in Neural Information Processing Systems 36*, pages 51991–52008. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2023. doi: 10.52202/075280-2264. URL https://proceedings.neurips.cc/paper/2023/hash/a3621ee907def47c1b952ade25c67698-Abstract-Conference.html.

[90] Haitao Li, Junjie Chen, Jingli Yang, Qingyao Ai, Wei Jia, Youfeng Liu, Kai Lin, Yueyue Wu, Guozhi Yuan, Yiran Hu, Wuyue Wang, Yiqun Liu, and Minlie Huang. LegalAgentBench: Evaluating LLM agents in legal domain. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2322–2344, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.116. URL https://aclanthology.org/2025.acl-long.116/.

[91] Hao Li, Xiaogeng Liu, Hung-Chun Chiu, Dianqi Li, Ning Zhang, and Chaowei Xiao. DRIFT: Dynamic rule-based defense with injection isolation for securing LLM agents. In *Advances in Neural Information Processing Systems*, volume 38, pages 83262–83290. Curran Associates, Inc., 2025. doi: 10.48550/arXiv.2506.12104. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/77f3b26c7907aa27b207df9b9d43f29a-Paper-Conference.pdf.

[92] Jing-Jing Li, Jianfeng He, Chao Shang, Devang Kulshreshtha, Xun Xian, Yi Zhang, Hang Su, Sandesh Swamy, and Yanjun Qi. STAC: When innocent tools form dangerous chains to jailbreak LLM agents, 2026. URL https://arxiv.org/abs/2509.25624.

[93] Mufei Li, Shikun Liu, Dongqi Fu, Haoyu Wang, Yinglong Xia, Hong Li, Hong Yan, and Pan Li. KVEraser: Learning to steer KV cache for efficient localized context erasing, 2026. URL https://arxiv.org/abs/2606.17034.

[94] Nathaniel Li, Alexander Pan, Anjali Gopal, Summer Yue, Daniel Berrios, Alice Gatti, Justin D. Li, Ann-Kathrin Dombrowski, Shashwat Goel, Gabriel Mukobi, Nathan Helm-Burger, Rassin Lababidi, Lennart Justen, Andrew Bo Liu, Michael Chen, Isabelle Barrass, Oliver Zhang, Xiaoyuan Zhu, Rishub Tamirisa, Bhrugu Bharathi, Ariel Herbert-Voss, Cort B. Breuer, Andy Zou, Mantas Mazeika, Zifan Wang, Palash Oswal, Weiran Lin, Adam Alfred Hunt, Justin Tienken-Harder, Kevin Y. Shih, Kemper Talley, John Guan, Ian Steneker, David Campbell, Brad Jokubaitis, Steven Basart, Stephen Fitz, Ponnurangam Kumaraguru, Kallol Krishna Karmakar, Uday Tupakula, Vijay Varadharajan, Yan Shoshitaishvili, Jimmy Ba, Kevin M. Esvelt, Alexandr Wang, and Dan Hendrycks. The WMDP benchmark: Measuring and reducing malicious use with unlearning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 28525–28550. PMLR, 2024. URL https://proceedings.mlr.press/v235/li24bc.html.

[95] Peiran Li, Jiashuo Sun, Fangzhou Lin, Shuo Xing, Tianfu Fu, Suofei Feng, Chaoqun Ni, and Zhengzhong Tu. Traversal-as-policy: Log-distilled gated behavior trees as externalized, verifiable policies for safe, robust, and efficient agents, 2026. URL https://arxiv.org/abs/2603.05517.

[96] Sijia Li, Yuchen Huang, Zifan Liu, Zijian Li, Jingjing Fu, Lei Song, Jiang Bian, Jun Zhang, and Rui Wang. Experience-evolving multi-turn tool-use agent with hybrid episodic-procedural memory, 2026. URL https://arxiv.org/abs/2512.07287.

[97] Xindi Li, Zhe Liu, Tong Zhang, Jiahao Chen, Qingming Li, Jinbao Li, and Shouling Ji. TWIST: Text-encoder weight-editing for inserting secret trojans in text-to-image models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11025–11041, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.541. URL https://aclanthology.org/2025.acl-long.541/.

[98] Xu Li, Simon Yu, Minzhou Pan, Yiyou Sun, Bo Li, Dawn Song, Xue Lin, and Weiyan Shi. Unsafer in many turns: Benchmarking and defending multi-turn safety risks in tool-using agents, 2026. URL https://arxiv.org/abs/2602.13379.

[99] Yibo Li, Zijie Lin, Ailin Deng, Xuan Zhang, Yufei He, Shuo Ji, Tri Cao, and Bryan Hooi. Just-in-time reinforcement learning: Continual learning in LLM agents without gradient updates, 2026. URL https://arxiv.org/abs/2601.18510.

[100] Yuanchun Li, Hao Wen, Weijun Wang, Xiangyu Li, Yizhen Yuan, Guohong Liu, Jiacheng Liu, Wenxing Xu, Xiang Wang, Yi Sun, Rui Kong, Yile Wang, Hanfei Geng, Jian Luan, Xuefeng Jin, Zilong Ye, Guanjing Xiong, Fan Zhang, Xiang Li, Mengwei Xu, Zhijun Li, Peng Li, Yang Liu, Ya-Qin Zhang, and Yunxin Liu. Personal LLM agents: Insights and survey about the capability, efficiency and security, 2024. URL https://arxiv.org/abs/2401.05459.

[101] Zeyi Liao, Jaylen Jones, Linxi Jiang, Yuting Ning, Eric Fosler-Lussier, Yu Su, Zhiqiang Lin, and Huan Sun. RedTeamCUA: Realistic adversarial testing of computer-use agents in hybrid web-OS environments. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=yWwrgcBoK3.

[102] Ruixiao Lin, Xinhao Deng, Qingming Li, Jianan Ma, Yunhao Feng, Yuqi Qing, Zhenyuan Li, Yechao Zhang, Shiwen Cui, Changhua Meng, Tianwei Zhang, Xingjun Ma, Qi Li, Ke Xu, and Shouling Ji. Safety in self-evolving LLM agent systems: Threats, amplification, and case studies, 2026. URL https://arxiv.org/abs/2606.23075.

[103] Zehao Lin, Xixuan Hao, Renyu Fu, Shaobo Cui, Kai Chen, Chunyu Li, Zhiyu Li, and Feiyu Xiong. A survey on long-term memory security in LLM agents: Attacks, defenses, and governance across the memory lifecycle, 2026. URL https://arxiv.org/abs/2604.16548.

