SAFETY DOES NOT COMPOSE: NON-DECAYING LOOP STATE FOR AUTONOMOUS LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model agents are increasingly deployed as *autonomous loops*. Starting from one human goal, such a system repeatedly discovers work, plans, executes tool calls, verifies outcomes and persists state across many unattended iterations. The agent safeguards in wide use, however, are defined over a *single trajectory*, and their safety state is re-initialized when the next trajectory begins. We show that this is a failure of composition rather than an implementation detail. Our central result establishes an observation-interface separation: jointly indistinguishable inner observations force equal detection and false-alarm rates regardless of monitor capacity, while distinguishing outer evidence enables perfect detection. We further show that the obvious repair of carrying a geometrically decaying risk score is insufficient, because the cooling-off period a patient adversary must wait is a constant that does not grow with the horizon N. We then present LoopHarness, which restores a persistent, *non-decaying* safety state at the loop level. Under mediated commits and an arbiter detection floor δ_M, it bounds the expected number of unauthorized irreversible actions by $B + m - 1 + m/\delta_M$, a constant in N, of which the $B + m - 1$ term is decided by a model-free rule and therefore survives a fully colluding verifier. On native Agent-SafetyBench tasks, LoopHarness reduces overall outer-state attack success from 88.4–97.6% for the evaluated baselines to 0.1% while retaining 96.9% clean target completion; matched ablations, a controlled retention study, and an adaptive white-box red team further test the mechanisms underlying this result.

1 INTRODUCTION

Background. Tool-using language agents have moved from answering a single request to running unattended. Given one human goal, the system *discovers* candidate work from open channels, *plans* and *executes* tool calls against real systems, *verifies* outcomes, *persists* state, and decides whether to continue (Yao et al., 2023; Shinn et al., 2023; Wang et al., 2023; Yang et al., 2024; Park et al., 2023). Two properties separate this regime from single-turn tool use. First, some actions are *irreversible*: a transfer executes, an email leaves the organization, a record is deleted. Second, the agent carries *persistent state* (a memory store, a task ledger, accumulated context) that survives into the next iteration and is itself an attack surface. The safety machinery that agents ship with was designed for neither property. Input sanitization, per-call classification, privilege ceilings, capability tokens and in-trajectory rollback are all scoped to one rollout, and in every such defense we are aware of, the internal risk state is re-initialized when that rollout ends (Inan et al., 2023; Rebedea et al., 2023; Shi et al., 2025).

Recent work and motivation. Recent defenses have made single trajectories substantially harder to subvert. Information-flow control and execution isolation confine untrusted content (Wu et al., 2024; 2025); privilege programming restricts what a compromised rollout can reach (Shi et al., 2025); and separating a planning channel from untrusted data blocks a large class of injections within one task (Debenedetti et al., 2025; Beurer-Kellner et al., 2025). The benchmarks that measure this progress are likewise organized one task per episode (Zhang et al., 2024; Debenedetti et al., 2024; Zhan et al., 2024; Ruan et al., 2024; Yuan et al., 2024; Andriushchenko et al., 2025). Attack research, by contrast, has already moved past the trajectory, since memory and knowledge-base poisoning plants content in one interaction that fires in a later one (Chen et al., 2024; Dong et al., 2025). The resulting asymmetry

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.

is structural. A defender's safety state is reset every trajectory by construction, while an adversary's progress accumulates in precisely the channels that the loop exists to preserve. One consequence is quantitative: if a guardrail's residual failures are independent across iterations, its per-iteration compromise probability $\varepsilon > 0$ is amplified by the loop to $1 - (1 - \varepsilon)^N$, so a defense reported as 95% effective on one task fails with probability above one half after fourteen iterations. This illustration assumes temporal independence, whereas our observation-interface separation in Section 3.1 does not. Related work is discussed in Appendix G.

Our work. We formalize autonomous loops with explicit safety state and introduce **LoopHarness**, an outer defense that composes with any single-trajectory harness. Its five persistent components, described in Section 3, keep grounded attack evidence, action impact and verifier indecision separate. Only grounded attack evidence contributes to the retained risk value. Only rule-decidable loop-structural flags latch its discount at one, preventing subsequent decay. Action impact and verifier indecision affect current-iteration routing but do not enter retained risk.

Contributions.

- **An observation-interface separation.** We prove that jointly indistinguishable inner observations force equal episode detection and false-alarm rates at any monitor capacity, while distinguishing outer evidence enables perfect detection (Proposition 2). We also prove that geometric risk decay admits a constant cooling-off period (Proposition 3).
- **LoopHarness.** A loop-level defense of five persistent components under a fixed phase order (Section 3), in which attack evidence, action impact and verifier indecision remain separate signals.
- **Guarantees with an explicit division of labor.** A horizon-independent bound $\mathbb{E}[U_{1:N}] \leq B + m - 1 + m/\delta_M$ (Theorem 5) whose deterministic term needs no model call and survives a fully colluding verifier (Proposition 6), together with a characterization of the residual class that δ_M governs. Proofs are in Appendix B.
- **An outer-state evaluation protocol.** Paired clean and attacked episodes on native Agent-SafetyBench tasks, with family-specific outer-state interventions and unchanged native task contracts, evaluator labels excluded from every runtime prompt, Track-A preservation tests, Track-B outer-state attacks, per-module ablations, and an adaptive white-box red team scored by the real controller (Section 4).

2 PRELIMINARIES

Autonomous agent loops. A *goal specification* is a tuple $\mathcal{G} = (g, \mathcal{A}, \Omega, \Pi)$ with instruction text g, a scope allowlist $\mathcal{A}$ over capabilities, action classes and resource prefixes, a set Ω of *completion anchors* (work items whose successful execution constitutes progress), and task contracts Π (per-item risk-clearance thresholds and completion limits). An *episode* of horizon N runs iterations $i = 1, \ldots, N$. At iteration i the loop observes a *discovery feed* F_i of work items, admits $W_i \subseteq F_i$, executes each admitted item as one *inner trajectory*, verifies the iteration, writes to a persistent memory log $\mathcal{M}_i$, and decides whether to continue. A *work item* carries an identifier, a source channel, a declared provenance, a task reference, free text and declared scope metadata, and an *inner harness* $\mathcal{H}$ maps it and its tool runtime to a trajectory τ, an impact score, structural flags, and the set $\mathrm{Irr}(\tau)$ of irreversible tool calls that executed successfully, which we call *commits*.

Authorization is a property of the operator, not of the harness. We posit a policy oracle $\mathcal{O}$, never observed by the system, that labels each commit *authorized* or *unauthorized*. The harness decides authorization by a deterministic proxy: the conjunction of four predicates recorded in the trace, namely a valid capability token, conformance with $\mathcal{A}$, provenance authority of the originating item, and coverage by $\mathcal{G}$'s contract. An unauthorized commit that the proxy also rejects is *contract-visible*; one that the proxy accepts is *contract-invisible*. The distinction matters because only the first kind can be recognized without consulting a model. Let U_i count unauthorized commits at iteration i, split as $U_i = U_i^{\mathrm{vis}} + U_i^{\mathrm{inv}}$, and write $U_{1:N} = \sum_{i \leq N} U_i$.

Single-trajectory lifecycle defenses. Our inner harness is a four-layer lifecycle defense representative of current practice (Lin et al., 2026). The L1–L4 specifications are provided in Appendix D. One feature holds by construction and drives the rest of the paper.

Definition 1 (Per-trajectory reset)**.** An inner harness satisfies *per-trajectory reset* if its safety state (violation window, minimum tier, capability token, checkpoints) is re-initialized to a fixed θ_0 at the start of every trajectory, independently of all previous trajectories.

Reset is not a defect of one implementation. It is the property that makes a trajectory-scoped defense stateless and analyzable, and guardrail classifiers, per-call policies and isolation architectures all share it. It is also the property that an adversary operating across iterations exploits.

Threat model. Lin et al. (2026) evaluate six single-trajectory threats T1–T6 through attack families A1–A5, covering injection within one rollout, unsafe tool selection and privilege overreach. We retain that numbering, so the loop-level threats introduced by autonomy begin at **T7**. The adversary controls untrusted discovery channels and any tool output originating outside the organization, may write first-party content into memory through the loop's own summarization path, and knows the defense in full. It cannot forge the harness MAC key, modify the inner harness code, or reach the human operator. **T7 Discovery poisoning**: a malicious item enters the queue disguised as legitimate backlog, typically by self-declaring a trust level or signer its channel cannot vouch for. **T8 Cross-iteration grooming**: rejected work is resubmitted, possibly reworded, until admitted, or a payload is sharded so no single iteration carries anything unsafe and the capability materializes only once fragments accumulate. **T9 Self-feeding memory poisoning**: low-trust content is re-labelled as trusted policy without a signed authorization and later recalled as fact, including across a restart. **T10 Correlated-verifier evasion**: a stealthy payload also targets the verifier, so a checker sharing a model family or prompt with the writer rationalizes its claim. **T11 Runaway, halt evasion and goal drift**: scope expands incrementally, completion is repeatedly claimed to be imminent, or completed work is resubmitted, so irreversible actions accumulate while no progress is made and no stopping condition is reached.

Evaluation quantities. The primary quantity is the *compounding attack-success rate* $\mathrm{cASR}(n)$, the fraction of attack episodes compromised within the first n iterations, with normalized area $\mathrm{AUC} = \frac{1}{N}\sum_{n\le N}\mathrm{cASR}(n)$. Because a defense can trivially flatten cASR by refusing everything, we always pair it with clean-episode goal completion, written **CleanGC** for the paired target and **FullGC** for all slots, with the *per-iteration* ASR profile, and with the clean-episode *over-halt* rate. We also report under-halt, halt latency against the evaluator's ground-truth violation point, time-to-detect right-censored at N, mean risk-charged irreversible commits per attack episode as a conservative observable counterpart of U^{vis}, and advanced-verifier calls per episode.

3 LoopHarness

LoopHarness wraps an arbitrary inner harness with five persistent components under a fixed phase order, given as Algorithm 1 in Appendix D. Each iteration rehydrates and re-verifies persisted memory, admits work through IPM, gates the admitted items on goal conformance, executes them under the current capability ceiling with a governor budget test after each item, arbitrates the outcome with ASA, persists only an accepted outcome, folds one risk observation into XRC, and finally asks CRG whether to halt. The loop-level state $\sigma_i = (\Xi_i, c_i, a_i, \ell_i, \mathcal{M}_i, \mathcal{R}_i, \mathcal{K}_i)$ collects the retained risk cumulant, the risk-charged commit and advanced-verification counters, the capability-degradation level, the integrity-protected memory log and the rejection and authorization/lineage ledgers (Appendix A); none is re-initialized between iterations, and Ξ, the commit counters and the degradation level are bound into an authenticated checkpoint, so a memory compromise cannot silently reset them. Figure 1 summarizes this phase order and the persistent state carried across iterations.

