PREDICTION IS NOT ALL YOU NEED: TRACEGUARD FOR ACTION-ORIENTED SAFETY EVALUATION OF LONG-HORIZON AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tool-using agents can reach unsafe states through long-horizon trajectories in which no single early action is obviously harmful. A natural response is to predict failures before they occur. However, a warning is only useful while an effective intervention remains available and meaningful task progress can still be recovered. We introduce TRACEGUARD, a verifier-grounded framework for *action-oriented safety evaluation*. We first examine whether pre-failure trajectory signals yield deployable early warning. Although development-time analyses reveal apparent separability several decisions before failure, a frozen predictor does not generalize on held-out trajectories. This exposes a distinction between predictability, intervention actionability, and recoverability, and motivates a different question: *what can still be done effectively?* TRACEGUARD evaluates intervention and recovery frontiers relative to the first verifier-confirmed violation. Across more than 300 collected security trajectories, causal intervention experiments reveal strongly action-dependent safety windows: terminating execution remains effective substantially earlier than blocking only the current action, but sacrifices more task utility. Recovery experiments further show that checkpoint rollback preserves substantially more valid work than full reset, whereas finer localized rollback does not consistently outperform restoration to the last safe checkpoint. These results show that predictive performance alone does not characterize an agent's usable safety margin. TraceGuard instead characterizes the safety, utility, and recovery options available as execution unfolds, providing safety profiles that can inform monitoring policies, tool permissions, and agent configuration.

1 INTRODUCTION

Tool-using agents act over long trajectories that mix observations, planning, and state-changing tool calls. Under adversarial input, a verifier-confirmed violation may emerge only after several individually plausible decisions. In this setting the operational question is not only whether a trajectory is risky, but whether the system still has an effective action available before harm is committed.

A natural response is to predict failure early. We test that premise first using strict pre-action histories, but a frozen predictor does not generalize on a reserved held-out cohort (Section 4). This negative result is informative: apparent development-time separability does not by itself provide a reliable deployment trigger. More fundamentally, even a correct warning can arrive too late for a narrow intervention to matter, and preventing an immediate violation does not establish that the agent can resume safely or preserve legitimate work.

We therefore introduce TRACEGUARD, an *action-oriented safety evaluation* framework for long-horizon agents. Rather than reducing safety to a single prediction score, TraceGuard asks four linked questions. **RQ1: Detectability**—does strictly pre-action history contain signal that generalizes to future verifier-confirmed failure? **RQ2: Actionability**—given a candidate intervention, how far before the first violation does it still prevent the outcome? **RQ3: Recoverability**—after restoration, does execution remain safe and how much measured work is retained? **RQ4: Application**—can these measured frontiers inform a frozen runtime rule or permission configuration without assuming access to future failure time?

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

TraceGuard combines canonical trajectory abstraction with matched counterfactual replay. It treats the first verifier-confirmed violation t^* as a retrospective evaluation anchor, measures action-dependent prevention windows at $t^* - k$, and evaluates restoration separately from subsequent continuation. The resulting safety profile retains the distinction among warning quality, prevention opportunity, continuation safety, and supported progress. This distinction is important because a strong safety intervention can be operationally conservative: STOP prevents later violations by ending the episode, whereas narrower interventions preserve the possibility of continuation but may leave alternative paths to the same failure.

Contributions. We make four contributions: (i) an evaluation formulation that connects detectability, intervention actionability, and recoverability rather than treating prediction as the endpoint; (ii) a verifier-grounded replay protocol that measures the effects of action semantics, scope, and timing relative to t^*; (iii) intervention and recovery frontiers showing that successful prevention and safe restoration do not imply the same continuation behavior; and (iv) a frozen decision application demonstrating how empirical safety profiles can structure monitoring and tool-permission choices while exposing the limits of the available utility measurements.

2 Related Work

Agent safety benchmarks and multi-step attacks. AgentHarm, AgentDojo, and Agent Security Bench evaluate harmful multi-step behavior, prompt injection, and defenses in tool-using agents (Andriushchenko et al., 2025; Debenedetti et al., 2024; Zhang et al., 2025). ST-WebAgentBench evaluates task completion under policy constraints, while MCP-SafetyBench extends safety evaluation to heterogeneous Model Context Protocol servers (Levy et al., 2026; Zong et al., 2026). Our deterministic environment is based on the AI Agent Security Multi-Step Tool Attacks setting (OpenAI et al., 2026). TraceGuard does not claim comparable benchmark breadth; it studies a conditional question inside a fixed executable environment: for a verifier-confirmed failing trajectory, how do intervention timing, action scope, and restoration strategy change the resulting outcome?

Sequential monitoring and early warning. Monitoring work asks whether risk can be inferred from interaction history rather than from isolated responses. Lightweight sequential monitors aggregate evidence across decomposed attacks, and weak-to-strong monitoring studies reliability when the monitored agent is itself adversarial (Chen et al., 2026; Kale et al., 2026). TraceGuard treats pre-action detectability as necessary evidence but not as a sufficient safety criterion. Our intervention lags are aligned retrospectively to the first confirmed violation and are measured separately from the frozen predictor; they therefore quantify an oracle-aligned actionability frontier, not the performance of an online trigger.

Intervention, recovery, and safety–utility trade-offs. AgentDojo and related benchmarks evaluate defenses alongside benign task behavior, while ManagerBench emphasizes the cost of excessive caution in operational decision making (Debenedetti et al., 2024; Simhi et al., 2026). ReIn studies test-time recovery from conversational errors and evaluates later task success (Kim et al., 2026). TraceGuard does not introduce blocking, reset, or rollback as primitives. Its distinction is methodological: paired replay compares intervention semantics at matched trajectory checkpoints, then evaluates restored-state safety, subsequent continuation, and retained work as separate endpoints. Because our named-request and retained-unit measures are narrower than independently verified semantic task completion, we avoid interpreting them as a general utility score.