[104] Daizong Liu, Mingyu Yang, Xiaoye Qu, Pan Zhou, Yu Cheng, and Wei Hu. A survey of attacks on large vision–language models: Resources, advances, and future trends. *IEEE Transactions on Neural Networks and Learning Systems*, 2025. doi: 10.1109/tnnls.2025.3592935. URL https://doi.org/10.1109/tnnls.2025.3592935.

[105] Genglin Liu, Shijie Geng, Sha Li, Hejie Cui, Sarah Zhang, Xin Liu, and Tianyi Liu. Webcoach: Self-evolving web agents with cross-session memory guidance, 2025. URL https://arxiv.org/abs/2511.12997.

[106] Hailin Liu, Eugene Ilyushin, Jie Ni, and Min Zhu. Safeagent: A runtime protection architecture for agentic systems, 2026. URL https://arxiv.org/abs/2604.17562.

[107] Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. Simplemem: Efficient lifelong memory for LLM agents, 2026. URL https://arxiv.org/abs/2601.02553.

[108] Jiaqi Liu, Xinyu Ye, Peng Xia, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. EvolveMem: Self-evolving memory architecture via AutoResearch for LLM agents, 2026. URL https://arxiv.org/abs/2605.13941.

[109] Shuochen Liu, Junyi Zhu, Long Shu, Junda Lin, Yuhao Chen, Haotian Zhang, Chao Zhang, Derong Xu, Jia Li, Bo Tang, Zhiyu Li, Feiyu Xiong, Enhong Chen, and Tong Xu. PERMA: Benchmarking personalized memory agents via event-driven preference and realistic task environments, 2026. URL https://arxiv.org/abs/2603.23231.

[110] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, volume 2024, pages 52989–53046, 2024.

[111] Xiaogeng Liu, Peiran Li, Edward Suh, Yevgeniy Vorobeychik, Zhuoqing Mao, Somesh Jha, Patrick McDaniel, Huan Sun, Bo Li, and Chaowei Xiao. AutoDAN-turbo: A lifelong agent for strategy self-exploration to jailbreak LLMs. In *International Conference on Learning Representations*, 2025. doi: 10.48550/arXiv.2410.05295. URL https://arxiv.org/abs/2410.05295.

[112] Yi Liu, Gelei Deng, Yuekang Li, Kailong Wang, Zihao Wang, Xiaofeng Wang, Tianwei Zhang, Yepang Liu, Haoyu Wang, Yan Zheng, Leo Yu Zhang, and Yang Liu. Prompt injection attack against LLM-integrated applications, 2023. URL https://arxiv.org/abs/2306.05499.

[113] Zesen Liu, Zihan Zhang, and Dongdong She. Safe to check, unsafe to use: Relinking at the compression boundary of LLM agents, 2026. URL https://arxiv.org/abs/2606.21732.

[114] Zhou Liu, Zhaoyang Han, Guochen Yan, Hao Liang, Bohan Zeng, Xing Chen, Yuanfeng Song, and Wentao Zhang. Datagovbench: Benchmarking LLM agents for real-world data governance workflows, 2025. URL https://arxiv.org/abs/2512.04416.

[115] Alireza Lotfi, Subangkar Karmaker Shanto, Imtiaz Karim, and Elisa Bertino. Securing agentic AI: From per-action checks to trajectory assurance, 2026. URL https://arxiv.org/abs/2608.01558. Accepted to the ACM AI Leadership Summit 2026, Visionary Track.

[116] Hanjun Luo, Shenyu Dai, Chiming Ni, Xinfeng Li, Guibin Zhang, Kun Wang, Tongliang Liu, and Hanan Salam. Agentauditor: Human-level safety and security evaluation for LLM agents. In *Advances in Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=2KKqp7MWJM.

[117] Jinghao Luo, Yuchen Tian, Chuxue Cao, Ziyang Luo, Hongzhan Lin, Kaixin Li, Chuyi Kong, Ruichao Yang, and Jing Ma. From storage to experience: A survey on the evolution of LLM agent memory mechanisms. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 41622–41652. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.2069. URL https://aclanthology.org/2026.findings-acl.2069/.

[118] Junyu Luo, Weizhi Zhang, Ye Yuan, Yusheng Zhao, Junwei Yang, Yiyang Gu, Bohan Wu, Binqi Chen, Ziyue Qiao, Qingqing Long, Rongcheng Tu, Xiao Luo, Wei Ju, Zhiping Xiao, Yifan Wang, Meng Xiao, Chenwu Liu, Jingyang Yuan, Shichang Zhang, Yiqiao Jin, Fan Zhang, Xian Wu, Hanqing Zhao, Dacheng Tao, Philip S. Yu, and Ming Zhang. Large language model agent: A survey on methodology, applications and challenges, 2025. URL https://arxiv.org/abs/2503.21460.

[119] Weidi Luo, Shenghong Dai, Xiaogeng Liu, Suman Banerjee, Huan Sun, Muhao Chen, and Chaowei Xiao. AGrail: A lifelong agent guardrail with effective and adaptive safety detection. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8104–8139, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.399. URL https://aclanthology.org/2025.acl-long.399/.

[120] Cristian Lupascu and Alexandru Lupascu. Elephantbroker: A knowledge-grounded cognitive runtime for trustworthy AI agents, 2026. URL https://arxiv.org/abs/2603.25097.

[121] Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. Agentboard: An analytical evaluation board of multi-turn LLM agents. In *Advances in Neural Information*

*Processing Systems 37*. Neural Information Processing Systems Foundation, Inc., 2024. doi: 10.52202/079017-2365. URL https://doi.org/10.52202/079017-2365.

[122] Jianan Ma, Xiaohu Du, Ruixiao Lin, Yaoxiang Bian, Jialuo Chen, Jingyi Wang, Xiaofang Yang, Shiwen Cui, Changhua Meng, Xinhao Deng, and Zhen Wang. Benchmarking autonomous agents against temporal, spatial, and semantic evasions, 2026. URL https://arxiv.org/abs/2605.22321.

[123] Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of llm agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13851–13870, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.747. URL https://aclanthology.org/2024.acl-long.747/.

[124] Xutao Mao, Liangjie Zhao, Leyao Wang, Rui Qian, Qiang Huang, Wentao Wang, Bo Han, Xiang Zheng, and Cong Wang. Agents don't just agree, they remember: Benchmarking persistent sycophancy in stateful personal agents, 2026. URL https://arxiv.org/abs/2607.10526.

[125] Xutao Mao, Liangjie Zhao, Xiang Zheng, and Cong Wang. Practice makes unsafe: Skill misevolution in self-improving LLM agents, 2026. URL https://arxiv.org/abs/2608.12851.