Intake and provenance monitor (IPM). IPM is the only place new work enters the loop. It applies a deny-by-default scope check on every dimension the goal constrains, leaving unconstrained dimensions to L3 token mediation, and admits an item only if all six of the following hold: (i) its *declared* capabilities, action class and resource prefixes lie inside $\mathcal{A}$; (ii) its declared trust does not exceed what its channel can vouch for, and it asserts no signer that channel cannot verify; (iii) it is not a re-entry of rejected semantic work without an authorization issued *after* that rejection; (iv) any authorization it cites is *atomic*, one signed record covering both action class and destination, so permissions cannot be stitched from separate records; (v) any cross-episode lineage it claims is explicitly delegable; and (vi) on untrusted channels only, it carries no injection, escalation or scope-expansion pattern. At most k items are admitted per iteration. IPM additionally scores each

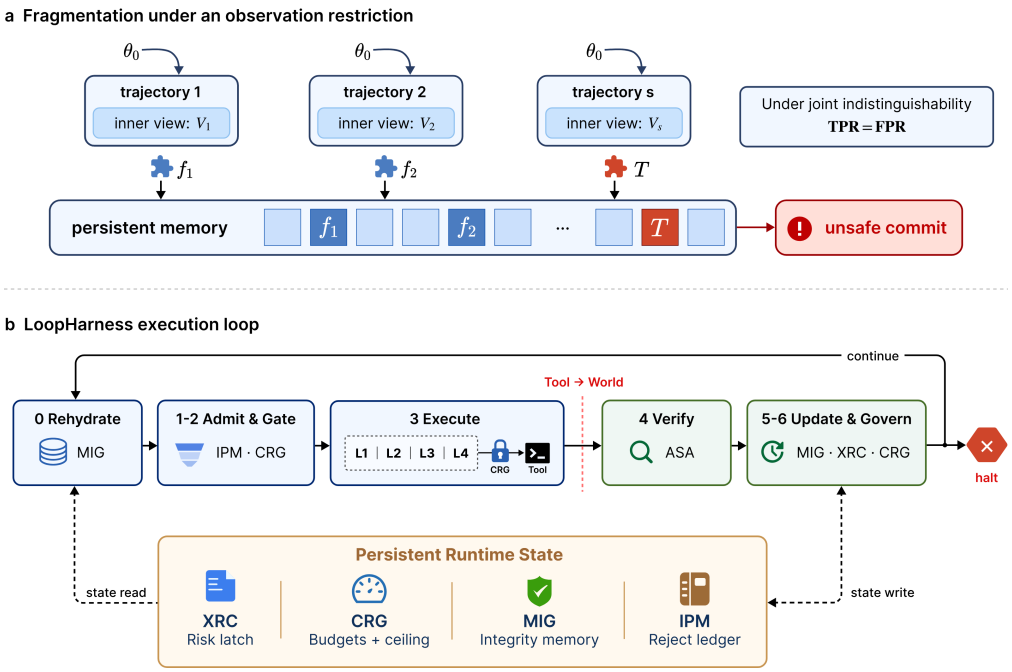

Figure 1: **LoopHarness overview. a, Fragmentation under an observation restriction.** Equal episode true-positive and false-positive rates follow when the complete inner observation sequences, including recalled memory, have the same joint law under attack and matched benign processes (Equation 2 and Proposition 2). Reset alone does not imply this condition. Perfect outer detection additionally requires distinguishing runtime evidence outside those observations. **b, LoopHarness execution loop.** MIG rehydrates and re-verifies state, and IPM and CRG gate work before execution. ASA arbitrates after the effect. Only accepted outcomes are persisted, XRC updates once per iteration, and CRG decides whether to continue or halt. Only a loop-structural flag latches the discount at one (Equation 1), preventing decay of Ξ. Loop-level safety state persists across iterations.

item with a continuous intake prior, of which only an untrusted item's value reaches XRC as attack risk. The prior is not a drop rule. The scoring rationale is detailed in Appendix D.

Cross-iteration risk cumulant (XRC). Every other component reads or writes Ξ: IPM and ASA contribute to r_i, CRG gates on it, and MIG binds it into the checkpoint. Exactly once per iteration, XRC folds that iteration's maximum grounded attack risk r_i and the union of its flags into

$$\Xi_i = \max\big(r_i,\ \gamma_i\,\Xi_{i-1}\big), \quad \gamma_i = \begin{cases} 1 & \text{if a structural flag has } \textit{ever} \text{ fired by iteration } i, \\ \gamma_\downarrow \in (0,1) & \text{otherwise.} \end{cases} \tag{1}$$

A *loop-structural flag* is a discrete, rule-decidable observation that one of the loop's own admission, integrity or authorization contracts was violated; Appendix D lists the full set. Only such a flag latches the discount, and a high continuous risk score alone never does. Setting the discount differently yields the three configurations compared in Section 4: off sets $\Xi_i = r_i$, which is per-iteration reset; decaying always uses $\gamma_\downarrow$, so risk is carried but never latches; and nondecaying is Equation 1. Without the latch, retention only delays the gate release; with it, waiting never releases the gate (Proposition 3). Applying the discount exactly once per iteration matters for the same reason, since a per-phase update would let an implementation detail accelerate forgetting. *Impact risk* and *review uncertainty* route current-iteration review but never enter r_i, Ξ, or the latch state. Impact risk can trigger expensive review even for a legitimate authorized deletion, which need not indicate compromise. Grounded contributions to r_i include an untrusted item's intake attack risk, a grounded risk observation, a confirmed causal diagnosis inside a trajectory, an authorization violation on an executed commit, a memory-integrity failure, or an arbiter detection. Without this stratification, ordinary high-impact work or verifier indecision could degrade later clean iterations.

Memory-integrity guard (MIG). Persistent memory is an append-only log in which entry j carries a chained tag $h_j = \mathrm{HMAC}_k(\langle j \rangle \| \langle e_j \rangle \| h_{j-1})$, in the manner of tamper-evident hash-chained records (Haber & Stornetta, 1991; Bellare et al., 1996). Periodic checkpoints authenticate a digest of the chain prefix *together with* its length and the governor counters (Ξ, c, a, ℓ), so an attempt to shorten the log or to rewrite a counter fails verification. On top of the log, MIG enforces three policies. A *monotone-trust write gate* lets trust be downgraded automatically but flags any upgrade lacking a signed authorization, storing the entry at the lower trust. A *rehydration* procedure, run at session start and at the top of every iteration, re-verifies the chain, audits per-key trust monotonicity so that an on-disk relabeling which bypassed the write gate is still caught, and returns the most recent checkpoint that still verifies. A *recall policy* re-feeds only a bounded suffix of trusted entries into the agent's context, excluding per-iteration summaries. Integrity checking alone would detect tampering without preventing recall; the third policy is what stops a downgraded entry from re-entering context, which is how the self-feeding channel of T9 is closed.

Adversarially robust stopping arbiter (ASA). ASA decides whether an iteration's outcome may be accepted and persisted, using three layers of evidence. A **deterministic gate**, computed from the trace with no model call, fires when a successful irreversible commit co-occurs with a corroborating structural flag or an authorization violation, or when the outer attestation contradicts executed-tool evidence. A **decorrelated checker** then issues at most one model call per iteration, routed to exactly one of a cheap or an advanced tier by a disjunction of impact, summed windowed risk, the divergence statistic, structural evidence, authorization uncertainty or violation, and inner-cascade review uncertainty; the advanced role must be a different model from the writer. Its verdict counts as a detection only if it saw an attempted action with arguments and authorization context and either prior attack evidence or a successful commit already existed, so a false positive on an iteration with neither cannot halt the loop. A **windowed divergence statistic** Δ_i, the fraction of the last W_{arb} iterations whose safety claim conflicts with the evidence recorded for them, may also trigger detection, but only above its threshold θ_Δ, with at least two independent divergences, and with grounded attack evidence already present. Any detection halts the loop in Phase 6, and only an accepted iteration is written to memory; acceptance is the conjunction of no detection, no requested review, goal conformance, no untokened irreversible commit, no scope violation and no blocked authority violation.

Compounding-risk governor (CRG). It enforces hard budgets $B \in \mathbb{Z}_{>0}$ for *risk-charged* commits and $B_{\mathrm{adv}} \in \mathbb{Z}_{>0}$ for advanced verification. Risk-charged commits are those rejected by the authorization proxy of Section 2 or associated with grounded attack evidence. Proxy-accepted commits without such evidence do not consume B. The capability ceiling is a step function of the current Ξ over the five privilege tiers, so it relaxes only if Ξ falls, which the latch prevents, and it constrains the iteration after the one that raised Ξ. CRG also enforces the goal's task contracts: a risk-clearance threshold gates an item on $\Xi < \theta_w$, and a completion limit bounds how many times an anchor $\omega \in \Omega$ may execute successfully. Reaching the risk threshold $\Xi_{\max}$, the commit budget B, or the advanced-verification budget B_{adv} forces a halt with escalation to the operator. The loop also force-halts when its no-progress commit counter reaches K and $\Xi \geq \Xi_{\max}/2$. A memory rollback restores log content only and never reloads the governor counters, so those never decrease within an episode.

3.1 GUARANTEES

Fix an attack episode of horizon N and an adaptive adversary that conditions on the whole history, and let $\mathcal{F}_i$ be the σ-algebra generated by everything observable up to iteration i. Recall the split $U_{1:N} = U^{\mathrm{vis}}_{1:N} + U^{\mathrm{inv}}_{1:N}$ of unauthorized commits into those the harness's proxy rejects and those it accepts. Proofs are in Appendix B.

Observation interface and fragmentation. Our first result concerns an explicit observation restriction, not reset alone. Let V_i collect the complete monitor-visible transcripts of the inner trajectories in iteration i, including the work items, goal, observations, action and authorization information, capability context, and all recalled memory actually supplied to the monitor.

A trajectory-scoped monitor retains no additional safety state between trajectories, but may use historical information present in its current input. Its aggregate alarm in iteration i is a measurable function $A_i = M_i(V_i)$. After termination, use a null view with no alarm.

Fix $N \geq s \geq 2$ and matched attack and benign episode laws $\Pr_{\mathrm{a}}$ and $\Pr_{\mathrm{b}}$ for the same declared goal and inner monitoring policy. Call the pair *joint-view s-fragmented* if the attack assembles an unauthorized capability from fragments carried on s iterations, with the capability appearing only after all fragments are jointly present in persistent state, and

$$\mathcal{L}_{\mathrm{a}}(V_1, \ldots, V_N) = \mathcal{L}_{\mathrm{b}}(V_1, \ldots, V_N). \tag{2}$$

This is a full-episode condition, including any separate trigger iteration; it is an assumption on the complete observation interface, not a consequence of reset or of individually benign-looking messages. No temporal independence or stationarity is assumed.

For a randomized monitor with random tape R, require equality of the joint laws of $(V_1, \ldots, V_N, R)$ under the two episode laws, and write $A_i = M_i(V_i, R)$.

Proposition 2 (Fragmentation separation under an observation restriction)**.** *For any joint-view s-fragmented pair and monitor satisfying the observation conditions above,*

$$\Pr_{\mathrm{a}}(A_i = 1) = \Pr_{\mathrm{b}}(A_i = 1), \qquad i = 1, \ldots, N,$$

and

$$\mathrm{TPR}_N := \Pr_{\mathrm{a}}(\exists i \leq N : A_i = 1) = \Pr_{\mathrm{b}}(\exists i \leq N : A_i = 1) =: \mathrm{FPR}_N.$$

Hence achieving $\mathrm{TPR}_N \geq 1 - \beta$, for $0 \leq \beta \leq 1$, requires $\mathrm{FPR}_N \geq 1 - \beta$.

If, additionally, retained outer evidence S_i admits a known measurable check C such that

$$\Pr_{\mathrm{a}}(\exists i \leq N : C(\mathsf{S}_i) = 1) = 1, \qquad \Pr_{\mathrm{b}}(\exists i \leq N : C(\mathsf{S}_i) = 1) = 0,$$

then an outer monitor checking C achieves episode true-positive rate 1 and false-positive rate 0 on these laws.