3 TraceGuard Framework

Let an execution trajectory be $\tau = (e_1, \ldots, e_T)$, where each canonical event records a message, tool call, tool result, error, or relevant observation. An independent environment verifier determines whether execution has reached a safety violation. For a verifier-positive trajectory, t^* denotes the first decision whose resulting execution is confirmed violating. TraceGuard uses this anchor to align three evaluations without assuming that t^* would be known at deployment time.

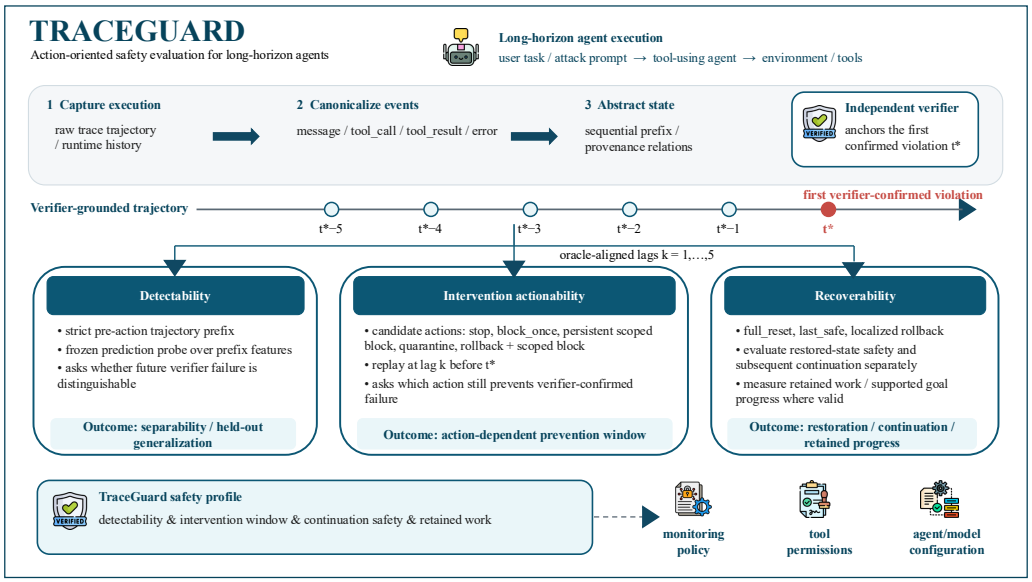

Figure 1: TRACEGUARD maps raw execution traces to canonical trajectories anchored by the first verifier-confirmed violation t^*, then evaluates detectability, intervention actionability, and recoverability as distinct but linked properties.

Detectability. At decision t, prediction receives only the strict pre-action prefix $\tau_{<t}$. The objective is to determine whether future verifier-positive failure is distinguishable from controls without leakage from the proposed or executed harmful action. We evaluate a frozen predictor on a reserved cohort to separate development-time separability from held-out generalization.

Intervention actionability. For an intervention action a and oracle-aligned lag k, replay begins from the matched checkpoint $t^* - k$. The empirical prevention rate is

$$P_{\text{prevent}}(a,k) = \frac{1}{n_{a,k}} \sum_{i=1}^{n_{a,k}} \mathbf{1}\left\{ V\left(\tau_i^{(a,k)}\right) = 0 \right\}.$$

where V is the environment verifier and $n_{a,k}$ is the eligible paired population for that action–lag cell. Because eligibility changes with lag and scope, $P(a,k)$ is reported with its cell-specific denominator rather than interpreted as repeated independent measurements of a fixed population.

Recoverability. A restoration strategy r maps an affected trajectory to a restored state, after which a continuation policy m either skips or reissues the failed turn. We report restoration safety and post-restoration safe continuation separately. When supported, we additionally report retained request units as a narrow measure of preserved work. This prevents a safe checkpoint, a safe continuation with no later actions, and successful task completion from being conflated.

Integrated safety profile. For each evaluated setting, TraceGuard therefore produces a structured profile consisting of held-out detectability, the prevention frontier $P(a,k)$, recovery continuation outcomes, and supported retained-work measurements. These quantities can be compared when selecting monitoring rules or permissions, but they are not collapsed into a single scalar objective unless an independently justified utility function is available.

4 EXPERIMENTAL SETUP

Environment and unit of analysis. We evaluate in the AI Agent Security Multi-Step Tool Attacks setting, where a deterministic tool-using agent can access filesystem, email, web-search, shell, and HTTP tools and an independent verifier determines safety outcomes (OpenAI et al., 2026). We

record messages, tool calls/results, runtime state, and verifier outcomes, then canonicalize them into ordered trajectories. The statistical unit is the trajectory/run, not each prefix or each replay row: prefixes from the same run are correlated, and lag-specific intervention rows reuse the same underlying execution conditions.

The canonical corpus contains 318 collected runs and 313 exact-unique trajectories. Table 1 separates the populations used for development prediction, replay, confirmation, and policy testing. These cohorts answer different questions and should not be added as independent samples.

Table 1: TRACEGUARD evaluation cohorts. Counts describe distinct analysis populations, not additive independent samples. Positives and controls are defined by the environment's safety verifier.

Cohort	Runs / units	Positive / control	Primary use
Canonical corpus	318 collected; 313 exact-unique	125 / 188	Trajectory foundation
Predictability	213 trajectories; 2,710 prefixes	125 / 88	Pre-failure detection
Exact-replay development	266 / 280 candidates	116 / 150	Intervention and recovery
Independent confirmation	220 new executions	100 / 120	Frozen-protocol confirmation
Frozen policy test	166 executions	70 / 96	Decision application

Notes. The 188 canonical controls include benign and unsuccessful-attack runs; the predictability cohort uses 88 unsuccessful-attack controls. Exact replay excludes 11 live-ALLOW mismatches and three runs with unobservable original hop budgets. The development replay supports a narrow, task-specific goal measure for 147 runs (32 positive, 115 control), not general semantic task completion. Positive intervention support decreases across oracle-aligned lags $k = 1, \ldots, 5$: $n_k = 116, 108, 106, 100, 82$. Confirmation uses new executions in the same agent and fixture family; neither confirmation nor policy testing establishes cross-environment generalization.