[126] Hongze Mi, Yibo Feng, WenJie Lu, Song Cao, Jinyuan Li, Yanming Li, Xuelin Zhang, Haotian Luo, Songyang Peng, He Cui, Tengfei Tian, Jun Fang, Hua Chai, and Naiqiang Tan. Darwinian memory: A training-free self-regulating memory system for GUI agent evolution, 2026. URL https://arxiv.org/abs/2601.22528.

[127] Microsoft. AI risk assessment for ML engineers. https://learn.microsoft.com/en-us/security/ai-red-team/ai-risk-assessment, 2024.

[128] Saroj Mishra, Suman Niroula, Umesh Yadav, Dilip Thakur, Srijan Gyawali, and Shiva Gaire. SoK: Agentic retrieval-augmented generation (RAG): Taxonomy, architectures, evaluation, and research directions, 2026. URL https://arxiv.org/abs/2603.07379.

[129] MITRE. MITRE ATLAS: Adversarial threat landscape for artificial-intelligence systems. https://atlas.mitre.org/, 2026.

[130] Akshay Mittal and Vivek Venkatesan. Practical integration of large language models into enterprise ci/cd pipelines for security policy validation: An industry-focused evaluation, 2025. URL https://doi.org/10.1109/sose67019.2025.00027.

[131] Mahmoud Mohammadi, Yipeng Li, Jane Lo, and Wendy Yip. Evaluation and benchmarking of llm agents: A survey. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, pages 6129–6139, New York, NY, USA, 2025. Association for Computing Machinery. doi: 10.1145/3711896.3736570. URL https://doi.org/10.1145/3711896.3736570.

[132] Ramasankar Molleti, Vinod Goje, Puneet Luthra, and Prathap Raghavan. Automated threat detection and response using LLM agents. *World Journal of Advanced Research and Reviews*, 2024. doi: 10.30574/wjarr.2024.24.2.3329. URL https://doi.org/10.30574/wjarr.2024.24.2.3329.

[133] Zahra Moslemi, Keerthi Koneru, Yen-Ting Lee, Sheethal Kumar, and Ramesh Radhakrishnan. POLARIS: Typed planning and governed execution for agentic AI in back-office automation, 2026. URL https://arxiv.org/abs/2601.11816.

[134] Manuel Mosquera, Juan Sebastián Pinzón, Yesid Fonseca, Manuel Ríos, Nicanor Quijano, Luis Felipe Giraldo, and Rubén Manrique. Can LLM-augmented autonomous agents cooperate? an evaluation of their cooperative capabilities through melting pot. *IEEE Transactions on Artificial Intelligence*, 2025. doi: 10.1109/tai.2025.3569192. URL https://doi.org/10.1109/tai.2025.3569192.

[135] Lajos Muzsai, David Imolai, and András Lukács. Hacksynth: LLM agent and evaluation framework for autonomous penetration testing, 2024. URL http://arxiv.org/abs/2412.01778.

[136] Tam Nguyen, Moses Ndebugre, and Dheeraj Arremsetty. Security considerations for multi-agent systems, 2026. URL https://arxiv.org/abs/2603.09002.

[137] OpenAI. OpenAI and Hugging Face partner to address security incident during model evaluation. OpenAI Security, July 2026. URL https://openai.com/index/hugging-face-model-evaluation-security-incident/. Published 21 July 2026; updated 29 July 2026; accessed 17 August 2026.

[138] OWASP Foundation. OWASP top 10 for large language model applications. https://owasp.org/www-project-top-10-for-large-language-model-applications/, 2025.

[139] Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems, 2023. URL https://arxiv.org/abs/2310.08560.

[140] Alexander Pan, Jun Shern Chan, Andy Zou, Nathaniel Li, Steven Basart, Thomas Woodside, Hanlin Zhang, Scott Emmons, and Dan Hendrycks. Do the rewards justify the means? measuring trade-offs between rewards and ethical behavior in the machiavelli benchmark. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 26837–26867. PMLR, 2023. URL https://proceedings.mlr.press/v202/pan23a.html.

[141] Samuel Panterino and Matthew Fellington. Dynamic moving target defense for mitigating targeted LLM prompt injection, 2024. URL http://dx.doi.org/10.36227/techrxiv.171822345.56781952/v1.

[142] Benji Peng, Hanxuan Chen, Keyu Chen, Qian Niu, Ziqian Bi, Ming Liu, Pohsun Feng, Tianyang Wang, Lawrence K. Q. Yan, Yizhu Wen, Yichao Zhang, Caitlyn Heqi Yin, Xinyuan Song, Riyang Bao, and Jiacheng Shi. Jailbreaking and mitigation of vulnerabilities in large language models, 2024. URL http://arxiv.org/abs/2410.15236.

[143] Mathis Pink, Qinyuan Wu, Vy A. Vo, Javier S. Turek, Jianing Mu, Alexander G. Huth, and Mariya Toneva. Position: Episodic memory is the missing piece for long-term LLM agents, 2025. URL http://arxiv.org/abs/2502.06975.

[144] Michał Podpora, M Baranowski, Maciej Chopcian, Lukasz Kwasniewicz, and W. Radziewicz. LLM firewall using validator agent for prevention against prompt injection attacks. *Applied Sciences*, 2025. doi: 10.3390/app16010085. URL https://doi.org/10.3390/app16010085.

[145] Chandra Prakash, Mary Lind, and Elyson De La Cruz. Hybrid real-time framework for detecting adaptive prompt injection attacks in large language models. *Journal of Computing Theories and Applications*, 2026. doi: 10.62411/jcta.15254. URL https://doi.org/10.62411/jcta.15254.

[146] Sidharth Pulipaka, Oliver Chen, Manas Sharma, Taaha S. Bajwa, Vyas Raina, and Ivaxi Sheth. Persistbench: When should long-term memories be forgotten by LLMs?, 2026. URL https://arxiv.org/abs/2602.01146.

[147] Sidharth Pulipaka, Stanislau Hlebik, Leonidas Raghav, Sahar Abdelnabi, Vyas Raina, Ivaxi Sheth, and Mario Fritz. Hidden in memory: Sleeper memory poisoning in llm agents, 2026. URL https://arxiv.org/abs/2605.15338.

[148] Jinhu Qi, Muzhi Li, Jiahong Liu, Yuqin Shu, Dianzhi Yu, Shicheng Ma, Wenqian Cui, Yiyang Zhao, Yiyi Chen, Ruoxi Jiang, Irwin King, and Zenglin Xu. Towards trustworthy agentic AI: A comprehensive survey of safety, robustness, privacy, and system security, 2026. URL https://arxiv.org/abs/2605.23989.

