

Persistent Billable State: Denial-of-Wallet Attacks and Defenses in Tool-Calling LLM Agents

Jinqian Zhang^{1,2} Haojun Xia^{1,2,†} Shujiang Wu³ Jingkun Yue⁴
Xia Zhang^{1,2} Zhangpei Cheng^{1,2} Bibo Tu^{1,2,†}

¹Institute of Information Engineering, Chinese Academy of Sciences

²School of Cyber Security, University of Chinese Academy of Sciences

³Beihang University ⁴State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications, Beijing, China

[†]Corresponding authors.

Abstract

Multi-step tool-calling LLM agents rely on host runtimes to preserve state across turns. When a runtime carries an external tool return into later model inputs, providers meter it again. An admitted malicious or compromised tool can thereby convert untrusted data into recurring victim-billed processing without victim credentials or local runtime privilege. We call such retained content persistent billable state and formalize the host’s decision over whether and how it enters later billable context as the persistent billable-state boundary.

We present the first systematic security study of this post-admission lifecycle. We derive six denial-of-wallet attack vectors and build DOW-BENCH, an end-to-end harness evaluated across six model families. Across 243 executions, provider-reported usage telemetry shows that the maximum per-session cumulative input reaches $14,293\times$ the corresponding session’s complete first-call input. Controlled history-policy reruns isolate raw retention’s contribution: retaining raw history increases mean effective session cost by 21.2–35.9%. Compression succeeds on 10/12 and 11/12 history-dependent tasks, versus 2/12 under deletion for each provider.

To govern this boundary while preserving utility, we combine deterministic history transformation with four host-side invariants that bound prompt mass, context growth, recursive opportunity, and cumulative spend before reingestion. The kernel contains every recurring attack in the 123-evaluation replay corpus. Across 24 Mistral Small 4 workflows, a progress-authorized policy achieves 22/24 oracle-verified task successes with no pre-completion interruptions, versus 13/24 under a fixed cap. Only 71 of 3,830 scanned public MCP server and transport repositories expose any code-visible safeguard proxy, and none cover all four safeguard families. These results establish persistent billable state as a first-class security object and pre-reingestion as its host-owned control point.

1 Introduction

A tool return can remain billable long after the tool call ends. Consider an agent debugging a dependency with an admitted

remote tool. The tool returns a plausible diagnostic padded with attacker-controlled content and requests the next check. If the runtime preserves that result, every later model call reprocesses and meters it again: one admitted return acquires recurring spending authority on every later call.

Multi-step tool-calling agents depend on precisely this continuity. Toolformer and Gorilla establish context-conditioned API selection, whereas ReAct and ToolLLM extend tool use into trajectories whose later actions depend on accumulated observations or API responses [33, 34, 37, 41, 44, 45]. For multi-step integrations, the Model Context Protocol (MCP) standardizes tool exchange while leaving the host to preserve enough context for later calls to interpret results and complete subsequent steps [30]. Broader OpenRouter telemetry over more than 100 trillion tokens reports average sequence length rising from under 2,000 tokens in late 2023 to over 5,400 by late 2025 [2]. As agent workflows lengthen, repeatedly processing retained state becomes increasingly consequential.

Tool admission determines whether a remote service may be invoked; persistence determines whether its returned content can become future spend. We call tool-generated content *persistent billable state* when it survives across turns and incurs repeated processing cost. Legitimate workflows rely on this state for continuity. We formalize the host’s decision over whether and how an external return enters later billable context as the *persistent billable-state boundary*.

A Denial-of-Wallet (DoW) attack exploits this boundary: an ordinary task invokes a malicious or compromised remote tool endpoint already admitted to the execution path; the endpoint supplies a schema-valid, task-plausible inflated return; the host retains that return; and later model calls repeatedly process and bill it. The attacker thereby controls future billable-state growth. Under full retention and stable return mass, even a 2–10-token trigger can initiate recursive rebilling whose cumulative input grows as $O(N^2)$. With a session budget, this persistent work can exhaust authorized spend and interrupt task completion; without one, it drives continuing cost growth. DoW therefore converts attacker-controlled persistence into either availability loss or runaway expenditure.