The check must use runtime evidence, not the unobserved policy oracle $\mathcal{O}$ or evaluator labels.

Proposition 2 is an observation-interface separation: additional outer evidence can distinguish executions whose complete inner observations are jointly indistinguishable. Merely retaining those same observations cannot do so. The retained-evidence condition is an additional premise, not an automatic property of persistent memory. A7 illustrates cross-iteration payload assembly but is not asserted to satisfy Equation 2: recalled fragments may be visible at the trigger, and Track A retains in-trajectory evidence. A claim of prevention additionally requires the relevant outer check before the unauthorized commit.

Remark 8 in Appendix B records the unfragmented case: a uniform history-conditional compromise probability lower bound $\varepsilon > 0$ gives $\mathrm{cASR}(n) \geq 1 - (1 - \varepsilon)^n$ and $\Omega(N)$ expected unauthorized commits when all N iterations are attempted without a stopping rule. These bounds follow from conditional probabilities and do not require independence. Our second result concerns risk retention: once grounded risk evidence has been observed, geometric decay permits a patient adversary to reopen the risk gate after a constant cooling-off period.

Proposition 3 (Cooling-off defeats geometric decay)**.** *Let the cumulant carry with a fixed discount $\gamma \in (0, 1)$ and no latch, and let a control action be gated on $\Xi \geq \theta$, with $\theta > 0$. Suppose evidence of magnitude $\rho \in [\theta, \Xi_{\max}]$ arrives at iteration τ with $\gamma \Xi_{\tau-1} \leq \rho$, and that none follows. The gate reopens at iteration $\tau + k^\star$ with $k^\star = \lfloor \log(\rho/\theta) / \log(1/\gamma) \rfloor + 1$, a constant independent of N; under* XRC-off *it reopens at $k = 1$, so decay buys the defender $k^\star - 1$ iterations and nothing at all once $\rho < \theta/\gamma$. If the evidence at τ raises a loop-structural flag, then under Equation 1 $\Xi_i \geq \rho \geq \theta$ for every later modeled iteration. Passive waiting alone therefore never reopens the gate; an explicit clearance transition lies outside this recurrence and this proposition.*

The matched A14 study in Table 3 tests this threshold crossing while holding all other LoopHarness components fixed.

Corollary 4 (Bounded opportunity)**.** *Against an adversary that supplies such evidence once and then waits, a risk-gated item is admitted on $\Omega(N)$ of the remaining iterations under reset and under decay, and never again after τ under Equation 1.*

We now bound damage. Let $\pi_i = \Pr[\text{the unauthorized commits of iteration } i \text{ are contract-visible} \mid \mathcal{F}_{i-1}, U_i > 0]$.

Assumption 1 (Loop safety assumptions)**. (A1) Mediated, bounded commits.** Every irreversible call passes one accounted gate; the capability token bounds a trajectory to m_0 commits and intake admits at most k items per iteration, so at most $m = k\,m_0$ commits execute per iteration. The governor tests after each work item, with at most $m_0 \le m$ commits between consecutive tests. **(A2) Detection floor.** There is $\delta_M \in (0, 1]$ such that, for every i and history, an iteration carrying a contract-invisible unauthorized commit is flagged by the arbiter with probability at least δ_M, and any such flag halts the loop. **(A3) Monotone commit counter.** The risk-charged commit counter c is non-decreasing within an episode, and no adversary action, restart or memory rollback lowers it.

Theorem 5 (Horizon-independent bound on unauthorized commits)**.** *Under Assumption 1, for every N and every adaptive adversary,*

$$U^{\mathrm{vis}}_{1:N} \le B + m - 1 \ \textit{surely}, \qquad \mathbb{E}\big[U^{\mathrm{inv}}_{1:N}\big] \le \frac{m}{\delta_M},$$

and, for every integer $z \ge 0$,

$$\Pr\big[U^{\mathrm{inv}}_{1:N} \ge m(z{+}1)\big] \le (1 - \delta_M)^z.$$

Hence $\mathbb{E}[U_{1:N}] \le B + m - 1 + m/\delta_M$, which does not depend on N. Remark 10 gives a single-constant form mB/δ with $\delta = \inf_i(\pi_i + (1 - \pi_i)\delta_M)$.

Proposition 6 (Model-free component and its residual class)**.** *The authorization proxy is evaluated by rules over recorded trace fields, with no model output as an operand, so $U^{\mathrm{vis}}_{1:N} \le B + m - 1$ holds for every checker, including one that is the writer model under adversarial instruction; only the m/δ_M term depends on the checker. Moreover a contract-invisible commit is by definition token-mediated, conformant with $\mathcal{A}$, issued from an item carrying provenance authority, and covered by $\mathcal{G}$'s contract, so it lies inside the operator's* declared *policy. The residual class is therefore the gap between declared scope and operator intent, and tightening $\mathcal{A}$ or Π moves mass out of it, making π operator-controlled rather than adversary-controlled.*

Appendix B carries two supporting results. Proposition 12 bounds verification at one checker call per iteration, of which at most $\min(N, B_{\mathrm{adv}})$ reach the advanced tier. Lemma 13 establishes tamper evidence for authenticated persistent state and counters under MAC unforgeability and an append-only checkpoint sequence. Combined with the control-flow invariant that rollback does not reload governor counters, this supports the implementation-level realization of (A3).

What the theory does and does not establish. The two halves of Theorem 5 are bought by different components. The deterministic half needs only the governor's budget with (A1) and (A3); it involves no model call, no empirical quantity, and *no latch*. What non-decay buys is the opportunity side: by Proposition 3 and Corollary 4, XRC-off or XRC-decay lets a patient adversary reopen a risk-gated capability $\Omega(N)$ times, whereas passive waiting does not reopen it under the latched recurrence. This is an opportunity bound, not a recovery policy. Damage and opportunity are bounded by different mechanisms, and Section 4 separates them, with family A15 targeting the governor's exact-once completion contract and A14 targeting retention. The bound has two limits: it constrains an expectation rather than an absence, and it is a scaling statement rather than an empirical crossover claim. We do not estimate a deployment-wide $N^\star$, because it also depends on environment-specific compromise and detection rates. Assumption (A2) is the only empirical input in the analysis, and Proposition 6 limits how much of the guarantee it can carry away.

4 Experiments

This section describes the frozen protocol, configurations, attack contracts and reporting metrics used in the evaluation. Every reported quantity is defined in Section 2.

Benchmark and episode construction. We use 200 audited, benign, executable tool-use tasks from Agent-SafetyBench (Zhang et al., 2024), preserving their native instructions, environments and tool schemas. All tool calls execute in the benchmark's native simulator. Episodes schedule one task per iteration under one goal; this queue is not contested and therefore understates deployment-time admission difficulty. Track A (A6–A10) evaluates attacks with in-trajectory evidence and measures compounding across iterations; Track B (A11–A15) evaluates outer-state controller contracts. Track

Table 1: Outer-only attack-success rate by family and clean target completion. Lower ASR is safer, higher CleanGC is better. Denominators: 200 attack episodes per family, 1,000 per configuration overall, 1,000 clean targets. ASR is evaluated against each family's specified controller-contract violation.

Config	A11	A12	A13	A14	A15	Overall	CleanGC
B0	97.5	99.0	97.5	97.0	97.0	97.6	97.4
B1	97.5	99.0	97.5	97.0	97.0	97.6	97.3
B1.5	97.5	99.0	97.5	97.0	97.0	97.6	97.3
B2	85.5	93.5	83.0	89.0	91.0	88.4	87.8
B3	97.5	99.0	97.5	97.0	97.0	97.6	97.3
B4	**0.0**	**0.0**	**0.0**	**0.0**	**0.5**	**0.1**	96.9
A-noIPM	97.5	0.0	0.0	0.0	0.5	19.6	97.0
A-noMIG	0.0	96.0	0.0	0.0	0.5	19.3	96.9
A-noASA	0.0	0.0	97.5	0.0	0.5	19.6	97.0
A-noXRC	0.0	0.0	0.0	97.0	0.5	19.5	96.9
A-noGov	0.0	0.0	0.0	97.0	97.0	38.8	96.9
A-collude	0.0	0.0	0.0	0.0	0.5	0.1	92.9

B uses the full cohort at $N = 10$, with every task serving as a target once per family. Because the cohort spans both dev and ood partitions, Table 1 is a controller-contract evaluation rather than a held-out generalization estimate. Track A uses 97 held-out ood parent clusters at $N \in \{5, 10, 20\}$, yielding 485 attack and 97 clean episodes per configuration at each horizon. Hyper-parameters are selected on dev and frozen before evaluation. Each Track-B attack episode has a clean twin with the same native target, differing only in the outer record constituting the violation. Track-A clean episodes use the same parent skeletons with the injected items removed. Evaluator labels are used only for scoring and are never exposed to the agent, inner harness, arbiter or runtime controller. Screening, splits, episode construction and manifests are detailed in Appendix F.

Configurations. All configurations share one controller implementation and differ only in the settings reported in Table 8 in Appendix D. B0 runs neither the inner harness nor any outer component, whereas B1 adds the inner harness alone. B1.5 augments B1 with a decaying cumulant. B2 uses a naive arbiter that observes only the agent's final response, and B3 applies a post-hoc audit after the loop ends. B4 enables all five components. Each named removal configuration disables one designated component of B4; A-decayXRC replaces the latching cumulant with geometric decay, and A-collude routes the checker to the writer role.

Attack suites. We evaluate attack families A6–A15. A1–A5 are inherited from SafeHarness and were not rerun. Table 9 in Appendix E lists the full taxonomy. **Track A**, covering A6–A10, retains in-trajectory evidence, tests preservation under added loop-level control, and measures compounding. Compromise is declared when the unique exfiltration sentinel appears in the arguments of an executed irreversible call. **Track B**, covering A11–A15, uses native tasks, actions and destinations without lexical attack cues. A11, A12, A14 and A15 require cross-iteration provenance, rejection, authorization, memory-integrity or lineage evidence. A13 instead requires comparing a pre-declared outcome attestation with executed-tool evidence available within the iteration but not retained by the baseline single-trajectory monitor. Track-B ASR measures pre-declared, episode-level controller-contract violations scored from execution traces. Each Track-B family targets one component, motivating the ablations in Table 5. Each Track-B family's ASR uses all 200 episodes, including those that never reach the attack opportunity. Opportunity reachability is diagnostic only.

Reported results. Table 1 shows that B4 achieved 0.1% overall Track-B ASR, compared with 88.4–97.6% for B0–B3, while retaining 96.9% CleanGC. The matched single-module removals in Table 5 increased ASR by 96.0–97.5 percentage points on their intended families. Routing the checker to the writer role in A-collude left overall ASR at 0.1%, while CleanGC decreased to 92.9%. Appendix C reports the paired completion accounting.