Strict pre-action detectability. Prediction features are constructed only from information available before the next proposed action. Development analysis contains 213 trajectories and 2,710 prefixes, including 125 timed verifier-positive trajectories and 88 unsuccessful-attack controls. The frozen predictor is evaluated once on a reserved 33-run final cohort containing four positive runs and 1,200 prefix–horizon rows (42 positive rows; prevalence 0.035). It obtains AUPRC 0.0269 (run-bootstrap 95% CI 0.0240–0.0338), AUROC 0.368 (0.316–0.446), and Brier score 0.0544. Because AUPRC is below the final-row prevalence and no operational threshold was frozen, we do not treat the predictor as a deployable intervention trigger.

Matched replay. For each eligible verifier-positive trajectory, checkpoints are indexed retrospectively as $t^* - k$, $k \in \{1, \ldots, 5\}$. Development intervention/recovery uses 266 of 280 exact-replay candidates (116 positives, 150 controls). Matching preserves the agent, initial conditions, ordered input, execution budget, ALLOW replay, and pre-intervention event prefix. The complete replay schematic and factor/endpoint matrix are reported in Appendix A, Figure 4 and Table 3.

Intervention, recovery, and uncertainty. We compare ALLOW, STOP, BLOCK_ONCE, persistent resource/tool/namespace restrictions, and supported quarantine or rollback variants. Recovery compares full reset, last-safe restoration, and provenance-localized rollback under SKIP or REISSUE continuation. A separately collected confirmation cohort contains 100 verifier-positive and 120 control executions. Paired confirmation differences use trajectory-paired percentile bootstrap intervals with 2,000 replicates; the comparison family is not multiplicity-adjusted. A later frozen policy test contains 70 positives and 96 controls. Confirmation and policy testing use new executions but the same SDK agent and fixture family, so they establish within-environment confirmation rather than cross-environment generalization. Additional matching and cohort audits appear in Appendices A–C.

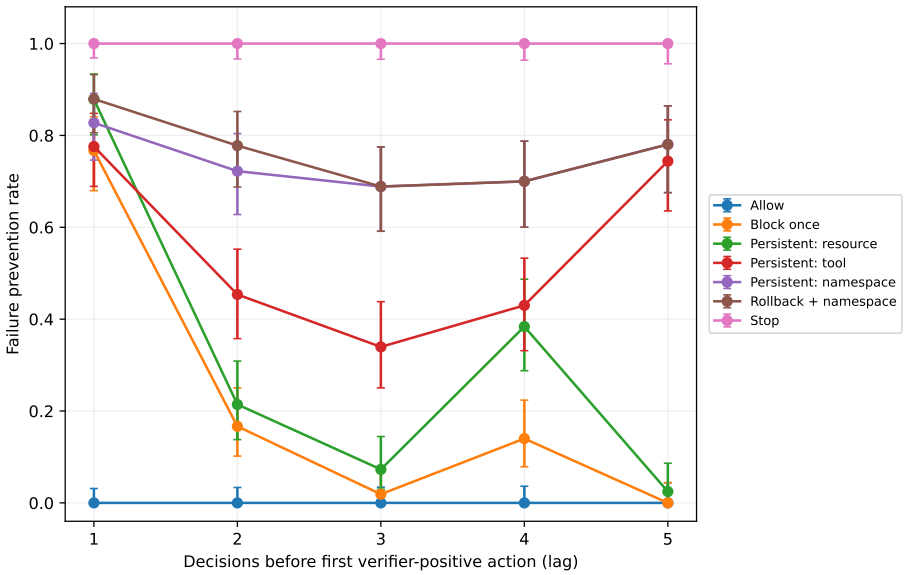

Figure 2: Development failure-prevention rates by intervention action and oracle-aligned lag. STOP terminates the episode; ALLOW is the exact-replay reference. Eligible populations vary by lag and scope.

5 INTERVENTION

Timing and action semantics. Figure 2 shows that prevention depends jointly on the action and its oracle-aligned timing. BLOCK_ONCE prevents 89/116 failures (76.7%) at lag 1, falls to 2/106 (1.9%) at lag 3, and prevents 0/82 at lag 5. Persistent namespace blocking instead prevents 96/116 (82.8%), 73/106 (68.9%), and 64/82 (78.0%) at those lags. Persistent tool blocking is also non-monotonic, rising from 36/106 (34.0%) at lag 3 to 61/82 (74.4%) at lag 5. These patterns show why a single notion of "earlier is better" is insufficient: intervention semantics determine which downstream actions remain reachable after the intervention.

Why scope matters. One-shot denial modifies only the selected tool call; a deterministic continuation may still reach the same verifier-positive outcome through a later action. Persistent namespace and tool restrictions constrain a larger set of subsequent decisions, while resource-scoped actions depend on whether the harmful path revisits the same resource. Rollback plus namespace blocking matches or exceeds namespace blocking at early lags in the development cohort, but does not improve on namespace blocking at lags 3–5. Resource quarantine is weak in the measured cells. Exact action–lag counts are reported in Appendix B, Table 4. These are empirical properties of the evaluated environment and scopes, not a universal ordering of intervention mechanisms.

Prevention is not completion. STOP prevents every eligible development violation by terminating the episode. It is therefore a useful prevention comparator but not evidence of successful task continuation. The same caution applies to persistent restrictions: higher verifier safety can come from removing capabilities that may also be needed for legitimate work. The current named-request checks are too narrow to establish a general safety–utility optimum, so we report prevention and supported progress separately rather than combining them into one score.