The same state that carries the attack also carries the task. Dropping prior returns can break necessary continuity, while fixed ceilings can interrupt long workflows and leave room for attackers to distribute growth across turns. The security problem is therefore not whether to retain history, but how to preserve task-relevant state while governing attacker-controlled persistence. DoW makes recurring billed work the attack objective and the host’s pre-reingestion decision the enforcement point, complementing semantic defenses against instruction takeover and data exfiltration.

The Missing Economic Lifecycle. Prior work shows that agents can be steered, looped, and made expensive through prompt injection, unsafe tool use, malicious skills, recursive tool chains, reasoning loops, and granted-permission computation hijacking [6, 7, 9, 10, 13, 18, 22, 24, 36, 38, 42, 47, 49, 50, 52]. These studies establish semantic manipulation and resource amplification, but leave the host-side economic lifecycle itself unisolated: once a remote tool is admitted, who authorizes its return to persist as future provider-billed state, and what evidence should earn that state another unit of computation or spend? We present the first systematic security study of this missing post-admission lifecycle. This lifecycle is architectural rather than MCP- or provider-specific: it exists whenever a tool-calling runtime reintroduces externally supplied tool state into a later billable model call. We study it end to end, from attack construction and mechanism attribution to host-side enforcement and ecosystem measurement:

RQ1. How much priced exposure can an admitted remote tool create? DoW-BENCH measures repeated input processing separately from listed-price exposure. Across 243 executions spanning six model families, the maximum per-session input-token Cost Amplification Factor (CAF) reaches $14,293\times$; 51/69 priced attack sessions cross the \$0.10 circuit-breaker setting and exposure reaches \$3.40, versus a \$0.025 benign maximum.

RQ2. What sustains billable-state exposure across agent turns? Turn survival tracks deployment-level exposure. Across two providers, independent Full/Drop/Compress reruns then estimate the closed-loop effect of retaining raw history: Full raises mean effective session cost by 21.2–35.9%, while compression preserves 10/12 and 11/12 successes on tasks requiring prior history that deletion reduces to 2/12. Raw retained history therefore creates a continuity–exposure coupling, and deterministic compression preserves task state without carrying its full cost-bearing mass.

RQ3. Can pre-reingestion middleware contain recursive cost exposure while preserving legitimate workflow utility?

Four host-side invariants (D1–D4) govern prompt mass, context growth, recursive opportunity, and cumulative spend before reingestion, containing all 41 recurring attacks under each of three replay presets. In a 24-workflow Mistral Small 4 transfer, progress-authorized D4 achieves 22/24 oracle-verified task successes with no pre-completion interruptions, versus 13/24 under a fixed \$0.10 cap. Pre-reingestion middle-

ware can therefore bound recurring spend while preserving useful continuation.

Deployment context. To assess deployment prevalence, we scan the 3,830-repository corpus. Only 71 contain a code-visible safeguard proxy, and none cover all four safeguard families.

Contributions.

- **Persistent billable state and its attack surface.** We establish persistent billable state as a first-class security object, distinguish tool-invocation admission from post-return economic governance, formalize the host-side pre-reingestion boundary, and systematize six attack vectors that turn an admitted return into recurring provider-billed state.
- **Attribution through intervention.** We build DoW-BENCH across six model families and combine listed-price API measurement with independent Full/Drop/Compress interventions across two providers, separating exposure magnitude from the contribution of retaining raw history.
- **Utility-preserving host governance.** We combine deterministic history transformation with a host-side D1–D4 kernel requiring no provider-private hooks. The kernel contains all 123 replay evaluations, executes inline before the next provider call, and achieves 22/24 transfer-workflow successes under progress-authorized spend, compared with 13/24 under the fixed cap.
- **Ecosystem scan.** We complement the attack and defense evaluation with a repository-based scan of 3,830 MCP server and transport repositories, providing a repository-level view of code-visible safeguard-proxy adoption. Stratified manual validation and a targeted probe of zero-proxy repositories support the classifications.