[149] Weiwei Qi, Zefeng Wu, Zhilin Guo, Tianhang Zheng, Chaochao Lu, Liang He, Zhan Qin, and Kui Ren. DARWIN: Evolving jailbreak adversary and guardrail for LLM safety evaluation and protection, 2026. URL https://arxiv.org/abs/2607.19829.

[150] Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. Fine-tuning aligned language models compromises safety, even when users do not intend to!, 2023. URL https://arxiv.org/abs/2310.03693.

[151] Brandon Radosevich and John Halloran. MCP safety audit: LLMs with the model context protocol allow major security exploits, 2025. URL https://arxiv.org/abs/2504.03767.

[152] Shreyas Rajesh, Pavan Holur, Mehmet Yigit Turali, Chenda Duan, and Vwani Roychowdhury. Panini: Continual learning in token space via structured memory, 2026. URL https://arxiv.org/abs/2602.15156.

[153] Javier Rando and Florian Tramèr. Universal jailbreak backdoors from poisoned human feedback, 2023. URL https://arxiv.org/abs/2311.14455.

[154] Zixin Rao, Wentian Zhu, Chan Aristella Lu, Zhaorun Chen, Wei Niu, Le Guan, Bo Li, and Zhen Xiang. FragFuse: Bypassing access control of large language model agents via memory-based query fragmentation and fusion, 2026. URL https://arxiv.org/abs/2606.15609.

[155] Vishal Rathod, Seyedsina Nabavirazavi, Samira Zad, and Sundararaja Sitharama Iyengar. Privacy and security challenges in large language models, 2025. URL https://doi.org/10.1109/ccwc62904.2025.10903912.

[156] Santhosh Kumar Ravindran. Portable agent memory: A protocol for cryptographically-verified memory transfer across heterogeneous AI agents, 2026. URL https://arxiv.org/abs/2605.11032.

[157] Pavan Reddy and Aditya Sanjay Gujral. Echoleak: The first real-world zero-click prompt injection exploit in a production LLM system. *Proceedings of the AAAI Symposium Series*, 2025. doi: 10.1609/aaaiss.v7i1.36899. URL https://doi.org/10.1609/aaaiss.v7i1.36899.

[158] Matthew Renze and Erhan Guven. Self-reflection in LLM agents: Effects on problem-solving performance, 2024. URL http://arxiv.org/abs/2405.06682.

[159] Oscar J. Romero, John Zimmerman, Aaron Steinfeld, and Anthony Tomasic. Synergistic integration of large language models and cognitive architectures for robust AI: An exploratory analysis. *Proceedings of the AAAI Symposium Series*, 2024. doi: 10.1609/aaaiss.v2i1.27706. URL https://doi.org/10.1609/aaaiss.v2i1.27706.

[160] Andrei Sabelfeld and Andrew C. Myers. Language-based information-flow security. *IEEE Journal on Selected Areas in Communications*, 21(1):5–19, 2003. doi: 10.1109/JSAC.2002.806121. URL https://doi.org/10.1109/JSAC.2002.806121.

[161] Ahmed Salem, Andrew Paverd, and Boris Köpf. Maatphor: Automated variant analysis for prompt injection attacks, 2023. URL http://arxiv.org/abs/2312.11513.

[162] Sander Schulhoff, Jeremy Pinto, Anaum Khan, Louis-François Bouchard, Chenglei Si, Svetlina Anati, Valen Tagliabue, Anson Liu Kost, Christopher Carnahan, and Jordan Boyd-Graber. Ignore this title and hackaprompt: Exposing systemic vulnerabilities of LLMs through a global scale prompt hacking competition. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4945–4977. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.302. URL https://aclanthology.org/2023.emnlp-main.302/.

[163] Shuai Shao, Qihan Ren, Chen Qian, Boyi Wei, Dadi Guo, Jingyi Yang, Xinhao Song, Linfeng Zhang, Weinan Zhang, Dongrui Liu, and Jing Shao. Your agent may misevolve: Emergent risks in self-evolving LLM agents. In *International Conference on Learning Representations*, 2026. doi: 10.48550/arXiv.2509.26354. URL https://arxiv.org/abs/2509.26354.

[164] Natalie Shapira, Chris Wendler, Avery Yen, Gabriele Sarti, Koyena Pal, Olivia Floody, Adam Belfki, Alex Loftus, Aditya Ratan Jannali, Nikhil Prakash, Jasmine Cui, Giordano Rogers, Jannik Brinkmann, Can Rager, Amir Zur, Michael Ripa, Aruna Sankaranarayanan, David Atkinson, Rohit Gandikota, Jaden Fiotto-Kaufman, EunJeong Hwang, Hadas Orgad, P Sam Sahil, Negev Taglicht, Tomer Shabtay, Atai Ambus, Nitay Alon, Shiri Oron, Ayelet Gordon-Tapiero, Yotam Kaplan, Vered Shwartz, Tamar Rott Shaham, Christoph Riedl, Reuth Mirsky, Maarten Sap, David Manheim, Tomer Ullman, and David Bau. Agents of chaos, 2026. URL https://arxiv.org/abs/2602.20021.

[165] Xinjie Shen, Rongzhe Wei, Peizhi Niu, Haoyu Wang, Ruihan Wu, Eli Chien, Bo Li, Pin-Yu Chen, and Pan Li. One turn too late: Response-aware defense against hidden malicious intent in multi-turn dialogue, 2026. URL https://arxiv.org/abs/2605.05630.

[166] Jiawen Shi, Zenghui Yuan, Guiyao Tie, Pan Zhou, Neil Zhenqiang Gong, and Lichao Sun. Prompt injection attack to tool selection in LLM agents. In *Network and Distributed System Security Symposium*, 2026. doi: 10.14722/ndss.2026.230675. URL https://www.ndss-symposium.org/ndss-paper/prompt-injection-attack-to-tool-selection-in-llm-agents/.

[167] Tianneng Shi, Jingxuan He, Zhun Wang, Hongwei Li, Linyu Wu, Wenbo Guo, and Dawn Song. Progent: Securing AI agents with privilege control, 2025. URL https://arxiv.org/abs/2504.11703.

[168] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems 36*. Neural Information Processing Systems Foundation, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

[169] Dong Shu, Chong Zhang, Mingyu Jin, Zihao Zhou, and Lingyao Li. Attackeval: How to evaluate the effectiveness of jailbreak attacking on large language models. *ACM SIGKDD Explorations Newsletter*, 2025. doi: 10.1145/3748239.3748242. URL https://doi.org/10.1145/3748239.3748242.

[170] Vincent Siu, Jingxuan He, Kyle Montgomery, Zhun Wang, Neil Gong, Chenguang Wang, and Dawn Song. A framework for formalizing LLM agent security, 2026. URL https://arxiv.org/abs/2603.19469.