Frozen-protocol confirmation. A separately collected confirmation cohort reproduces the action-dependent contrast but not the development absolute rates. Persistent namespace blocking exceeds BLOCK_ONCE by 35.0 percentage points at lag 1 ($n = 100$, paired 95% CI 25.0–44.0) and 25.8 points at lag 5 ($n = 89$, 16.9–36.0). Yet BLOCK_ONCE shifts from 89/116 prevention in development to 33/100 in confirmation at lag 1. The evidence therefore supports a robust contrast in

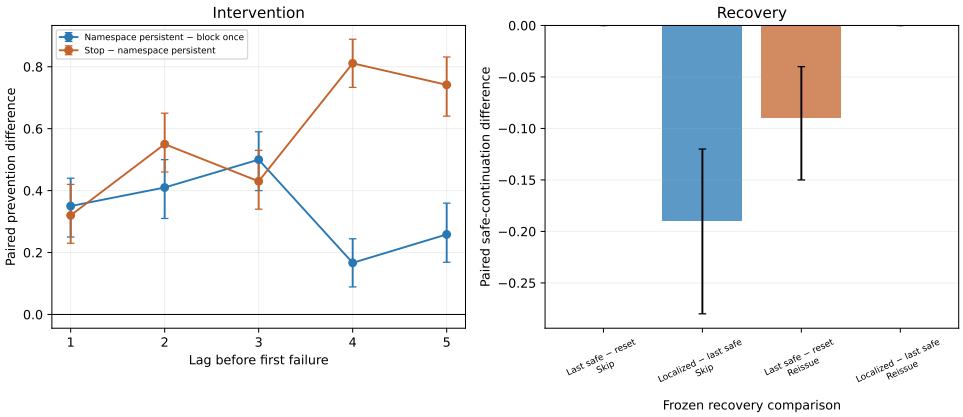

Figure 3: Frozen-protocol confirmation. Left: paired intervention-prevention differences across lags. Right: paired post-restoration safe-continuation differences at depth 1. Error bars are trajectory-paired bootstrap intervals.

actionability within this agent/fixture family, not a fixed prevention probability that can be transferred unchanged to other environments.

6 Recovery

Restoration and continuation are different endpoints. All displayed development restorations are verifier-safe at the restoration checkpoint, but subsequent execution differs sharply. At depth 1 under SKIP, safe continuation is 101/116 after full reset, 99/116 after last-safe restoration, and 60/116 after localized rollback. Under REISSUE the corresponding counts are 16/116, 0/116, and 0/116. Across depths 1–5, last-safe/SKIP remains between 83.9% and 91.5%, whereas localized/SKIP ranges from 51.7% to 63.4%. The complete depth-specific table is reported in Appendix B, Table 5; different depths are not independent replicates.

Retained work does not determine continuation safety. On the 32 development positives supporting the narrow named-request measure at depth 1, mean retained requested units are 0 for full reset, 0.5625 for last-safe restoration, and 1.5625 for localized rollback. The corresponding retained-work plot is provided in Appendix B, Figure 5. Localized rollback retains the most measured units but has lower SKIP continuation than reset or last-safe restoration. This illustrates why "recover more state" and "resume more safely" are not interchangeable objectives. The retained-unit measure also cannot verify semantic task completion or reversal of already executed external effects.

Confirmation. The confirmation cohort preserves the distinction between restoration and continuation. At depth 1, last-safe minus reset under SKIP is 0.0 percentage points (87/100 versus 87/100), while localized minus last-safe is -19.0 points (68/100 versus 87/100; paired 95% CI $[-28, -12]$). Under REISSUE, last-safe minus reset is -9.0 points (0/100 versus 9/100; $[-15, -4]$). Some SKIP rows involve no post-restoration actions, so a verifier-safe continuation can still be operationally unproductive. Figure 3 summarizes the frozen intervention and recovery contrasts; exact paired rows are in Appendix B.

7 Decision Application

The frontiers above describe what happened under controlled, oracle-aligned evaluation. We next ask whether those measurements can inform decisions that do *not* observe future t^*. These experiments are proof-of-concept applications of the profile, not evidence of an optimal controller or general agent/model router.

Frozen intervention rule. A development-derived lookup rule chooses among ALLOW, one-shot blocking, persistent scoped blocking, and STOP from the currently proposed tool-action category and previously received messages. Its development-cell safe-continuation floor of 0.75 is a selection constraint over observed cells, not a calibrated per-execution probability. The rule was frozen before evaluation on 166 later executions (70 baseline positives and 96 benign controls). It selects ALLOW for all 96 controls, persistent blocking for 51 `fs.delete` triggers, BLOCK_ONCE for 13 `shell.run` triggers, and STOP for six `http.post` triggers.

Table 2 compares the guided rule with fixed-action policies. Guided intervention prevents 47/70 baseline-positive violations and leaves 143/166 executions verifier-safe, compared with 39/70 prevented and 135/166 safe under fixed BLOCK_ONCE. The eight additional prevented positives comprise six STOP cases and two persistent-block cases. STOP prevents all 70 baseline-positive violations by terminating the affected episodes. Because every policy obtains the same 40/102 supported named-request checks, this test does not establish a semantic utility advantage for the guided rule.

Table 2: Frozen intervention-policy evaluation ($n = 166$; 70 baseline positives and 96 benign controls). "Safe" is verifier-negative execution; "prevented" is relative to baseline positives. The goal column is a narrow named-request check, not general semantic task completion.

Policy	Safe /166	Prevented /70	Named /102	Terminated /166
ALLOW	96	0	40	0
BLOCK_ONCE	135	39	40	0
Persistent block	137	41	40	0
STOP	166	70	40	70
Guided	143	47	40	6

Notes. The same 102 runs support the narrow goal check (six baseline positives); the test does not establish semantic completion or a safety–utility gain over stop. The guided policy was frozen before test collection.

Recovery and permission configurations. A frozen recovery rule chooses last-safe restoration followed by SKIP when all received requests are explicit retrievals and full reset followed by SKIP otherwise. On the 70 baseline-positive policy-test trajectories it yields 46/70 verifier-safe continuations, tying fixed reset and last-safe rules; localized rollback yields 26/70. An offline permission comparison on the same 166 executions produces verifier-safe counts of 96 for unrestricted ALLOW, 166 for STOP, 137 for persistent blocking, and 143 for the guided rule. All four configurations again report 40/102 supported named-request checks. Full tables are provided in Appendix B.