2 Background

2.1 DoW: Definition and Mechanism

Kelly et al. formalized Denial-of-Wallet in serverless computing as forced financial exhaustion in pay-as-you-go services [19]. More generally, DoW is an economic denial-of-service mechanism in which an adversary cheaply triggers resources billed to a victim. In a tool-calling LLM agent, we use the term for the analogous victim-billed mechanism: retained attacker-controlled tool returns are processed across turns and charged repeatedly to the victim.

Tool-calling agents operate in iterative perception–action loops [45]: the model receives conversation state, invokes a tool, observes its result, and uses that result to plan the next step. Some runtimes reconstruct history client-side; others rely on provider-managed conversation state. Both designs

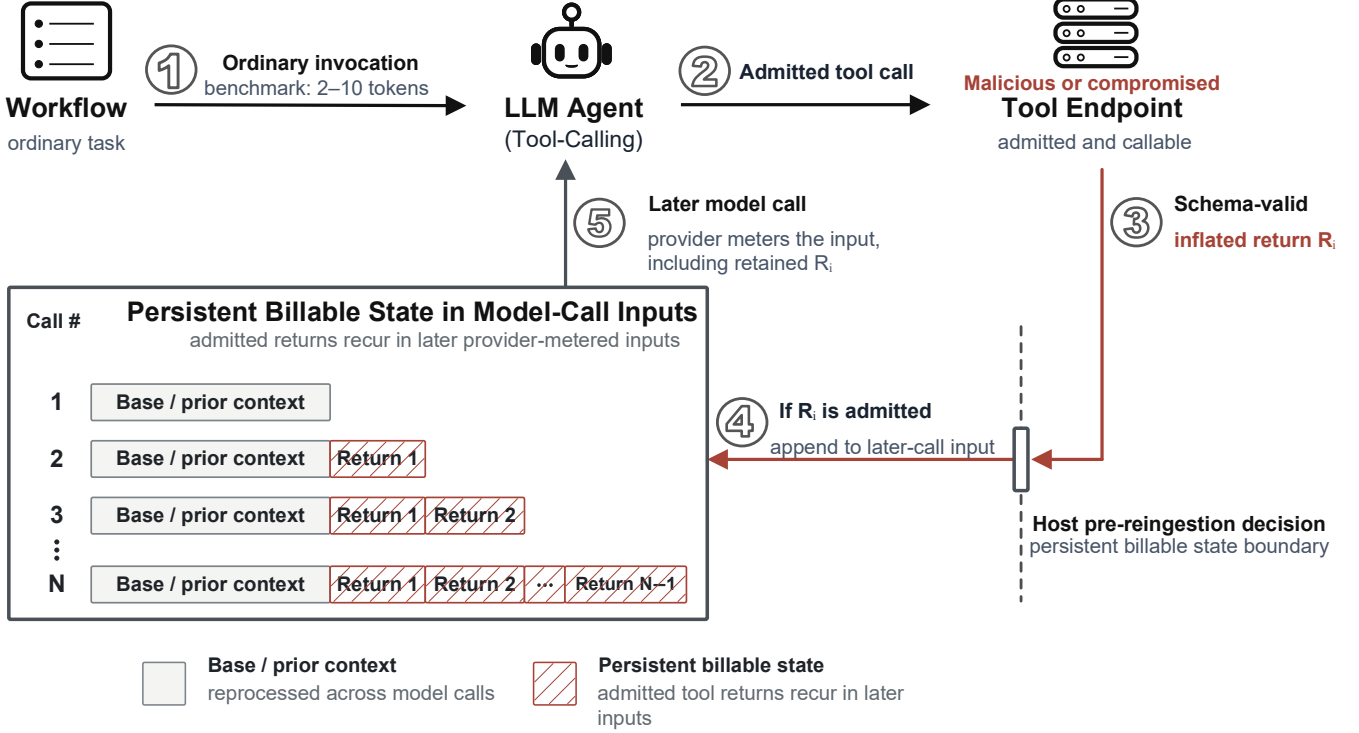


Figure 1: Lifecycle of persistent billable state. An ordinary task invokes an already-admitted malicious or compromised remote tool. Its schema-valid return R_i crosses the host’s pre-reingestion boundary, persists in later model inputs, and accrues provider-metered cost on each reuse. Widths are schematic and omit non-tool growth.