[171] Renan Souza, Timothy Poteet, Brian D. Etz, Daniel Rosendo, Amal Gueroudji, Woong Shin, Prasanna Balaprakash, and Rafael Ferreira da Silva. LLM agents for interactive workflow provenance: Reference architecture and evaluation methodology, 2025. URL https://doi.org/10.1145/3731599.3767582.

[172] Saksham Sahai Srivastava and Haoyu He. Memorygraft: Persistent compromise of LLM agents via poisoned experience retrieval, 2025. URL https://arxiv.org/abs/2512.16962.

[173] Changzhi Sun, Xiangyu Chen, Jixiang Luo, Dell Zhang, and Xuelong Li. Beyond heuristics: A decision-theoretic framework for agent memory management, 2025. URL https://arxiv.org/abs/2512.21567.

[174] Balachandra Devarangadi Sunil, Isheeta Sinha, Piyush Maheshwari, Shantanu Todmal, Shreyan Mallik, and Shuchi Mishra. Memory poisoning attack and defense on memory based llm-agents, 2026. URL https://arxiv.org/abs/2601.05504.

[175] Xuchen Suo. Signed-prompt: A new approach to prevent prompt injection attacks against LLM-integrated applications, 2024. URL https://arxiv.org/abs/2401.07612.

[176] Georgios Syros, Anshuman Suri, Jacob Ginesin, Cristina Nita-Rotaru, and Alina Oprea. SAGA: A security architecture for governing AI agentic systems. In *Network and Distributed System Security Symposium*, 2026. doi: 10.14722/ndss.2026.230869. URL https://www.ndss-symposium.org/ndss-paper/saga-a-security-architecture-for-governing-ai-agentic-systems/.

[177] Elham Tabassi. Artificial intelligence risk management framework (ai rmf 1.0). (NIST AI 100-1), 2023. doi: 10.6028/NIST.AI.100-1. URL https://doi.org/10.6028/NIST.AI.100-1.

[178] Hamed Taheri. Governed memory: A production architecture for multi-agent workflows, 2026. URL https://arxiv.org/abs/2603.17787.

[179] Zhen Tao, Jinxiang Zhao, Peng Liu, Dinghao Xi, Yanfang Chen, Wei Xu, and Zhiyu Li. MemConflict: Evaluating long-term memory systems under memory conflicts, 2026. URL https://arxiv.org/abs/2605.20926.

[180] Zhengwei Tao, Ting-En Lin, Xiancai Chen, Hangyu Li, Yuchuan Wu, Yongbin Li, Zhi Jin, Fei Huang, Dacheng Tao, and Jingren Zhou. A survey on self-evolution of large language models, 2024. URL https://arxiv.org/abs/2404.14387.

[181] Yingzhe Tong, Leyu Dai, and Songhui Guo. AID-Guard: Stateful authorization for delegated agent effects, 2026. URL https://arxiv.org/abs/2608.21159.

[182] Vicenç Torra and Maria Bras-Amorós. Memory poisoning and secure multi-agent systems, 2026. URL https://arxiv.org/abs/2603.20357.

[183] George Torres, Sharad Shrestha, and Satyajayant Misra. When agents remember too much: Memory poisoning attacks on large language model agents, 2026. URL https://arxiv.org/abs/2607.06595.

[184] Harold Triedman, Rishi Dev Jha, and Vitaly Shmatikov. Multi-agent systems execute arbitrary malicious code. In *Proceedings of the Conference on Language Modeling (COLM)*, 2025. doi: 10.48550/arXiv.2503.12188. URL https://openreview.net/forum?id=DAozI4etUp.

[185] Isaac Triguero, Daniel Molina, Javier Poyatos, Javier Del Ser, and Francisco Herrera. General purpose artificial intelligence systems (GPAIS): Properties, definition, taxonomy, societal implications and responsible governance. *Information Fusion*, 2023. doi: 10.1016/j.inffus.2023.102135. URL https://doi.org/10.1016/j.inffus.2023.102135.

[186] Giorgio Dalla Volta, Giacomo Longo, and Alessio Merlo. Experimental campaign for political behaviour in LLM agents: Experimental insights on governance, global power asymmetries, security risks, and knowledge construction dynamics. *Zenodo (CERN European Organization for Nuclear Research)*, 2026. doi: 10.5281/zenodo.19052474. URL https://doi.org/10.5281/zenodo.19052474.

[187] Bo Wang, Weiyi He, Shenglai Zeng, Zhen Xiang, Yue Xing, Jiliang Tang, and Pengfei He. Unveiling privacy risks in llm agent memory. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 25241–25260. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.1227. URL https://aclanthology.org/2025.acl-long.1227/.

[188] Chenxu Wang, Chaozhuo Li, Songyang Liu, Zejian Chen, Jinyu Hou, Ji Qi, Rui Li, Litian Zhang, Qiwei Ye, Zheng Liu, Xu Chen, Xi Zhang, and Philip S. Yu. The devil behind moltbook: Anthropic safety is always vanishing in self-evolving AI societies, 2026. URL https://arxiv.org/abs/2602.09877.

[189] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models, 2023. URL https://arxiv.org/abs/2305.16291.

[190] Junjian Wang, Lidan Zhao, and Xi Sheryl Zhang. MADRA: Multi-agent debate for risk-aware embodied planning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 6852–6876. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.340. URL https://aclanthology.org/2026.findings-acl.340/.

[191] Kaixiang Wang, Yidan Lin, Jiong Lou, Zhaojiacheng Zhou, Bunyod Suvonov, and Jie Li. E-mem: Multi-agent based episodic context reconstruction for LLM agent memory, 2026. URL https://arxiv.org/abs/2601.21714.

[192] Kaixiang Wang, Jiong Lou, Zhaojiacheng Zhou, and Jie Li. Oep: Poisoning self-evolving llm agents via locally correct but non-transferable experiences, 2026. URL https://arxiv.org/abs/2605.18930.

[193] Le Wang, Zonghao Ying, Tianyuan Zhang, Siyuan Liang, Shengshan Hu, Mingchuan Zhang, Aishan Liu, and Xianglong Liu. Manipulating multimodal agents via cross-modal prompt injection, 2025. URL https://doi.org/10.1145/3746027.3755211.

[194] Qihao Wang, Ziming Cheng, Shuo Zhang, Fan Liu, Rui Xu, Heng Lian, Kunyi Wang, Xiaoming Yu, Jianghao Yin, Sen Hu, Yue Hu, Shaolei Zhang, Yanbing Liu, Ronghao Chen, and Huacan Wang. Memgovern: Enhancing code agents through learning from governed human experiences, 2026. URL https://arxiv.org/abs/2601.06789.