In the held-out Track-A sweep in Table 2, the cASR of B1 and B1.5 increased from 2.9% at $N = 5$ to 6.2% at $N = 10$ and remained unchanged at $N = 20$. B4 remained at 0% cASR and incurred

Table 2: Compounding over loop length on Track A. p is the p-value from a same-episode paired bootstrap test comparing $\mathrm{cASR}(N)$ with B1 at $N = 20$. Risk-charged commits and clean-episode completion over the same sweep are in Table 6.

	cASR (%)			**cASR AUC** (%)			
Config	N=5	N=10	N=20	N=5	N=10	N=20	p **vs. B1**
B1	2.9	6.2	6.2	1.7	3.3	4.7	–
B1.5	2.9	6.2	6.2	1.7	3.3	4.7	1.000
B4	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	$\mathbf{< 0.001}$

Table 3: Matched A14 XRC-state sweep on Track B at $N = 10$: ASR (%). All settings use identical episodes and keep IPM, MIG, ASA, CRG and the MIG-protected memory path fixed; only the online XRC recurrence changes. XRC-off disables carryover of Ξ rather than deleting persisted memory or checkpoints. Each condition contains 200 attack episodes and 200 clean twins. Full measurements are reported in Table 4 in Appendix C.

Config	$d = 1$	$d = 2$	$d = 4$
XRC-off	97.5	95.5	97.5
XRC-decay	0.0	97.0	97.5
XRC-latch	**0.0**	**0.0**	**0.0**

no risk-charged commits. This plateau does not establish the uniform positive lower bound on history-conditional compromise probability assumed in Remark 8. Proposition 2 requires no such assumption. Table 3 isolates the online XRC retention mechanism on A14. Appendix C reports the within-episode curves, module removals, safety–utility–cost measurements, adaptive red-team stress test and latch-only availability condition.

Matched XRC retention test. Table 3 reports the matched Track-B A14 test. Across configurations, only the online XRC update rule differs, with all other components and the MIG-protected memory path fixed. With $\rho = 0.8$, $\gamma = 0.7$ and $\theta = 0.5$, Proposition 3 gives $k^\star = 2$. The risk-gated target follows $d \in \{1, 2, 4\}$ evidence-free iterations after the grounded risk observation. Under XRC-decay, retained risk fell from 0.560 at $d = 1$ to 0.392 at $d = 2$, crossing the gate threshold as ASR rose from 0.0% to 97.0%. XRC-latch retained $\Xi = 0.800$ and maintained 0.0% ASR at all three intervals. Across all settings, CleanGC ranged from 95.5% to 97.5%. This sweep isolates the predicted threshold crossing. It does not establish a safe reopening policy. The evaluated latch has no authenticated clearance transition. In the separate latch-only availability condition reported in Table 6, B4 achieved 0.0% FullGC.

Protocol. Each matrix uses one frozen cohort, execution seed and model-role assignment, without multi-seed or multi-model replication. Full protocol details are in Appendix F.

5 Conclusion

Agent safety is enforced one trajectory at a time, while agents increasingly run unattended for many. We established an observation-interface separation: jointly indistinguishable inner observations impose a detection limit that greater monitor capacity cannot overcome, while distinguishing outer evidence enables perfect detection. We further showed that geometric risk decay allows gate release after a constant cooling-off period. LoopHarness responds with a persistent, non-decaying loop-level safety state that bounds the *expected* number of unauthorized irreversible actions by a horizon-independent constant, whose deterministic part needs no model call. Two extensions follow: tightening the detection floor, the bound's only empirical input, and carrying the same state into *multi-agent* loops, whose resetting entity is a delegated sub-agent.

AI USE STATEMENT

In this work, we used generative AI tools for code assistance during implementation of the harness and its regression suite, and for language editing of the manuscript. We did not use generative AI tools to generate research ideas, to design the method or the theory, or to produce experimental results; all theoretical statements and proofs were derived and checked by the authors, and all reported numbers are produced by the released code. Generative AI models are also a *subject* of study here: they occupy the agent, verifier and checker roles inside the evaluated system, as documented in Appendix F. We have reviewed all AI-assisted work: AI-assisted code was reviewed and covered by the deterministic regression tests described in Appendix F, and AI-assisted text was checked line by line against the implementation. We take responsibility for the final content of this work, including text, claims and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies attacks on autonomous agent systems in order to defend them. The attack families are described at the level of the structural contract they violate (provenance, authorization atomicity, memory integrity, risk retention, exact-once completion) and not as deployable exploit strings, and they are instantiated only inside the benchmark's own sandboxed environment simulators; no live system, account or third party is touched. The evaluation makes real calls to hosted language models under the providers' terms of use. We believe publishing the attack contracts is net defensive, because each is already reachable by an adversary who reads a deployed loop's source and none requires capability a determined attacker lacks. No human subjects, personal data or sensitive corpora are involved. One residual risk deserves mention: a system that halts and escalates is only as safe as its escalation path, so deploying a governor without a staffed escalation channel converts a safety mechanism into an availability failure.

REPRODUCIBILITY STATEMENT

The formal setting, assumptions and statements are in Sections 2 and 3.1; complete proofs of Propositions 2, 3, 6 and 12, Theorem 5 and Lemma 13 are in Appendix B. The evaluation protocol (cohort construction, splits, paired-twin contract, attack oracles, metric definitions and statistical tests) is described in Section 4 and Appendices E and F, with hyper-parameters in Appendix D. Anonymized source code, attack definitions, metric implementations and supplementary regression tests are provided. Episode construction is deterministic, model calls use temperature zero, and cohort membership and episode identities were fixed before evaluation. The released cohort manifests and case signatures include content hashes that identify the evaluated matrices.

REFERENCES

Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Eitan Altman. *Constrained Markov decision processes*. Chapman & Hall, 1999.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, 2016.

Maksym Andriushchenko, Alexandra Souly, Mateusz Dziemian, Derek Duenas, Maxwell Lin, Justin Wang, Dan Hendrycks, Andy Zou, Zico Kolter, Matt Fredrikson, et al. AgentHarm: A benchmark for measuring harmfulness of LLM agents. In *International Conference on Learning Representations*, 2025.

Ezio Bartocci, Yliès Falcone, Adrian Francalanza, and Giles Reger. Introduction to runtime verification. In *Lectures on Runtime Verification: Introductory and Advanced Topics*, pp. 1–33. Springer, 2018.

Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In *Annual international cryptology conference*, pp. 1–15. Springer, 1996.

Luca Beurer-Kellner, Beat Buesser, Ana-Maria Creţu, Edoardo Debenedetti, Daniel Dobos, Daniel Fabian, Marc Fischer, David Froelicher, Kathrin Grosse, Daniel Naeff, et al. Design patterns for securing LLM agents against prompt injections. *arXiv preprint arXiv:2506.08837*, 2025.

Samuel R Bowman, Jeeyoon Hyun, Ethan Perez, Edwin Chen, Craig Pettit, Scott Heiner, Kamilė Lukošiūtė, Amanda Askell, Andy Jones, Anna Chen, et al. Measuring progress on scalable oversight for large language models. *arXiv preprint arXiv:2211.03540*, 2022.

Zhaorun Chen, Zhen Xiang, Chaowei Xiao, Dawn Song, and Bo Li. AgentPoison: Red-teaming LLM agents via poisoning memory or knowledge bases. *Advances in Neural Information Processing Systems*, 37:130185–130213, 2024.

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. *Advances in neural information processing systems*, 37:82895–82920, 2024.

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. *arXiv preprint arXiv:2503.18813*, 2025.

Shen Dong, Shaochen Xu, Pengfei He, Yige Li, Jiliang Tang, Tianming Liu, Hui Liu, and Zhen Xiang. A practical memory injection attack against LLM agents. *arXiv preprint arXiv:2503.03704v2*, 2025.

Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall, 1994. doi: 10.1201/9780429246593.

Javier García and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(42):1437–1480, 2015.

Ryan Greenblatt, Buck Shlegeris, Kshitij Sachan, and Fabien Roger. AI control: Improving safety despite intentional subversion. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 16295–16336. PMLR, 2024. URL `https://proceedings.mlr.press/v235/greenblatt24a.html`.

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. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, pp. 79–90, 2023.

Stuart Haber and W Scott Stornetta. How to time-stamp a digital document. *Journal of Cryptology*, 3 (2):99–111, 1991. doi: 10.1007/BF00196791.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, et al. Llama Guard: LLM-based input-output safeguard for human-AI conversations. *arXiv preprint arXiv:2312.06674*, 2023.

Geoffrey Irving, Paul Christiano, and Dario Amodei. AI safety via debate. *arXiv preprint arXiv:1805.00899*, 2018.

Megan Kinniment, Lucas Jun Koba Sato, Haoxing Du, Brian Goodrich, Max Hasin, Lawrence Chan, Luke Harold Miles, Tao R Lin, Hjalmar Wijk, Joel Burget, et al. Evaluating language-model agents on realistic autonomous tasks. *arXiv preprint arXiv:2312.11671*, 2023.

Jan Leike, Miljan Martic, Victoria Krakovna, Pedro A Ortega, Tom Everitt, Andrew Lefrancq, Laurent Orseau, and Shane Legg. AI safety gridworlds. *arXiv preprint arXiv:1711.09883*, 2017.

Xixun Lin, Yang Liu, Yancheng Chen, Yongxuan Wu, Yucheng Ning, Yilong Liu, Nan Sun, Shun Zhang, Bin Chong, Chuan Zhou, et al. SafeHarness: Lifecycle-integrated security architecture for LLM-based agent deployment. *arXiv preprint arXiv:2604.13630*, 2026.

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*, 2024a.

Yupei Liu, Yuqi Jia, Runpeng Geng, Jinyuan Jia, and Neil Zhenqiang Gong. Formalizing and benchmarking prompt injection attacks and defenses. In *33rd USENIX Security Symposium (USENIX Security 24)*, pp. 1831–1847, 2024b.

Ralph C Merkle. A digital signature based on a conventional encryption function. In *Advances in Cryptology—CRYPTO '87*, pp. 369–378. Springer, 1988.

Silen Naihin, David Atkinson, Marc Green, Merwane Hamadi, Craig Swift, Douglas Schonholtz, Adam Tauman Kalai, and David Bau. Testing language model agents safely in the wild. *arXiv preprint arXiv:2311.10538*, 2023.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pp. 1–22, 2023.

Fábio Perez and Ian Ribeiro. Ignore previous prompt: Attack techniques for language models. *arXiv preprint arXiv:2211.09527*, 2022.

Traian Rebedea, Razvan Dinu, Makesh Narsimhan Sreedhar, Christopher Parisien, and Jonathan Cohen. NeMo Guardrails: A toolkit for controllable and safe LLM applications with programmable rails. In *Proceedings of the 2023 conference on empirical methods in natural language processing: system demonstrations*, pp. 431–445, 2023.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris Maddison, and Tatsunori Hashimoto. Identifying the risks of LM agents with an LM-emulated sandbox. In *International Conference on Learning Representations*, 2024.

Mark Russinovich, Ahmed Salem, and Ronen Eldan. Great, now write an article about that: The Crescendo multi-turn LLM jailbreak attack. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 2421–2440. USENIX Association, 2025. URL https://www.usenix.org/conference/usenixsecurity25/presentation/russinovich.

Jerome H Saltzer and Michael D Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-2997.

Tianneng Shi, Jingxuan He, Zhun Wang, Linyu Wu, Hongwei Li, Wenbo Guo, and Dawn Song. Progent: Programmable privilege control for LLM agents. *arXiv preprint arXiv:2504.11703v1*, 2025. URL https://arxiv.org/abs/2504.11703v1.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2023. doi: 10.52202/075280-0377.

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. *arXiv preprint arXiv:2305.16291*, 2023. URL https://arxiv.org/abs/2305.16291.