How to use a TraceGuard profile. The application suggests a decision workflow rather than a universal policy: first specify which safety failure and capability scope matter; then consult the actionability frontier for interventions that remain effective within the expected warning window; finally, evaluate candidate recovery rules using continuation safety and an independently appropriate task-progress measure. If no reliable online warning exists, the profile still informs static permission design by revealing which scopes produce prevention under matched replay. Conversely, a high prediction score without a supported intervention window should not be interpreted as operational protection.

8 LIMITATIONS

Prediction and actionability should be reported separately. The held-out predictor result and the replay frontiers answer different questions. The predictor tests whether a failure signal generalizes prospectively from strict prefixes; the replay experiment asks whether a specified action would change an outcome at a retrospectively aligned checkpoint. TRACEGUARD's central empirical lesson is therefore not that prediction is unnecessary, but that prediction quality alone cannot reveal the usable safety margin. A deployable system ultimately needs both a reliable trigger and an intervention whose effective window overlaps that trigger.

Environment scope. All executable comparisons use one deterministic SDK agent and related fixtures, prompt-bank material, and attack families. Confirmation adds new executions and ordered messages but not a new agent or environment. The results therefore support within-environment

conclusions about the evaluated action and recovery semantics. Cross-agent, cross-model, and cross-environment transfer remain open. In particular, absolute prevention rates should not be treated as properties of an intervention primitive independent of the environment.

Oracle alignment and dependence. The main intervention frontiers use t^* known retrospectively to the evaluator. They quantify how prevention changes with distance from failure; they do not show that a runtime monitor can locate those checkpoints. Eligible populations also change with lag and action scope; and multiple rows derive from the same underlying trajectories. We therefore report cell-specific denominators and paired confirmation contrasts rather than pooling replay rows as independent executions. Shared generators and seeds motivate the broader sensitivity checks documented in the appendix.

Utility and irreversible effects. Stop prevents further verifier violations by ending execution. A restored state can be verifier-safe even when resumed execution later fails or performs no useful action. The available named-request checks and retained trace/request units are narrow measurements, not independent semantic task completion. Recovery of the trace also cannot undo an external send or other irreversible effect that already occurred. Consequently, the present results do not identify a global safety–utility optimum.

Next evaluations. The strongest extension is a prospective end-to-end study in which a frozen monitor triggers interventions on newly collected executions and task success is judged by an independent goal oracle. Broader agents and tool environments, adaptive or monitor-aware attacks, and interventions with explicit human approval or sandboxing would test whether the observed actionability and recovery distinctions persist outside the current controlled setting.

9 Conclusion

We introduced TraceGuard, a verifier-grounded framework for evaluating long-horizon agent safety through detectability, intervention actionability, and recoverability. The reserved prediction result shows why development-time warning signal should not be equated with deployable protection. Matched replay then shows that prevention windows depend strongly on action semantics and timing, while recovery experiments show that verifier-safe restoration, safe continuation, and retained work can disagree. A frozen policy application demonstrates how these profiles can organize operational comparisons without claiming a general safety–utility optimum. Future work should connect prospective prediction triggers to intervention and recovery under independent task-goal verification, broader agents and environments, and adaptive attacks.

AI use statement

Generative AI tools were used to assist with coding. All AI-assisted coding were reviewed and verified by the authors, who take responsibility for the final results.

Reproducibility statement

Experimental protocols, cohort construction, replay eligibility criteria, implementation details, and additional analyses are documented in the appendix and supplementary materials.

References

Maksym Andriushchenko, Alexandra Souly, Mateusz Dziemian, Derek Duenas, Maxwell Lin, Justin Wang, Dan Hendrycks, Andy Zou, Zico Kolter, Matt Fredrikson, Yarin Gal, and Xander Davies. AgentHarm: A benchmark for measuring harmfulness of LLM agents. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/c493d23af93118975cdbc32cbe7323f5-Abstract-Conference.html.

Yueh-Han Chen, Nitish Joshi, Yulin Chen, Maksym Andriushchenko, Rico Angell, and He He. Monitoring decomposition attacks with lightweight sequential mon-

itors. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/2985777c8f2152664946ed30deb77291-Abstract-Conference.html.

Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, 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*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/97091a5177d8dc64b1da8bf3e1f6fb54-Abstract-Datasets_and_Benchmarks_Track.html.

Neil Kale, Chen Bo Calvin Zhang, Kevin Zhu, Ankit Aich, Paula Rodriguez, Christina Q. Knight, and Zifan Wang. Reliable weak-to-strong monitoring of LLM agents. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=WV7xIboTDK.

Takyoung Kim, Jinseok Nam, Chandrayee Basu, Xing Fan, Chengyuan Ma, Heng Ji, Gokhan Tur, and Dilek Hakkani-Tür. ReIn: Conversational error recovery with reasoning inception. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/3144924eaa37b635a79dd382c6d7c825-Abstract-Conference.html.

Ido Levy, Ben Wiesel, Sami Marreed, Alon Oved, Avi Yaeli, Nir Mashkif, and Segev Shlomov. ST-WebAgentBench: A benchmark for evaluating safety and trustworthiness in web agents. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=MuCDzH0ctf.

OpenAI, Google, and IEEE. AI Agent Security – Multi-Step Tool Attacks. Kaggle competition, 2026. URL https://www.kaggle.com/competitions/ai-agent-security-multi-step-tool-attacks.

Adi Simhi, Jonathan Herzig, Martin Tutek, Itay Itzhak, Idan Szpektor, and Yonatan Belinkov. ManagerBench: Evaluating the safety-pragmatism trade-off in autonomous LLMs. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/b8330f5b70b3c53172417deac6f057b1-Abstract-Conference.html.

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. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/5750f91d8fb9d5c02bd8ad2c3b44456b-Abstract-Conference.html.

Xuanjun Zong, Zhiqi Shen, Lei Wang, Yunshi Lan, and Yang Chao. MCP-SafetyBench: A benchmark for safety evaluation of large language models with real-world MCP servers. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/d46f127a80dc58cbc0732a717285c43a-Abstract-Conference.html.