[195] Shang Wang, Tianqing Zhu, Bo Liu, Ming Ding, Dayong Ye, Wanlei Zhou, and Philip S. Yu. Unique security and privacy threats of large language models: A comprehensive survey, 2024. URL https://arxiv.org/abs/2406.07973.

[196] Thomas Wang and Haowen Li. Openguardrails: A configurable, unified, and scalable guardrails platform for large language models, 2025. URL https://arxiv.org/abs/2510.19169.

[197] Yifei Wang, Dizhan Xue, Shengjie Zhang, and Shengsheng Qian. Badagent: Inserting and activating backdoor attacks in LLM agents, 2024. URL https://doi.org/10.18653/v1/2024.acl-long.530.

[198] Yiming Wang, Jiahao Chen, Qingming Li, Tong Zhang, Rui Zeng, Xing Yang, and Shouling Ji. AEIOU: A unified defense framework against NSFW prompts in text-to-image models, 2025. URL https://arxiv.org/abs/2412.18123.

[199] Yixu Wang, Xin Wang, Yang Yao, Xinyuan Li, Xibang Yang, Yan Teng, Xingjun Ma, and Yingchun Wang. Agenticeval: Toward agentic and self-evolving safety evaluation of large language models. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 14789–14808. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.727. URL https://aclanthology.org/2026.findings-acl.727/.

[200] Yuhao Wang, Shengfang Zhai, Guanghao Jin, Yinpeng Dong, Linyi Yang, and Jiaheng Zhang. Mempot: Defending against memory extraction attack with optimized honeypots, 2026. URL https://arxiv.org/abs/2602.07517.

[201] Yuhui Wang, Tanqiu Jiang, Jiacheng Liang, Charles Fleming, and Ting Wang. MAGE: Safeguarding LLM agents against long-horizon threats via shadow memory, 2026. URL https://arxiv.org/abs/2605.03228.

[202] Zhun Wang, Vincent Siu, Zhe Ye, Tianneng Shi, Yuzhou Nie, Xuandong Zhao, Chenguang Wang, Wenbo Guo, and Dawn Song. AGENTVIGIL: Automatic black-box red-teaming for indirect prompt injection against LLM agents. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 23159–23172, Suzhou, China, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-emnlp.1258. URL https://aclanthology.org/2025.findings-emnlp.1258/.

[203] Zhun Wang, Nico Schiller, Hongwei Li, Srijiith Sesha Narayana, Milad Nasr, Nicholas Carlini, Xiangyu Qi, Eric Wallace, Elie Bursztein, Luca Invernizzi, Kurt Thomas, Yan Shoshitaishvili, Wenbo Guo, Jingxuan He, Thorsten Holz, and Dawn Song. ExploitGym: Can AI agents turn security vulnerabilities into real attacks?, May 2026. URL https://arxiv.org/abs/2605.11086.

[204] Qianshan Wei, Tengchao Yang, Yaochen Wang, Xinfeng Li, Lijun Li, Zhenfei Yin, Yi Zhan, Thorsten Holz, Zhiqiang Lin, and XiaoFeng Wang. A-memguard: A proactive defense framework for LLM-based agent memory, 2025. URL https://arxiv.org/abs/2510.02373.

[205] Rongzhe Wei, Peizhi Niu, Xinjie Shen, Tony Tu, Yifan Li, Ruihan Wu, Eli Chien, Pin-Yu Chen, Olgica Milenkovic, and Pan Li. The trojan knowledge: Bypassing commercial LLM guardrails via harmless prompt weaving and adaptive tree search, 2025. URL https://arxiv.org/abs/2512.01353.

[206] Xiaodong Wu, Yu Shi, Qi Li, Zhimin Zhao, Xiangman Li, Bram Adams, Ahmed E. Hassan, and Jianbing Ni. EVOMAL: Self-poisoning in self-evolving coding agents, 2026. URL https://arxiv.org/abs/2608.25776.

[207] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wensen Cheng, Qi Zhang, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huang, and Tao Gui. The rise and potential of large language model based agents: A survey, 2023. URL https://arxiv.org/abs/2309.07864.

[208] Boming Xia, Qinghua Lu, Liming Zhu, Zhenchang Xing, Dehai Zhao, and Hao Zhang. Evaluation-driven development and operations of LLM agents: A process model and reference architecture, 2024. URL https://arxiv.org/abs/2411.13768.

[209] 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. GuardAgent: Safeguard llm agents by a guard agent via knowledge-enabled reasoning, 2024. URL https://arxiv.org/abs/2406.09187.

[210] Weiwei Xie, Shaoxiong Guo, Fan Zhang, Tian Xia, Xue Yang, Lizhuang Ma, Junchi Yan, and Qibing Ren. MemEvoBench: Benchmarking safety risks from memory misevolution in LLM agents, 2026. URL https://arxiv.org/abs/2604.15774.

[211] Mingzhe Xing, Rongkai Zhang, Hui Xue, Qi Chen, Fan Yang, and Zhen Xiao. Understanding the weakness of large language model agents within a complex android environment, 2024. URL https://doi.org/10.1145/3637528.3671650.

[212] Wenjun Xiong, Yijin Zhou, Jiaqian Wang, Shangding Gu, Bo Tang, Zhiyu Li, Feiyu Xiong, Ying Wen, and Muning Wen. MAPLE-Guard: Memory-aware link enforcement against memory-link poisoning in multi-agent systems, 2026. URL https://arxiv.org/abs/2608.00426.

[213] Lingxiang Xu, Jiaoyun Yang, Min Hu, Hongtu Chen, and Ning An. When should memory stay silent: Measuring memory-use boundaries in memory-augmented conversational agents, 2026. URL https://arxiv.org/abs/2606.06055.

[214] Naen Xu, Hengyu An, Shuo Shi, Jinghuai Zhang, Chunyi Zhou, Changjiang Li, Tianyu Du, Zhihui Fu, Jun Wang, and Shouling Ji. When agents "Misremember" collectively: Exploring the mandela effect in LLM-based multi-agent systems. In *International Conference on Learning Representations*, 2026. doi: 10.48550/arXiv.2602.00428. URL https://openreview.net/forum?id=yIoMqDes7O.

[215] Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for LLM agents, 2025. URL http://arxiv.org/abs/2502.12110.

[216] Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. A survey on knowledge distillation of large language models, 2024. URL http://arxiv.org/abs/2402.13116.

[217] Bingyu Yan, Xiaoming Zhang, Chaozhuo Li, Ziyi Zhou, Yirui Qi, and Litian Zhang. Benign alone, harmful together: Exploiting experience composition in self-evolving LLM agents, 2026. URL https://arxiv.org/abs/2608.01759.