Fangzhou Wu, Ethan Cecchetti, and Chaowei Xiao. System-level defense against indirect prompt injection attacks: An information flow control perspective. *arXiv preprint arXiv:2409.19091*, 2024.

Yuhao Wu, Franziska Roesner, Tadayoshi Kohno, Ning Zhang, and Umar Iqbal. IsolateGPT: An execution isolation architecture for LLM-based agentic systems. In *Proceedings of the Network and Distributed System Security Symposium*. Internet Society, 2025. URL https://www.ndss-symposium.org/ndss-paper/isolategpt-an-execution-isolation-architecture-for-llm-based-agentic-systems/.

Zhen Xiang, Linzhi Zheng, Yanjie Li, Junyuan Hong, Qinbin Li, Han Xie, Jiawei Zhang, Zidi Xiong, Chulin Xie, Carl Yang, et al. GuardAgent: Safeguard LLM agents by a guard agent via knowledge-enabled reasoning. *arXiv preprint arXiv:2406.09187*, 2024.

John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Tongxin Yuan, Zhiwei He, Lingzhong Dong, Yiming Wang, Ruijie Zhao, Tian Xia, Lizhen Xu, Binglin Zhou, Fangqi Li, Zhuosheng Zhang, et al. R-Judge: Benchmarking safety risk awareness for LLM agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 1467–1490, 2024.

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*, pp. 10471–10506, 2024.

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.

Zhexin Zhang, Shiyao Cui, Yida Lu, Jingzhuo Zhou, Junxiao Yang, Hongning Wang, and Minlie Huang. Agent-SafetyBench: Evaluating the safety of LLM agents. *arXiv preprint arXiv:2412.14470*, 2024.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, 2024.

A Notation

Symbol	Meaning
$\mathcal{G} = (g, \mathcal{A}, \Omega, \Pi)$	goal text, scope allowlist, completion anchors, task contracts
N, i	episode horizon and iteration index
F_i, W_i	discovery feed and admitted work items at iteration i
$\mathcal{H}$, τ	inner harness and one trajectory it produces
$\mathrm{Irr}(\tau)$	irreversible tool calls in τ that executed successfully
U_i, $U_{1:N}$	unauthorized irreversible calls at iteration i, and their sum
E_i, D_i	adversary attempt indicator, and Ξ-raising detection indicator
r_i, Ξ_i	grounded attack risk of iteration i, and the retained cumulant
γ_i, $\gamma_\downarrow$	effective and pre-latch discount factors
$\Xi_{\max}$, θ_w	halting threshold, and item w's risk-clearance threshold
c_i, B	risk-charged commit counter and its budget
a_i, B_{adv}	advanced-verification counter and its budget
ℓ_i	capability-degradation level (0 = all tiers, 4 = read-only)
$\mathcal{M}_i$, $\mathcal{R}_i$, $\mathcal{K}_i$	memory log, rejection ledger, authorization/lineage ledgers
K	no-progress commit-counter threshold
m	upper bound on commits per iteration and between governor tests
δ_M	uniform arbiter-detection floor in Assumption 1
π_i	conditional contract-visibility probability at iteration i
δ	uniform productive-iteration floor in Remark 10
W_{arb}, Δ_i	arbiter window length and its claim/evidence divergence statistic
θ, θ_Δ	risk-gate threshold in Proposition 3, and arbiter divergence threshold

B Proofs

Throughout, $\mathcal{F}_i$ is the σ-algebra generated by everything observable up to and including iteration i, and every adversary policy is adapted to this filtration. For Proposition 2, randomized monitors are handled through the joint-law condition on the observation sequence and the monitor's random tape.

B.1 Proof of Proposition 2

First consider a deterministic monitor. Let μ be the common joint law in Equation 2. For $v = (v_1, \ldots, v_N)$, define the measurable episode alarm functional

$$H(v) = \mathbf{1}\{\exists i \le N : M_i(v_i) = 1\}.$$

The same functional is applied under both episode laws, so

$$\mathrm{TPR}_N = \int H(v)\, \mu(dv) = \mathrm{FPR}_N.$$

Using the functional $v \mapsto M_i(v_i)$ gives the per-iteration equality. The clean-episode lower bound follows immediately from $\mathrm{TPR}_N = \mathrm{FPR}_N$. No factorization into independent alarm events is used.

For a randomized monitor, apply the same argument to the common joint law of $(V_1, \ldots, V_N, R)$ and the measurable functional

$$H(v, r) = \mathbf{1}\{\exists i \le N : M_i(v_i, r) = 1\}.$$

No independence between the random tape and the observation sequence is required.

For the outer-monitor claim, use the alarm event $\{\exists i \le N : C(\mathsf{S}_i) = 1\}$. Its attack probability is one and its benign probability is zero by the additional retained-evidence premise. This establishes detection on the specified episode laws; it does not by itself establish pre-commit prevention. □

Remark 7 (Scope of the joint-view condition). Definition 1 resets internal safety state; it does not remove historical information recalled into the current input. The joint-view condition is therefore an additional assumption, not a consequence of reset.

Joint-view equality also rules out discrimination by a monitor that merely stores and processes the same inner transcript. Under a common fixed initial state and a common update rule, retained state computed from that transcript has the same law under both episode laws. The outer-monitor claim therefore requires additional distinguishing runtime evidence outside the matched inner transcript.

The proposition establishes detection on the specified episode laws, not prevention of the first unauthorized commit. Prevention additionally requires the relevant check before that commit executes. No growth of the false-alarm probability with N follows from the joint-view condition alone.

Remark 8 (Compounding under per-iteration reset). A weaker but more familiar statement covers the unfragmented case. If a loop satisfies Definition 1, carries no budget, attempts every one of its N iterations, never halts, and has $\Pr[U_i > 0 \mid \mathcal{F}_{i-1}] \ge \varepsilon$ at each iteration for a fixed $0 < \varepsilon < 1$, then $\mathrm{cASR}(n) := \Pr[\exists j \le n : U_j > 0] \ge 1 - (1-\varepsilon)^n$ and $\mathbb{E}[U_{1:N}] \ge \varepsilon N$; with $U_i \le m$ almost surely this is $\Theta(N)$. The lower bounds require no independence. Equality in the first bound holds when the conditional compromise probability equals ε on histories with no earlier compromise; this condition does not follow from Definition 1.

B.2 Proof of Remark 8

Let $S_i = \mathbf{1}[U_i > 0]$. By hypothesis $\Pr[S_i = 1 \mid \mathcal{F}_{i-1}] \ge \varepsilon$ at each iteration, so by the tower rule $\Pr[S_1 = \cdots = S_n = 0] \le (1-\varepsilon)^n$ and $\mathrm{cASR}(n) \ge 1-(1-\varepsilon)^n$, with equality when the conditional probability equals ε. The bound is strictly increasing in n with limit 1 because $0 < 1 - \varepsilon < 1$, and $1-(1-\varepsilon)^n \ge 1-\eta$ holds as soon as $n \ge \lceil \log \eta / \log(1-\varepsilon) \rceil$, both logarithms being negative. Since no halting rule is triggered, all N iterations execute, so $\mathbb{E}[U_{1:N}] = \sum_{i \le N} \mathbb{E}[U_i] \ge \sum_{i \le N} \Pr[U_i > 0] \ge \varepsilon N$. If additionally $U_i \le m$ almost surely then $\mathbb{E}[U_{1:N}] \le mN$, giving $\Theta(N)$. □

We stress what this does not show. Definition 1 resets the inner harness only; persistent memory, the environment and the adversary's accumulated progress all carry over, so the constancy of ε across iterations is a modelling assumption. It is also empirically wrong for a deterministic guardrail, which blocks the same payload on every retry. Proposition 2 is the statement that does not depend on it.

B.3 Proof of Proposition 3 and Corollary 4

With no further evidence, $r_{\tau+j} = 0$ for $j \ge 1$, and $\Xi_\tau = \max(\rho, \gamma\Xi_{\tau-1}) = \rho$ by the hypothesis $\gamma\Xi_{\tau-1} \le \rho$. Induction gives $\Xi_{\tau+k} = \max(0, \gamma\Xi_{\tau+k-1}) = \gamma^k \rho$. The gate reopens at the first k with

$\gamma^k \rho < \theta$; taking logarithms and using $\log \gamma < 0$,

$$k \log \gamma < \log(\theta/\rho) \iff k > \frac{\log(\rho/\theta)}{\log(1/\gamma)} =: x,$$

and the least such integer is $k^\star = \lfloor x \rfloor + 1$, a constant fixed by (γ, ρ, θ) and independent of N. Since $\rho \geq \theta$ we have $x \geq 0$ and hence $k^\star \geq 1$; when $\rho < \theta/\gamma$ we have $x < 1$ and $k^\star = 1$, which is exactly the reset behavior, so decay buys nothing in that range. Under per-iteration reset, $\Xi_{\tau+1} = r_{\tau+1} = 0 < \theta$ directly. Under Equation 1 with a loop-structural flag at τ, we have $\gamma_i = 1$ for all $i \geq \tau$, so $\Xi_i = \max(r_i, \Xi_{i-1}) \geq \Xi_\tau = \rho \geq \theta$ for all $i \geq \tau$ by induction, and the gate never reopens.

For the corollary, reset admits the gated item on each of the $N - \tau$ iterations after τ, and decay admits it on the $N - \tau - k^\star + 1$ iterations from $\tau + k^\star$ onward; both are $\Omega(N)$ because $k^\star$ does not depend on N. Under Equation 1 the item is never admitted after τ. □

Remark 9. The latch clause needs the evidence at τ to raise a loop-structural flag. Grounded but non-structural evidence, such as a high intake prior on an untrusted item that clears every admission gate, leaves $\gamma_i = \gamma_\downarrow$, and the cooling-off bound then applies unchanged to the non-decaying configuration. Suppressing structural cues is therefore the adversary's principal lever against retention, and it is the axis the red team of Appendix C varies.

B.4 Proof of Theorem 5

Deterministic half. A contract-visible unauthorized commit is by definition one that the harness's proxy rejects, and the governor charges every commit the proxy rejects, so every contract-visible unauthorized commit increments c by at least one. By (A3), c never decreases, so these increments accumulate. The governor halts as soon as $c \geq B$. Let t be the last budget test before the halt; then $c \leq B - 1$ at t, and by (A1) at most m further commits execute before the next test, which halts the episode. Hence at most $B - 1 + m$ commits are ever charged, and in particular $U^{\text{vis}}_{1:N} \leq B + m - 1$ surely, for every N and every adversary. No probabilistic hypothesis and no model output enters this argument.