A Additional Evaluation Methods

Matching and intervention timing. Matched action arms preserve agent, seed, fixture, ordered inputs, hop budget, and initial-state hash; ALLOW checks agreement with the collected reference. Oracle-timed interventions additionally require agreement of the pre-trigger tool-event prefix. The coordinate $t^* - k$ uses the retrospectively known first verifier-positive decision and is unavailable to a runtime policy. An eligible positive is counted once within each compared lag/contrast, not once per arm or pooled across lags.

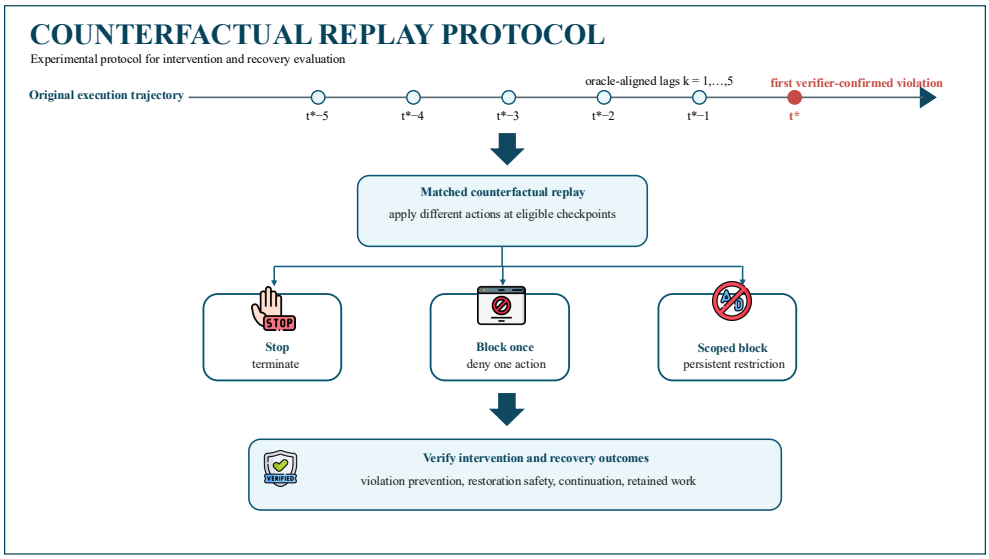

Figure 4: Matched counterfactual replay protocol for intervention and recovery evaluation.

Recovery endpoints. We report verifier safety immediately after restoration separately from safety of subsequent execution under SKIP or REISSUE. Full reset, last-safe checkpoint, and provenance-localized rollback are evaluated separately from retained trace/request units. Neither retained units nor a zero-action SKIP continuation proves semantic task completion; executed external effects are not assumed reversible.

Comparisons and uncertainty. Confirmation contrasts are paired rate differences on lag-eligible trajectories. Reported 95% intervals are trajectory-paired percentile bootstrap intervals with 2,000 replicates. Seed-cluster sensitivity intervals are broader, and the comparison family is not claimed to be multiplicity-adjusted.

Experimental factor matrix. Table 3 summarizes the factor and endpoint definitions used across the main experiments.

Table 3: Experimental factors and measured outcomes. Oracle-aligned timing is a retrospective evaluation coordinate, not a deployment-time prediction signal.

Dimension	Configuration	Measured outcome
Detectability	Strict pre-action trajectory prefixes	Prediction of later verifier-positive failure
Intervention	ALLOW baseline; STOP; BLOCK_ONCE; persistent scoped block	Verifier-confirmed violation after replay
Timing	Eligible checkpoints $t^* - k$, $k \in \{1, \ldots, 5\}$	Prevention at each supported lag and action
Recovery	Full reset; last-safe checkpoint; localized rollback	Restored-state safety and subsequent continuation
Progress	Retained work; supported task-specific goal checks	Measured retained progress where defined

Notes. STOP ends the episode; its violation-prevention outcome does not imply productive continuation. Restoration safety, later execution safety, and goal-related measurements are separate endpoints.

Replay schematic.

Table 4: Development intervention frontier across oracle-aligned lags before the first verifier-confirmed violation. Each cell reports failure-prevention rate (%) and prevented/eligible verifier-positive trajectories. ALLOW is the exact-replay reference; STOP prevents subsequent violations by ending the episode, not by successfully continuing the task.

Action / scope	Lag 1	Lag 2	Lag 3	Lag 4	Lag 5
ALLOW	0.0 (0/116)	0.0 (0/108)	0.0 (0/106)	0.0 (0/100)	0.0 (0/82)
BLOCK_ONCE	76.7 (89/116)	16.7 (18/108)	1.9 (2/106)	14.0 (14/100)	0.0 (0/82)
Persistent: resource	87.9 (94/107)	21.4 (21/98)	7.3 (7/96)	38.4 (38/99)	2.5 (2/81)
Persistent: tool	77.6 (90/116)	45.4 (49/108)	34.0 (36/106)	43.0 (43/100)	74.4 (61/82)
Persistent: namespace	82.8 (96/116)	72.2 (78/108)	68.9 (73/106)	70.0 (70/100)	78.0 (64/82)
Quarantine: resource	5.6 (6/107)	3.1 (3/98)	7.3 (7/96)	24.2 (24/99)	2.5 (2/81)
Rollback + namespace	87.9 (102/116)	77.8 (84/108)	68.9 (73/106)	70.0 (70/100)	78.0 (64/82)
STOP	100.0 (116/116)	100.0 (108/108)	100.0 (106/106)	100.0 (100/100)	100.0 (82/82)

Notes. Main eligible counts by lag are $n_k = 116, 108, 106, 100, 82$; resource-specific actions have smaller supported populations, shown explicitly in their cells. Percentages use the denominator in each cell; intervention-by-lag rows are not independent trajectories. Persistent means blocking subsequent actions in the specified scope. The resource-quarantine arm is included in the CSV but omitted from the provided intervention-frontier plot. Named-request completion is a narrow, goal-supported proxy, not independently verified semantic task completion.