[218] Biwei Yan, Kun Li, Minghui Xu, Yueyan Dong, Yue Zhang, Zhaochun Ren, and Xiuzhen Cheng. On protecting the data privacy of large language models (LLMs) and LLM agents: A literature review. *High-Confidence Computing*, 2025. doi: 10.1016/j.hcc.2025.100300. URL https://doi.org/10.1016/j.hcc.2025.100300.

[219] Hongming Yang, Hao Liu, Xin Yuan, Kai Wu, Wei Ni, J. Andrew Zhang, and Ren Ping Liu. Synergizing intelligence and privacy: A review of integrating internet of things, large language models, and federated learning in advanced networked systems. *Applied Sciences*, 15(12):6587, 2025. doi: 10.3390/app15126587. URL https://www.mdpi.com/2076-3417/15/12/6587.

[220] Tingting Yang, Ping Feng, Qixin Guo, Jindi Zhang, Xiufeng Zhang, Jiahong Ning, Xinghan Wang, and Zhongyang Mao. Autohma-LLM: Efficient task coordination and execution in heterogeneous multi-agent systems using hybrid large language models. *IEEE Transactions on Cognitive Communications and Networking*, 2025. doi: 10.1109/tccn.2025.3528892. URL https://doi.org/10.1109/tccn.2025.3528892.

[221] Wenxian Yang, Hanzheng Qiu, Bangqun Zhang, Chengquan Li, Zhiyong Huang, Xiaobin Feng, Rongshan Yu, and Jiahong Dong. When openclaw meets hospital: Toward an agentic operating system for dynamic clinical workflows, 2026. URL https://arxiv.org/abs/2603.11721.

[222] Xianglin Yang, Yufei He, Shuo Ji, Bryan Hooi, and Jin Song Dong. Zombie agents: Persistent control of self-evolving LLM agents via self-reinforcing injections, 2026. URL https://arxiv.org/abs/2602.15654.

[223] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2022. URL https://arxiv.org/abs/2210.03629.

[224] Chenchen Ye, Ziniu Hu, Yihe Deng, Zijie Huang, Mingyu Derek Ma, Yanqiao Zhu, and Wei Wang. Mirai: Evaluating LLM agents for event forecasting, 2024. URL http://arxiv.org/abs/2407.01231.

[225] Qiming Ye, Peixain Zhang, Yupeng He, Zifan Peng, and Gareth Tyson. Behind EvoMap: Characterizing a self-evolving agent-to-agent collaboration network, 2026. URL https://arxiv.org/abs/2605.25815.

[226] Caglar Yildirim. Differential harm propensity in personalized LLM agents: The curious case of mental health disclosure, 2026. URL https://arxiv.org/abs/2603.16734.

[227] Xingkun Yin and Hongyang Du. GLOVE: Global verifier for LLM memory-environment realignment, 2026. URL https://arxiv.org/abs/2601.19249.

[228] Zonghao Ying, Xiangfan Wu, Huiyu Wu, Xing Zheng, Huangsheng Cheng, Xiaorong Shi, and Jing Guo. SkillJack: Persistent skill backdoors in self-evolving agents, 2026. URL https://arxiv.org/abs/2608.03509.

[229] Miao Yu, Fanci Meng, Xinyun Zhou, Shilong Wang, Junyuan Mao, Linsey Pang, Tianlong Chen, Kun Wang, Xinfeng Li, Yongfeng Zhang, Bo An, and Qingsong Wen. A survey on trustworthy llm agents: Threats and countermeasures, 2025. URL https://arxiv.org/abs/2503.09648.

[230] Xinyi Yu, Xiang Yin, Shaoyuan Li, and Zhaojian Li. Security-preserving multi-agent coordination for complex temporal logic tasks. *Control Engineering Practice*, 2022. doi: 10.1016/j.conengprac.2022.105130. URL https://doi.org/10.1016/j.conengprac.2022.105130.

[231] Jiayi Yuan, Jonathan Nöther, Natasha Jaques, and Goran Radanović. Agenticred: Evolving agentic systems for red-teaming, 2026. URL https://arxiv.org/abs/2601.13518.

[232] Lei Yuan, Ziqian Zhang, Ke Xue, Hao Yin, Feng Chen, Cong Guan, Lihe Li, Chao Qian, and Yang Yu. Robust multi-agent coordination via evolutionary generation of auxiliary adversarial attackers. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023. doi: 10.1609/aaai.v37i10.26388. URL https://doi.org/10.1609/aaai.v37i10.26388.

[233] Tongxin Yuan, Zhiwei He, Lingzhong Dong, Yiming Wang, Ruijie Zhao, Tian Xia, Lizhen Xu, Binglin Zhou, Fangqi Li, Zhuosheng Zhang, Rui Wang, and Gongshen Liu. R-judge: Benchmarking safety risk awareness for llm agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1384–1400, Miami, Florida, USA, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.79. URL https://aclanthology.org/2024.findings-emnlp.79/.

[234] Puyu Zeng, Simeng Qin, Jingzhi Li, Ju Jia, Zheli Liu, and Xiaojun Jia. ColluSkill: Adversarial cross-skill composition for evading agent skill scanners, 2026. URL https://arxiv.org/abs/2608.09732.

[235] Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. Injecagent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10471–10506, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.624. URL https://aclanthology.org/2024.findings-acl.624/.

[236] Qiusi Zhan, Richard Fang, Henil Shalin Panchal, and Daniel Kang. Adaptive attacks break defenses against indirect prompt injection attacks on llm agents. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 7116–7132, Albuquerque, New Mexico, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-naacl.395. URL https://aclanthology.org/2025.findings-naacl.395/.

[237] Qiuyang Zhan, Rui Zhang, Sheng Guo, Lepeng Zhao, and Zhuotao Liu. When memory becomes authority: Benchmarking authority collapse at the memory consolidation boundary, 2026. URL https://arxiv.org/abs/2608.01679.

[238] Boyang Zhang, Yicong Tan, Yun Shen, Ahmed Salem, Michael Backes, Savvas Zannettou, and Yang Zhang. Breaking agents: Compromising autonomous LLM agents through malfunction amplification. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 34952–34964. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.1771. URL https://aclanthology.org/2025.emnlp-main.1771/.

[239] Ce Zhang, Jinxi He, Junyi He, Katia Sycara, and Yaqi Xie. Evolving contextual safety in multi-modal large language models via inference-time self-reflective memory. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 41182–41192, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Zhang_Evolving_Contextual_Safety_in_Multi-Modal_Large_Language_Models_via_Inference-Time_CVPR_2026_paper.html.