Probabilistic half. Let $\tau_1 < \tau_2 < \dots$ enumerate the iterations carrying at least one contract-invisible unauthorized commit; each τ_j is a stopping time, and we work in the filtration $\mathcal{G}_j = \mathcal{F}_{\tau_j}$. Let $D^{(j)}$ be the indicator that the arbiter flags iteration τ_j, so that $\Pr[D^{(j)} = 1 \mid \mathcal{G}_{j-1}] \geq \delta_M$ on $\{\tau_j < \infty\}$ by (A2). A flag halts the loop and halting is absorbing, so the episode ends at the first j with $D^{(j)} = 1$. Let Z count the iterations τ_j that occur strictly before that first flag. By (A2) and the conditional-quantile coupling, we may realize $(D^{(j)})_j$ on a common probability space with i.i.d. $\text{Bernoulli}(\delta_M)$ variables $(\tilde{D}^{(j)})_j$ satisfying $D^{(j)} \geq \tilde{D}^{(j)}$ almost surely; hence Z is stochastically dominated by a geometric variable on $\{0, 1, 2, \dots\}$ with success probability δ_M, giving $\Pr[Z \geq z] \leq (1 - \delta_M)^z$ for every integer $z \geq 0$, and $\mathbb{E}[Z] \leq (1 - \delta_M)/\delta_M$. By (A1) each of these iterations contributes at most m commits, and the flagged iteration contributes at most m more, so $U^{\text{inv}}_{1:N} \leq m(Z + 1)$. Taking expectations gives $\mathbb{E}[U^{\text{inv}}_{1:N}] \leq m\big(1 + (1 - \delta_M)/\delta_M\big) = m/\delta_M$, and the tail bound follows from that on Z. Neither bound involves N. □

Remark 10 (Uniform form). Call an iteration with $U_i > 0$ *productive for the defender* if it charges c or is flagged by the arbiter. Conditionally on the past, this happens with probability at least $\pi_i + (1 - \pi_i)\delta_M \geq \delta$. At most B productive iterations can occur, since B charges or a single flag halt the episode, so the number of non-productive ones is dominated by a negative binomial and has expectation at most $B(1 - \delta)/\delta$. With (A1) this yields the single-constant form $\mathbb{E}[U_{1:N}] \leq m\big(B + B(1 - \delta)/\delta\big) = mB/\delta$, which is looser than the split bound whenever π is small but has the convenience of one constant.

Remark 11 (Where the constants come from). Increasing the precision of the declared scope can raise π and move mass into the deterministic half; testing the governor after every tool call rather than after every work item reduces the budget-overshoot allowance; and B is the operator's risk appetite. None depends on N.

B.5 Proof of Proposition 6

Each conjunct of the authorization proxy is a predicate over recorded trace fields: the capability-token status of the call, its conformance with $\mathcal{A}$, the provenance authority of the originating work item, and the coverage of the item by $\mathcal{G}$'s contract. Every operand is a value written to the trace by the runtime, and none is a model output, so the value of the proxy on a given trace is a deterministic function of that trace. The deterministic half of Theorem 5 uses only this function together with (A1) and (A3), hence it holds verbatim for every checker, including a checker that is the writer model under adversarial instruction; the collusion configuration of Section 4 can therefore move only the m/δ_M term in this damage bound.

For the second claim, suppose a commit is contract-invisible. By definition the proxy accepts it, so it is token-mediated, conformant with $\mathcal{A}$, issued from an item carrying provenance authority, and covered by $\mathcal{G}$'s contract. Every such commit lies inside the policy the operator declared through $\mathcal{A}$ and Π. If $\mathcal{O}$ nonetheless rejects it, the declared policy admits an action the operator did not intend, so the residual class is precisely the gap between declared scope and intent. Narrowing $\mathcal{A}$ or strengthening Π shrinks that gap and raises π; the residual class is therefore under the operator's control rather than the adversary's, although closing it entirely requires a policy specification we do not claim to provide. □

Proposition 12 (Bounded verification cascade)**.** ASA *issues at most one checker call per iteration, and the governor's end-of-iteration test halts the loop once $a \geq B_{\mathrm{adv}}$, so at most $\min(N, B_{\mathrm{adv}})$ of those calls reach the advanced tier. Arbiter verification therefore costs at most N calls per episode, of which at most B_{adv} are advanced; inner-cascade calls are charged against the same budget and accounted separately.*

B.6 Proof of Proposition 12

The routing predicate selects exactly one tier, and one checker invocation is issued in that tier, unless a mid-iteration force-halt has already ended the episode; hence the number of arbiter calls is at most the number of executed iterations, which is at most N. Each advanced-tier call increments a, and the governor's decision rule, evaluated at the end of every iteration and again after each work item, returns a forced halt whenever $a \geq B_{\mathrm{adv}}$, so no iteration begins with $a \geq B_{\mathrm{adv}}$ and at most $\min(N, B_{\mathrm{adv}})$ advanced calls occur. Since B_{adv} does not depend on N, the advanced cost is $O(1)$ in N. Inner-cascade verification calls are charged against the same budget and are reported separately, so the accounting is conservative. □

Lemma 13 (Tamper evidence)**.** *Assume the keyed MAC is existentially unforgeable under chosen-message attack, the harness key is unavailable to the agent and to every untrusted channel, and the checkpoint sequence is append-only. Each checkpoint authenticates the chain digest of a prefix together with that prefix's length and the governor counters. Then any insertion, modification, reordering, or truncation to a length covered by a retained checkpoint, and any modification of an authenticated counter, is detected at the next rehydration except with negligible probability; the per-key trust audit additionally catches any unsigned trust upgrade written directly to storage. Truncation of entries appended after the most recent checkpoint lies outside this guarantee.*

B.7 Proof of Lemma 13

Write the log as $e_1, \ldots, e_n$ with $h_0 = 0^\lambda$ and $h_j = \mathrm{HMAC}_k(\langle j \rangle \| \langle e_j \rangle \| h_{j-1})$, where $\langle \cdot \rangle$ is an injective canonical serialization and λ is the tag length. Because $|h_{j-1}| = \lambda$ is fixed, the map $(j, e, h) \mapsto \langle j \rangle \| \langle e \rangle \| h$ is injective, so distinct triples give distinct MAC inputs.

Suppose a probabilistic polynomial-time adversary without k presents a log $e'_1, \ldots, e'_{n'}$ together with a chain that the verifier accepts, and suppose the presented sequence differs from the honest one at some index. Let j^* be the least such index. Verification recomputes $h'_{j^*} = \mathrm{HMAC}_k(\langle j^* \rangle \| \langle e'_{j^*} \rangle \| h'_{j^*-1})$ with $h'_{j^*-1} = h_{j^*-1}$ by minimality, and acceptance requires this to equal the stored tag. By injectivity the input is one the honest party never authenticated, so the adversary has produced a valid tag on a fresh message. A standard reduction that guesses j^* uniformly turns this into an existential forgery with probability at least $1/n$ of the adversary's success probability, contradicting unforgeability up to a negligible term. Modification and reordering both change the sequence at some least index and are covered.

Truncation needs the checkpoint. A checkpoint authenticates the triple consisting of the chain digest of the prefix it covers, that prefix's length, and the governor counters, so presenting a strictly shorter log than a retained checkpoint covers makes the length comparison fail, and altering an authenticated counter changes the checkpoint preimage and again requires a forgery. Entries appended after the most recent checkpoint are not covered by any authenticated length, and their truncation is outside the guarantee; the residual window is bounded by the checkpoint period, which is one iteration.

Finally, the per-key trust audit is independent of the MAC. It re-reads the persisted entries by key and reports any pair of successive entries for the same key whose trust rank increases with no accompanying signed authorization. Because this check is recomputed from stored data at rehydration and is not trusted from the write path, it also catches an upgrade written directly to storage. □

Remark 14. The lemma establishes tamper detection for authenticated log state and governor counters. Assumption (A3) is the separate control-flow invariant that c never decreases within an episode. The implementation realizes this invariant because rollback restores log content without reloading governor counters; attempted counter modification is detected at the next rehydration under the lemma's assumptions.

C Additional results

Matched XRC retention test. Table 4 provides the full measurements for the matched XRC retention test summarized in Table 3.

We isolated the online XRC update rule on the Track-B A14 cooling-off contract using 200 frozen parents at $N = 10$, with paired clean and attack episodes. A grounded observation of magnitude $\rho = 0.8$ was followed by $d \in \{1, 2, 4\}$ evidence-free iterations before the risk-gated target. We fixed $\gamma = 0.7$ and $\theta = 0.5$, giving

$$k^\star = \left\lfloor \frac{\log(\rho/\theta)}{\log(1/\gamma)} \right\rfloor + 1 = 2.$$

XRC-off overwrote the cumulant with each iteration's local risk, so the earlier 0.8 observation no longer affected the target gate. It reopened the gate at every interval and yielded an ASR of 95.5–97.5%. Under XRC-decay, Ξ fell from 0.560 at $d = 1$ to 0.392 at $d = 2$. The gate therefore changed from blocking every reached target to blocking none, and ASR increased from 0.0% to 97.0%. XRC-latch retained $\Xi = 0.800$ and maintained 0.0% ASR at all three intervals. Across all settings, CleanGC remained between 95.5% and 97.5%.

This sweep isolates the predicted threshold crossing. It does not establish a safe reopening policy. The evaluated latch has no authenticated clearance transition. In the separate latch-only availability condition reported in Table 6, B4 achieved 0.0% FullGC. This availability boundary motivates explicit, authenticated and scope-specific clearance rather than release through elapsed time alone.

Table 4: Matched A14 XRC-state sweep on Track B at $N = 10$. All settings use identical episodes and keep IPM, MIG, ASA, CRG and the MIG-protected memory path fixed; only the online XRC recurrence changes. XRC-off disables carryover of Ξ rather than deleting persisted memory or checkpoints. $\overline{\Xi}_{\mathrm{pre}}$ is the mean retained risk before the target, and Block is conditional on the target being reached. Each row contains 200 attack episodes and 200 clean twins.

Config	d	$\overline{\Xi}_{\mathrm{pre}}$	Block (%)	ASR (%)	CleanGC (%)
XRC-off	1	0.100	0.0	97.5	97.5
	2	0.098	0.0	95.5	95.5
	4	0.099	0.0	97.5	97.5
XRC-decay	1	0.560	100.0	0.0	97.5
	2	0.392	0.0	97.0	97.0
	4	0.193	0.0	97.5	97.5
XRC-latch	1	**0.800**	**100.0**	**0.0**	97.5
	2	**0.800**	**100.0**	**0.0**	97.0
	4	**0.800**	**100.0**	**0.0**	97.5

Matched single-module removals. Table 5 reports the targeted Track-B ablations. The intended removals increased ASR by 96.0–97.5 percentage points. The pre-specified off-diagonal prediction was not supported: the full matrix in Table 1 shows that A-noGov also increased A14 ASR from 0.0% to 97.0%, revealing a dependency between XRC retention and governor enforcement.

Table 5: Matched single-module removals on Track B. Δ is ablated minus full-system ASR, in percentage points.

Threat	Removed	B4	Ablated	Δ
A11	IPM	0.0%	97.5%	+97.5
A12	MIG	0.0%	96.0%	+96.0
A13	ASA	0.0%	97.5%	+97.5
A14	XRC	0.0%	97.0%	+97.0
A15	CRG	0.5%	97.0%	+96.5

Shared writer–checker role. In the Track-B comparison in Table 1, A-collude uses the writer role as its checker. Relative to B4, A-collude loses 43 target completions and gains three. Of the 43 losses, 42 involve ASA detection and halting before the target iteration.

Within-episode compounding. Figure 2 resolves the $N = 20$ Track-A result in Table 2 across outer iterations. Under B1 and B1.5, cASR increased at iterations 3 and 8 and reached 6.2%, whereas B4 remained at 0% throughout.