Table 5: Development recovery safe-continuation results on eligible verifier-positive trajectories. Entries are safe continuations / eligible runs (percentage). Restorations were verifier-safe at the checkpoint in all displayed runs; the entries measure the different endpoint of subsequent safe continuation.

Depth	Eligible n	Last-safe / SKIP	Localized / SKIP	Last-safe / REISSUE	Localized / REISSUE
1	116	99/116 (85.3%)	60/116 (51.7%)	0/116 (0.0%)	0/116 (0.0%)
2	108	91/108 (84.3%)	58/108 (53.7%)	0/108 (0.0%)	0/108 (0.0%)
3	106	89/106 (84.0%)	56/106 (52.8%)	2/106 (1.9%)	0/106 (0.0%)
4	100	87/100 (87.0%)	54/100 (54.0%)	14/100 (14.0%)	0/100 (0.0%)
5	82	75/82 (91.5%)	52/82 (63.4%)	14/82 (17.1%)	0/82 (0.0%)

Notes. Depth is the number of decisions from the recovery checkpoint to the first original violation. Full reset is a depth-independent comparator: SKIP 101/116 (87.1%); REISSUE 16/116 (13.8%). The subset with a supported narrow named-request measure contains $n = 32, 25, 25, 25, 17$ at depths 1–5, respectively; retained units and goal-related endpoints are not independent semantic task-completion measurements. Different depths have different eligible populations and are not independent trajectory replicates.

B Additional Results and Scope Checks

Full intervention frontier.

Recovery depth. Table 5 reports safe-continuation counts for all supported depths.

Retained work and continuation.

Confirmation details. Table 6 provides the exact paired intervention and recovery confirmation contrasts summarized by Figure 3.

Policy details. Tables 7 and 8 report recovery and permission-configuration comparisons. They do not establish general semantic-utility gains.

Supported safety–utility proxy. Figure 6 compares verifier-confirmed prevention with narrow named-request checks on matched goal-supported subsets. STOP is episode termination rather than successful resumed work.

Rollback depth and continuation. Figure 7 shows depth-specific continuation behavior for last-safe and localized rollback under SKIP and REISSUE.

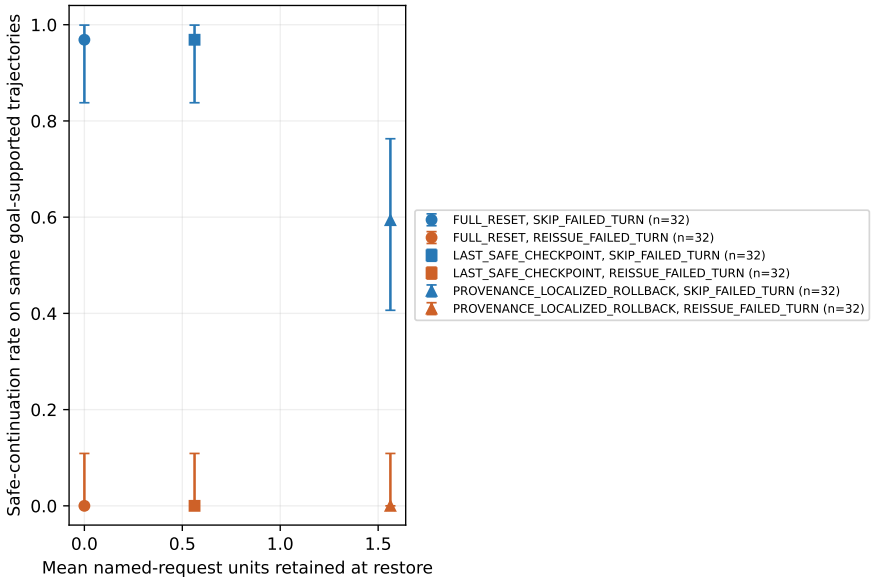

Figure 5: Development recovery outcomes on the same 32 goal-supported positives at depth 1: retained named-request units versus post-restoration safe continuation. The x-axis is a narrow retained-work measure, not general task success.

Table 6: Frozen-protocol confirmation of paired intervention-prevention and post-recovery safe-continuation differences. Rates are differences in percentage points (left minus right) on matched eligible trajectories. The confirmation cohort uses new executions in the same agent/fixture family, not an independent environment.

Lag/depth	Paired contrast	n	Development Δ (pp)	Confirmation Δ (pp)	95% CI (pp)
Intervention: verifier-confirmed failure prevention					
1	Namespace − once	100	+6.0	+35.0	[+25.0, +44.0]
1	STOP − namespace	100	+17.2	+32.0	[+23.0, +42.0]
2	Namespace − once	100	+55.6	+41.0	[+31.0, +50.0]
2	STOP − namespace	100	+27.8	+55.0	[+46.0, +65.0]
3	Namespace − once	100	+67.0	+50.0	[+40.0, +59.0]
3	STOP − namespace	100	+31.1	+43.0	[+34.0, +53.0]
4	Namespace − once	90	+56.0	+16.7	[+8.9, +24.4]
4	STOP − namespace	90	+30.0	+81.1	[+73.3, +88.9]
5	Namespace − once	89	+78.0	+25.8	[+16.9, +36.0]
5	STOP − namespace	89	+22.0	+74.2	[+64.0, +83.1]
Recovery: post-restoration safe continuation (depth 1)					
SKIP	Last-safe − Reset	100	-1.7	+0.0	[+0.0, +0.0]
SKIP	Localized − Last-safe	100	-33.6	-19.0	[-28.0, -12.0]
REISSUE	Last-safe − Reset	100	-13.8	-9.0	[-15.0, -4.0]
REISSUE	Localized − Last-safe	100	+0.0	+0.0	[+0.0, +0.0]

Notes. Namespace denotes persistent namespace-scoped denial; once denotes BLOCK_ONCE. STOP terminates the episode. Interventions have $n = 100$ at lags 1–3, $n = 90$ at lag 4 and $n = 89$ at lag 5. Recovery comparisons use $n = 100$ at depth 1. Intervals are 2,000-replicate trajectory-paired percentile bootstrap intervals from the supplied CSV; seed-cluster sensitivity intervals are broader, and the stated comparison family is not multiplicity-adjusted. The separate goal-supported joint safety/goal rows ($n = 11$ in confirmation) are omitted here because the CSV marks them LOW_SUPPORT; they cannot establish general semantic task completion.