[240] Hanrong Zhang, Jingyuan Huang, Kai Mei, Yifei Yao, Zhenting Wang, Chenlu Zhan, Hongwei Wang, and Yongfeng Zhang. Agent security bench (asb): Formalizing and benchmarking attacks and defenses in llm-based agents, 2024. URL https://arxiv.org/abs/2410.02644.

[241] Lingxi Zhang, Guangtao Zheng, and Hanjie Chen. When embedding-based defenses fail: Rethinking safety in LLM-based multi-agent systems, 2026. URL https://arxiv.org/abs/2605.01133.

[242] Wenxiao Zhang, Xiangrui Kong, Conan Dewitt, Thomas Bräunl, and Jin B. Hong. A study on prompt injection attack against LLM-integrated mobile robotic systems. In *2024 IEEE 35th International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pages 361–368. IEEE, 2024. doi: 10.1109/issrew63542.2024.00103. URL https://doi.org/10.1109/issrew63542.2024.00103.

[243] Xiao Zhang, Yusheng Wang, Yuhao Fei, Dongyuan Li, Zian Liang, Liuyu Xiang, Hongxun Gu, and Zhaofeng He. Harnesssafe: Evaluating safety across persistent carriers in agent harnesses, 2026. URL https://arxiv.org/abs/2608.06984.

[244] Xing Zhang, Yanwei Cui, Guanghui Wang, Ziyuan Li, Wei Qiu, Bing Zhu, and Peiyang He. Library drift: Diagnosing and fixing a silent failure mode in self-evolving LLM skill libraries, 2026. URL https://arxiv.org/abs/2605.19576.

[245] Xinyu Zhang, Huiyu Xu, Zhongjie Ba, Zhibo Wang, Yuan Hong, Jian Liu, Zhan Qin, and Kui Ren. Privacyasst: Safeguarding user privacy in tool-using large language model agents. *IEEE Transactions on Dependable and Secure Computing*, 2024. doi: 10.1109/TDSC.2024.3372777. URL https://doi.org/10.1109/TDSC.2024.3372777.

[246] Xuanye Zhang, Yongsen Zheng, Zhuqin Xu, Kaiyu Zhou, Bowen Shen, Haoran Ou, Tianwei Zhang, and Kwok-Yan Lam. Memmorph: Tool hijacking in llm agents via memory poisoning, 2026. URL https://arxiv.org/abs/2605.26154.

[247] Yao Zhang, Zijian Ma, Yunpu Ma, Zhen Han, Yu Wu, and Volker Tresp. Webpilot: A versatile and autonomous multi-agent system for web task execution with strategic exploration. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025. doi: 10.1609/aaai.v39i22.34505. URL https://doi.org/10.1609/aaai.v39i22.34505.

[248] Yechao Zhang, Shiqian Zhao, Jiawen Zhang, Jie Zhang, Gelei Deng, Xiaogeng Liu, Chaowei Xiao, and Tianwei Zhang. When claws remember but do not tell: Stealthy memory injection in persistent personal agents, 2026. URL https://arxiv.org/abs/2607.05189.

[249] Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model-based agents. *ACM Transactions on Information Systems*, 43(6):1–47, 2025. doi: 10.1145/3748302. URL https://doi.org/10.1145/3748302.

[250] Zhexin Zhang, Shiyao Cui, Yida Lu, Jingzhuo Zhou, Junxiao Yang, Hongning Wang, and Minlie Huang. Agent-safetybench: Evaluating the safety of llm agents, 2024. URL https://arxiv.org/abs/2412.14470.

[251] Lei Zhao, Abhay Bhaskar, and Edgar Dobriban. LivePI: More realistic benchmarking of agents against indirect prompt injection, 2026. URL https://arxiv.org/abs/2605.17986.

[252] Weixiang Zhao, Yingshuo Wang, Yichen Zhang, Yang Deng, Yanyan Zhao, Wanxiang Che, Bing Qin, and Ting Liu. Large language model agents are not always faithful self-evolvers, 2026. URL https://arxiv.org/abs/2601.22436.

[253] Weixiang Zhao, Yichen Zhang, Yingshuo Wang, Yang Deng, Yanyan Zhao, Xuda Zhi, Yongbo Huang, HaoHe, Wanxiang Che, Bing Qin, and Ting Liu. On safety risks in experience-driven self-evolving agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 42145–42169. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.2091. URL https://aclanthology.org/2026.findings-acl.2091/.

[254] Yujie Zhao, Boqin Yuan, Junbo Huang, Haocheng Yuan, Zhongming Yu, Haozhou Xu, Lanxiang Hu, Abhilash Shankarampeta, Zimeng Huang, Wentao Ni, Yuandong Tian, and Jishen Zhao. AMA-bench: Evaluating long-horizon memory for agentic applications, 2026. URL https://arxiv.org/abs/2602.22769.

[255] Yihe Zhou, Tao Ni, Wei-Bin Lee, and Qingchuan Zhao. A survey on backdoor threats in large language models (LLMs): Attacks, defenses, and evaluations. *Transactions on Artificial Intelligence*, page 3, 2025. doi: 10.53941/tai.2025.100003. URL https://www.sciltp.com/journals/tai/articles/2505000595.

[256] Zhenhong Zhou, Yuanhe Zhang, Hongwei Cai, Moayad Aloqaily, Ouns Bouachir, Linsey Pang, Prakhar Mehrotra, Kun Wang, and Qingsong Wen. MCPShield: A security cognition layer for adaptive trust calibration in model context protocol agents, 2026. URL https://arxiv.org/abs/2602.14281.

[257] Kunlun Zhu, Hongyi Du, Zhaochen Hong, Xiaocheng Yang, Shu Qing Guo, Zhe Wang, Zhihua Wang, Cheng Qian, Robert Tang, Heng Ji, and Jiaxuan You. Multiagentbench : Evaluating the collaboration and competition of LLM agents, 2025. URL https://doi.org/10.18653/v1/2025.acl-long.421.

[258] Yuxuan Zhu, Antony Kellermann, Akul Gupta, Philip Li, Richard Fang, Rohan Bindu, and Daniel Kang. Teams of LLM agents can exploit zero-day vulnerabilities. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23–35. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.eacl-long.2. URL https://aclanthology.org/2026.eacl-long.2/.

[259] Nan Zhuang, Boyu Cao, Yi Yang, Jing Xu, Mingda Xu, Yuxiao Wang, and Qi Liu. LLM agents can be choice-supportive biased evaluators: an empirical study. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 26436–26444, 2025. doi: 10.1609/aaai.v39i25.34843. URL https://doi.org/10.1609/aaai.v39i25.34843.