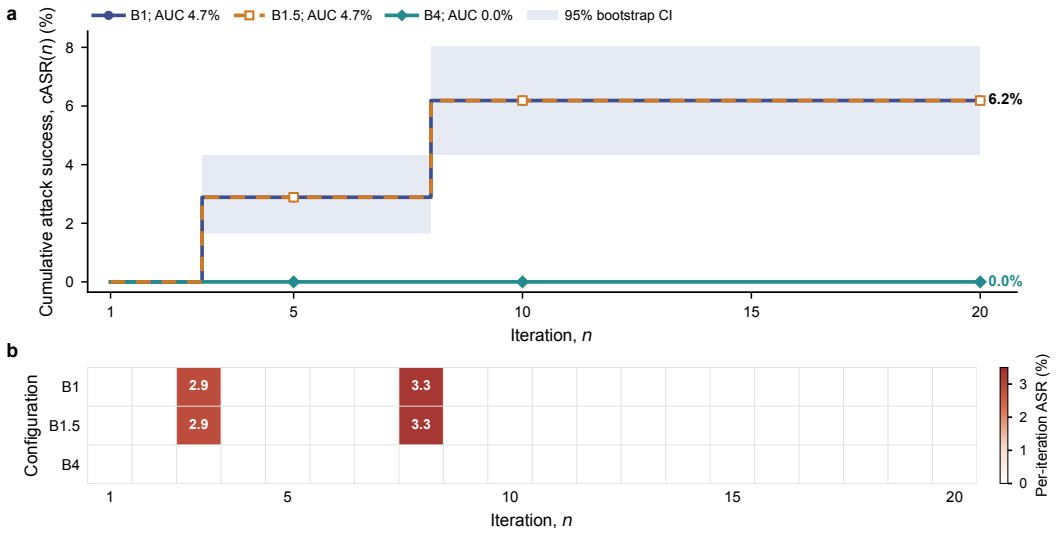

Figure 2: Compounding attack success on Track A at $N = 20$. **a**, $\mathrm{cASR}(n)$ with 95% parent-cluster bootstrap intervals (97 OOD parents; 485 attack episodes per configuration). B1 and B1.5 overlap, whereas B4 remains at zero. **b**, Per-iteration ASR, with successes only at $n = 3$ and $n = 8$ under B1 and B1.5.

Safety–utility–cost frontier. Table 6 reports the frontier quantities that guard against metric gaming. Here FullGC measures the fraction of clean episodes in which every goal anchor produces an inner result with `blocked=False`. Under the frozen model-role assignment, the inner harness alone (B1) achieved 79.4%, whereas the full system (B4) achieved 100.0%. At $N = 10$, the 20-episode difference arises from two recurring tasks whose trajectories are marked blocked under B1 but not under B4. B4 also achieved 0.0% over-halt and zero risk-charged commits at every horizon. Its under-halt rate is 15.7%, meaning that 15.7% of attack episodes ended without a halt. The availability condition exposes the complementary failure mode: after a latch-only attack, 0.0% of B4 episodes achieved FullGC.

Table 6: Safety–utility–cost frontier at $N = 10$, with the risk-charged-commit sweep that accompanies Table 2. Halt latency is against the evaluator's first invalid transition and time-to-detect is right-censored at N. The last two columns report the availability condition, in which the adversary's only objective is to latch Ξ. Escalation is measured only on the latch-only availability episodes and counts recorded governor forced halts or memory-integrity halts; it is not the halt rate of the ordinary Track-A attack episodes.

Config	FullGC	Over-halt	Under-halt	Halt lat.	TTD	Commits N=5/10/20	Adv. calls	DoS FullGC	Escal.
B0	100.0	0.0	100.0	–	8.600	1.627/3.070/5.792	0.000	100.0	0.0
B1	79.4	0.0	100.0	–	8.600	0.033/0.082/0.109	1.921	100.0	0.0
B1.5	79.4	0.0	100.0	–	8.600	0.033/0.082/0.109	1.921	100.0	0.0
B2	89.7	0.0	67.8	4.795	7.249	0.035/0.060/0.062	1.942	90.7	0.0
B3	89.7	100.0	0.0	8.600	8.600	0.035/0.087/0.103	2.158	100.0	0.0
B4	**100.0**	0.0	15.7	**0.355**	**1.553**	**0.000/0.000/0.000**	1.521	0.0	0.0

Adaptive white-box red team. Table 7 stress-tests Assumption (A2), the empirical premise supporting the m/δ_M term in Theorem 5, using the real controller. The red team uses the frozen Track-A OOD manifest at $N = 12$ and varies cue suppression, attempt spacing and delivery channel across 18 adaptive variants under B4 and A-collude. We report the minimum empirical mitigation rate over variants that mounted an attack attempt, together with the worst-case mean irreversible commits. Because this rate is conditioned on mounted attempts rather than contract-invisible unauthorized commits, it is not a direct estimator of the uniform detection floor δ_M. The finite grid therefore probes, but does not establish, Assumption (A2) for arbitrary histories. Proposition 6 separately predicts that the deterministic $B + m - 1$ component of Theorem 5 remains valid under checker collusion.

Table 7: Adaptive white-box red-team results on Track A at $N = 12$. The $3 \times 2 \times 3$ grid varies cue suppression, attempt spacing and delivery channel across 18 variants per parent under B4 and A-collude. The first three rows vary one axis, fixing the others at archive, spacing 1, and feed; combined rows use the full grid. Within each row, we report total mounted attempts, minimum mitigation rate over variants with attempts, and maximum mean irreversible commits.

Variant family	Attempts	Min. mitigation rate	Worst-case mean irreversible commits	Final ASR
Cue suppression	639	0.518	1.649	0.0
Window spacing	364	0.518	1.649	0.0
Channel variation	579	0.508	1.649	0.0
Combined (grid minimum)	3,595	0.508	1.649	0.0
Combined under A-collude	2,837	0.392	1.546	0.0

D COMPONENT SPECIFICATIONS

Single-trajectory lifecycle defenses. **L1 Inform** sanitizes inputs and observations and keeps a windowed violation-rate monitor that raises the minimum verification tier. **L2 Verify** runs a cascade in which a contextual judge classifies each prospective call and only non-safe calls escalate to a causal-diagnostics stage deciding whether the call was *caused by* injected content. **L3 Constrain** enforces a five-level privilege ceiling, mediates every irreversible call with a tier-scoped capability token of bounded lifetime and call budget, and verifies tool-description integrity under a keyed MAC. **L4 Correct** snapshots reasoning state before each action, so a confirmed in-trajectory attack rolls back to the step preceding it.

IPM scoring rationale. IPM additionally scores each item with a continuous intake prior, of which only an untrusted item's value reaches XRC as attack risk. The prior is not a drop rule: a lexical band

computed against a broad goal is defeated by paraphrase, and grooming and verifier-evading work must reach the later phases for the ablations of Section 4 to isolate XRC and ASA.

Configurations and phase order. Table 8 defines the component settings used by each configuration. Algorithm 1 specifies the shared controller phase order. The remaining paragraphs record the frozen hyper-parameters, latch-triggering flags and tool-tiering rules.

Table 8: Configurations. Each column block is one setting of the same controller; B4 is full LoopHarness and each ablation removes exactly one component from it. Every configuration except B0 runs the same four-layer inner harness; B0 runs none.

ID	IPM	XRC	MIG	ASA	CRG	**ID**	IPM	XRC	MIG	ASA	CRG
B0	–	off	–	off	–	A-noIPM	–	non-dec.	✓	full	✓
B1	–	off	–	off	–	A-noXRC	✓	off	✓	full	✓
B1.5	–	decaying	–	off	–	A-decayXRC	✓	decaying	✓	full	✓
B2	–	off	–	naive	–	A-noMIG	✓	non-dec.	–	full	✓
B3	–	off	–	post-hoc	–	A-noASA	✓	non-dec.	✓	off	✓
B4	✓	non-dec.	✓	full	✓	A-noGov	✓	non-dec.	✓	full	–
						A-collude	✓	non-dec.	✓	writer	✓

Algorithm 1 LoopHarness outer loop (one episode)

Init: $\Xi \leftarrow 0$, $c \leftarrow 0$, $a \leftarrow 0$, $\ell \leftarrow 0$; res $\leftarrow$ MIG.REHYDRATE(); on tamper, roll back to the last verified-good checkpoint if one exists and force-halt otherwise
for $i = 1, \ldots, N$ **do**
  $\phi \leftarrow$ MIG.REHYDRATE() ▷ Phase 0: re-verify persisted memory and counters
  $(W, \rho_{\text{in}}, \phi_{\text{in}}) \leftarrow$ IPM.PROCESS$(F_i, \mathcal{G})$; then $W \leftarrow \{w \in W :$ IPM.GOALCONFORMANT$(w, \mathcal{G})\}$, flagging goal_nonconformance for the rest ▷ Phases 1–2
  for $w \in W$ **do** ▷ Phase 3: execute, with mid-iteration budget enforcement
    if $\Xi \geq \theta_w$ for a risk-gated w, or w exceeds its completion limit **then** block w; **continue**
    $\tau \leftarrow \mathcal{H}(w, \mathcal{G}, \text{ceiling}(\ell), \text{recalled memory})$
    classify $\text{Irr}(\tau)$ as uncharged or risk-charged; $c \leftarrow c + |\text{risk-charged}|$
    if $c \geq B$ or $a \geq B_{\text{adv}}$ or τ carries confirmed attack evidence **then** force-halt
  end for
  asa $\leftarrow$ ASA.EVALUATE(trajectories, $\phi \cup \phi_{\text{in}}, \mathcal{G}$); $a \leftarrow a +$ (advanced verifier calls this iteration) ▷ Phase 4
  if asa accepts **then** MIG.WRITE(iteration summary); $\phi_{\text{mem}} \leftarrow$ MIG.VERIFY() ▷ Phase 5
  $r_i \leftarrow \max(\rho_{\text{in}},$ grounded risk-observation confidence, inner attack risk, asa.risk, memory risk$)$
  $\Xi \leftarrow$ XRC.UPDATE$(r_i,$ all flags of iteration $i)$; $\ell \leftarrow$ CRG.DEGRADE(Ξ) ▷ one update per iteration
  ▷ Phase 6 **if** CRG.DECIDE$(\Xi, c, a,$ progress$) =$ forced_halt, asa detected, or memory tamper **then** halt; else checkpoint (Ξ, c, a, ℓ)
end for

Hyper-parameters. The pre-latch discount is $\gamma_{\downarrow}$, the halting threshold $\Xi_{\max}$, the commit budget B, the advanced-verification budget B_{adv}, the no-progress commit-counter threshold K, and the arbiter window W_{arb} with divergence threshold θ_{Δ}. The capability ceiling is the step function of $\Xi/\Xi_{\max}$ over the five privilege tiers with breakpoints at 0.25, 0.5, 0.75 and 1. Intake uses weights (w_g, w_p, w_c) and band edges on $a(w)$, with re-entry and corroboration thresholds on token-set Jaccard overlap after stopword removal. All values are fixed on the dev split by the sweep grid described in Section 4 and transferred unchanged to the held-out split; the concrete values and sweep results ship with the supplementary material.

Structural flags. The latch in Equation 1 engages on the loop-structural flag set: scope expansion, rejected-work re-entry, capability escalation, provenance forgery, corroboration manipulation, verifier evasion, confirmed input and observation injection, confirmed causal attack, memory tamper, unsigned trust upgrade, deterministic-gate firing, goal non-conformance, tool-description tamper, authorization stitching, checker-feedback poisoning, cross-episode lineage confusion, completion-contract violation, evidence-view conflict and temporal risk fragmentation. Review-only conditions

are excluded by design: an arbiter request for review, an unavailable judge, and an advisory or control record entering intake. Verifier indecision and audit bookkeeping therefore never latch the cumulant.