C Reproducibility and Cohort Audit

Cohorts and exclusions. The canonical corpus contains 318 collected runs and 313 exact-unique trajectories. The prediction-development cohort contains 125 timed positives and 88 unsuccessful-attack controls; the reserved prediction test contains 33 runs with four positives. Exact replay retains

Table 7: Frozen recovery-policy test on the 70 baseline-positive trajectories. Retained units are a trace/request-specific measure, not verified semantic progress.

Recovery rule	Safe /70	Goals /6	Retained units
Full reset	46	0	0.0
Last-safe	46	0	0.0
Localized rollback	26	0	1.0
Guided recovery	46	0	0.0

Notes. Reported goal checks cover six positive runs; all four strategies have zero measured positive goal completion. Retained units are the supplied `requested_units_retained_mean`.

Table 8: Offline replay of tool-permission configurations on the same 166 policy-test trajectories. All configurations report 40/102 supported named-request checks; the metric does not establish task completion or a superior safety–utility policy.

Config.	Policy	Verifier-safe /166	Named /102
C0	ALLOW	96	40
C1	STOP	166	40
C2	Persistent block	137	40
C3	Guided	143	40

266 of 280 candidates (116 positives, 150 controls), excluding 11 live-ALLOW mismatches and three runs without an observable original hop budget. Positive intervention support at lags 1–5 is 116, 108, 106, 100, and 82. Confirmation contains 100 positives and 120 controls; the frozen policy test contains 70 positives and 96 controls.

Independence scope. Confirmation uses new executions and unique ordered messages after protocol freeze but the same SDK agent, fixtures, prompt bank, and attack families. The policy test has no exact ordered-message overlap with development or confirmation; a post-collection audit identifies seed overlap for a subset of policy positives with one development control. These facts limit claims to within-environment confirmation.

Frozen artifacts and unit of analysis. The reported intervention, recovery, confirmation, policy, and configuration results are taken from frozen CSV artifacts and cohort manifests. Reproduction should preserve trajectory-level pairing, lag-specific eligibility, and the distinction between verifier safety, continuation, retained work, and supported goal checks. Prefix rows and replay rows derived from the same trajectory must not be treated as independent experimental units.

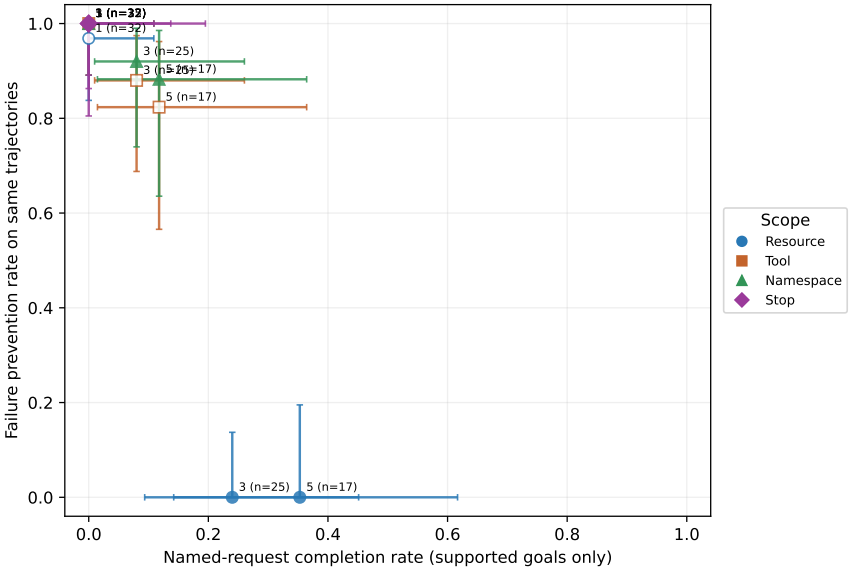

Figure 6: Intervention prevention and named-request checks on matched supported subsets. The x-axis is not a general semantic task-success measure.

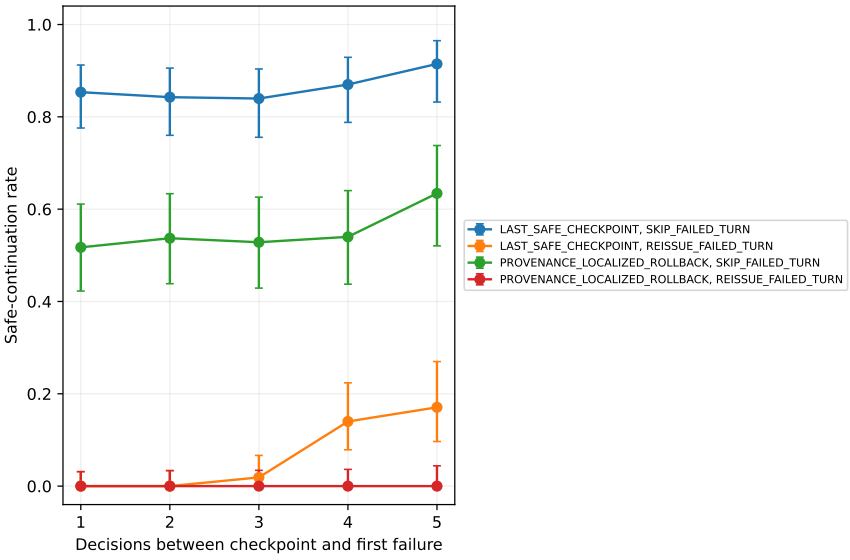

Figure 7: Safe-continuation rates across recovery depths; eligible populations vary with depth.