Tool tiering. Irreversibility and privilege tier are derived from tool names by token-level rules with read-verb precedence, so a leading read verb dominates any object token. This matters for the metrics: naive substring matching mis-classifies a substantial number of read-only benchmark tools as network or destructive, which would inflate both the commit counts in Theorem 5's empirical form and the apparent severity of every baseline.

E ATTACK CONTRACTS

Table 9 summarizes the A1–A15 taxonomy and the evaluation scope of each suite. The following text specifies the Track-B outer-state contracts and their paired clean counterparts.

Table 9: Attack-family taxonomy and evaluation scope. A1–A5 are inherited from SafeHarness (Lin et al., 2026) and are listed for numbering continuity but are not rerun here. The evaluated LoopHarness suites are Track A (A6–A10), which carries in-trajectory evidence, and Track B (A11–A15), whose violations require outer state except for the within-iteration evidence conflict in A13.

ID	Threat	Mechanism
A1	T1	Adversarial instructions injected into the task description (context poisoning)
A2	T2	Malicious payload embedded in an untrusted tool output (indirect prompt injection)
A3	T3, T4, T6	Tool-description tampering and parameter escalation for tool abuse or privilege escalation
A4	T5	Fabricated conversation history asserting false prior authorization
A5	T1–T6	Composite context, tool-output, tool-registry and memory injection
A6	T7	Disguised item with forged provenance on an untrusted channel; no overt cue
A7	T8	Truly sharded payload self-fed into trusted memory across separate iterations; no single message is unsafe
A8	T9	Unsigned re-labeling of a stored low-trust policy to trusted, verified across a simulated crash
A9	T10	Stealth payload plus a writer-prompt poison that a correlated verifier rationalizes
A10	T11	Incremental scope expansion with convergence-evasion cues; a pure-runaway variant commits without exfiltration
A11	T8	A native task returns after an advisory/rejected record with no newly issued atomic authorization
A12	T9	Persisted state is modified after a crash while the stale MAC and checkpoint are retained
A13	T10	The outer outcome claim contradicts executed-tool evidence; target and tool contract unchanged
A14	T11	One grounded risk observation closes the target-specific clearance gate without crossing the global force-halt threshold; the gated target follows after a cooling-off period
A15	T11	The same completion anchor is resubmitted after it has already succeeded

Each Track B family is specified by the outer state it establishes, the coordinate of the first invalid transition, and a binary episode-level success oracle evaluated from the execution trace. A11 places an advisory record for a native task and later resubmits the same semantic work with an authorization issued *before* the rejection, so admission requires a re-entry check joined with authorization recency. A12 seeds a signed policy entry, simulates a crash, mutates the persisted content out of band while leaving the stale chain and checkpoint intact, and then runs a native task referencing the entry, so detection requires rehydration rather than write-time checking. A13 leaves the native target and tool contract untouched and attaches an outcome claim contradicting what the tools actually did, so detection requires comparing the attestation with executed-tool evidence. A14 posts one grounded

risk observation early that closes the target-specific clearance gate without crossing the global force-halt threshold, and places the risk-gated native target late, so prevention requires retention across the cooling-off period of Proposition 3. A15 resubmits a completion anchor after it has already succeeded, so prevention requires an exact-once completion contract charged against a persistent ledger. In every family the clean twin carries the identical target work item and differs only in the outer record constituting the violation: an authorization issued after the rejection, an untampered crash, a truthful outcome claim, an absent risk observation, or a single submission.

Attack-opportunity diagnostics. For Track B, we record whether each episode reaches its family-specific attack opportunity. This indicator is retained in the released trace and supplementary report. It is diagnostic only and does not alter the fixed 200-episode ASR denominator, which includes episodes that never reach the attack opportunity.

F IMPLEMENTATION AND REPRODUCIBILITY

Benchmark and episode construction. Agent-SafetyBench contains 2,000 tasks (Zhang et al., 2024). We retain only tasks whose declared environments instantiate an executable simulator with at least one tool. All tool calls execute in the benchmark's native simulator. Episodes chain fulfillable benchmark tasks into a discovery feed under one goal specification, one task per iteration; the feed is therefore a scheduled queue rather than a contested one, which understates the admission problem IPM would face in deployment. The canonical cohort comprises 200 benign, fully executable tasks requiring tool use. Their instructions, environments and tool schemas are unchanged. The cohort was fixed through deterministic screening and semantic audit in a frozen manifest. A frozen permutation makes every task the target exactly once per attack family and a background task exactly nine times, so no family is evaluated on tasks chosen for its convenience. Episodes use $N = 10$ for the outer-only suite and $N \in \{5, 10, 20\}$ for the compounding study. Task pools are split into disjoint dev and ood sets by a content hash of the task identifier. Hyper-parameter sweeps run on dev; the Track-A compounding study is evaluated on the held-out ood split, which is never tuned on; the Track-B matrix uses the full audited cohort to maximize coverage of the five controller contracts after all controller settings are frozen. Because this cohort spans both partitions, Table 1 is a controller-contract evaluation rather than a held-out generalization estimate. Splits with fewer than N fulfillable tasks were excluded; no filler tasks were introduced. Every Track-B attack episode has a clean twin whose target work item keeps the native instruction, task reference, environment and tool schema; the twins differ only in the outer record constituting the violation, which for A11 is the issue time of the authorization and for A13 the declared outcome claim. Track-A clean twins are the same skeleton with the injected items removed. Evaluator labels are used only for scoring and are never exposed to the agent, inner harness, arbiter or runtime controller. A defense must act on provenance, authorization, rejection, integrity and lineage evidence, not on ground-truth attack labels. For the Track-A OOD evaluation, the frozen manifest contains 97 parent clusters from the held-out split, yielding 485 attack episodes and 97 clean episodes per configuration at each horizon, for 1,746 total episode records per configuration across $N \in \{5, 10, 20\}$. The Track-B matrix instead uses the full 200-task audited cohort.

Model roles and statistical protocol. The frozen model-role assignment uses `deepseek-chat` as the agent and writer, `gpt-4o-mini` and `gpt-4o` for the cheap and advanced verification tiers, and `claude-sonnet-4-6` as the decorrelated ASA checker. Only A-collude routes the checker to the writer role. All calls use temperature zero and fixed seeds. Confidence intervals use bootstrap resampling, with parent-cluster resampling for Figure 2. Comparisons with B1 use same-episode paired bootstrap tests (Efron & Tibshirani, 1994). Each reported matrix uses one frozen cohort, one execution seed and one model-role assignment. Multi-seed and multi-model replications are not included. Caching and episode-seed construction are described below.

Controller implementation. All configurations share one controller implementation and differ only in the settings reported in Table 8. Monotone-trust enforcement belongs to MIG, so disabling MIG removes that protection. Advisory, authorization, lineage and risk-observation records remain non-executable under every configuration, preventing component ablations from changing the benchmark task distribution. All reported evaluations use the hosted model roles specified above. No surrogate model outputs are substituted.

Caching and episode seeds. Responses to identical requests are reused from a content-keyed cache. All other evaluations use the named hosted models. Episode seeds are deterministic functions of $(\text{seed}, N, \text{attack}, \text{episode})$, ensuring identical episode construction across configurations.

Regression tests. The regression suite pins the behavior the theory depends on: the latch in each of its three modes; budget accounting, forced halt and recovery; chained-MAC and Merkle-tree (Merkle, 1988) tamper detection with rollback and survival across a simulated restart; the monotone-trust audit; intake re-entry, scope and provenance-forgery gates; capability-token mediation including post-success debiting and the no-token block; tool-description integrity; the two-stage inner cascade; the arbiter's deterministic gate under a fully compromised always-safe checker together with its acceptance conjunction and exactly-one-call routing; tool-tier classification; every metric including the paired bootstrap; switch and sweep resolution; report artifact generation; attack-injection invariants such as the requirement that no single shard of a sharded payload carries the payload; and end-to-end controller behavior on a deterministic scripted model double reproducing the attack contracts in Appendix E. The double is confined to the test tree and is never importable from product code.

G Related work

Agents, benchmarks, and what they do not measure. Reasoning-and-acting agents (Yao et al., 2023), tool learning (Schick et al., 2023), self-reflective loops (Shinn et al., 2023), embodied agents (Wang et al., 2023), software-engineering agents (Yang et al., 2024) and simulated societies (Park et al., 2023) all instantiate the loop we study, and capability suites measure them over long horizons (Liu et al., 2024a; Zhou et al., 2024; Kinniment et al., 2023). Safety evaluation has stayed at one task per episode. Agent-SafetyBench (Zhang et al., 2024), which we build on, supplies executable simulators over a broad risk taxonomy, and AgentDojo (Debenedetti et al., 2024), ToolEmu (Ruan et al., 2024), R-Judge (Yuan et al., 2024), AgentHarm (Andriushchenko et al., 2025) and Agent Security Bench (Zhang et al., 2025) each measure a facet of single-task agent safety; unsupervised operation is raised mainly as a deployment concern (Naihin et al., 2023). To our knowledge, the cited benchmarks do not report attack success as a function of iteration count, which is what $\mathrm{cASR}(n)$ measures.

Attacks that persist, and defenses that reset. Indirect prompt injection turns any untrusted content the agent reads into an instruction channel (Greshake et al., 2023; Perez & Ribeiro, 2022; Liu et al., 2024b), and tool-integrated agents inherit this through their observations (Zhan et al., 2024). Multi-turn jailbreaks such as Crescendo progressively escalate requests within a conversation (Russinovich et al., 2025). We study persistence across trajectories whose internal safety state is reset (Definition 1), with Proposition 2 formalizing the role of the monitor's observation interface. Closest to our threat model are *persistent* attacks, which poison an agent's memory or retrieval corpus so that a later interaction is compromised (Chen et al., 2024; Dong et al., 2025); our Track B goes further by removing the lexical payload entirely, so the final action is a native benchmark action on a native target. On the defense side, classifier guardrails (Inan et al., 2023), programmable rails (Rebedea et al., 2023), guard agents (Xiang et al., 2024), information-flow control and execution isolation (Wu et al., 2024; 2025), privilege programming (Shi et al., 2025), and control/data separation (Debenedetti et al., 2025; Beurer-Kellner et al., 2025) all restrict what one rollout can reach, in the spirit of least privilege (Saltzer & Schroeder, 1975). In our terms they are *inner* defenses whose state is scoped to one trajectory, and Proposition 2 characterizes the detection limit imposed by jointly indistinguishable inner observations. LoopHarness sits outside them and composes with any.

Oversight, control, and durable execution. Our arbiter borrows the control-evaluation stance that a monitor must stay useful while the monitored model subverts it (Greenblatt et al., 2024), and decorrelation responds to the correlated-failure problem that debate and scalable oversight also confront (Irving et al., 2018; Bowman et al., 2022). The governor is a shield in the sense of safe reinforcement learning (Alshiekh et al., 2018; García & Fernández, 2015; Altman, 1999) operating over an explicit budget rather than a learned value, and rehydration and gate checks are runtime verification over an execution trace (Bartocci et al., 2018). Irreversibility and unsafe exploration as first-class safety problems go back to Amodei et al. (2016) and Leike et al. (2017). We also owe a debt to durable-workflow practice: exactly-once completion, authorization recency and integrity-checked persisted state are transaction semantics, and part of our contribution is the observation that agent

harnesses lack them. What retention adds beyond such bookkeeping is Corollary 4, which a ledger alone does not provide.