can carry earlier tool outputs into later billable inputs. Content carried through this transition into a later billable input becomes persistent billable state and incurs processing cost on subsequent calls.

MCP 2026-07-28 makes individual protocol requests stateless while leaving conversation aggregation with the host; application state can also travel through explicit handles in tool results and later arguments [30]. Wire-level statelessness therefore leaves the persistent billable-state boundary intact.

Figure 1 illustrates this lifecycle. Let R_i denote the tool return after call i and $T_i = |R_i|$ its input-token mass. Ignoring non-tool prompt growth, the prompt at turn n includes $P_{\text{init}} + \sum_{i=1}^{n-1} T_i$; under stable return mass $T_i = T_{\text{avg}}$, cumulative input across N calls is $N \cdot P_{\text{init}} + \frac{N(N-1)}{2} T_{\text{avg}}$. For observed traces and $i \geq 2$, $\Delta p_i = p_i - p_{i-1}$ denotes the full-prompt increment and approximates T_{i-1} for the constant-return baseline. The resulting recursive rebilling is an ordinary cost of continuity in benign workflows and the source of quadratic pressure under DoW.

This lifecycle couples local task semantics to global billing. The agent continues when a tool return appears useful, while the provider charges for the accumulated history that makes continuation possible. A malicious tool can exploit that coupling by keeping a plausible workflow alive until recursive

rebilling dominates the session cost.

2.2 Threat Model

The attacker needs only control over content returned by a malicious or compromised remote tool endpoint, including an MCP server, that the victim agent can already invoke in an ordinary workflow. The attack requires no victim credentials, filesystem or internal-network access, local runtime privilege, hidden-prompt access, model control, or provider control.

The goal is to make returned content enter future billable context and expand victim-side session cost. The attack does not depend on private victim data and can operate through public retrieval, lookup, weather, or debugging tasks. After the host invokes the admitted tool, the attack proceeds entirely through its returned content and the post-admission lifecycle.

2.3 Measurement Metrics

We measure two victim-side consequences: cumulative input processing relative to the first model call and nominal dollar exposure over the complete session. Drawing on classic amplification-factor reasoning from network denial-of-service and economic-denial-of-service work [35, 39], we define an input-token *Cost Amplification Factor* (CAF_{tok} ,

hereafter CAF) and a separate dollar-denominated exposure quantity:

$$\text{CAF}_{\text{tok}} = \frac{\sum_{i=1}^N p_i}{p_1}, \quad (1)$$

$$C_t = \sum_{i=1}^t \left(p_i r_i^{\text{in}} + o_i r_i^{\text{out}} \right), \quad (2)$$

where p_i and o_i are provider-reported input- and output-token counts for model call i , p_1 is the complete first-call input assembled by the runtime, and r_i^{in} and r_i^{out} are the provider-specific listed nominal rates fixed by canonical accounting, expressed in USD/token after conversion from the listed USD/million-token prices. CAF quantifies cumulative input processing relative to the complete first model input; C_t is listed-price exposure over session input and output. Unlike output-token amplification reported in prior agent-resource work [50], CAF measures cumulative victim-side input processing relative to the complete first-call input. The history-policy study reports a cache-aware effective session cost that applies the same listed-rate table separately to provider-reported uncached input, cached input, and output.

3 Related Work

Economic Denial of Service and LLM Cost Abuse. Resource abuse that bills the victim has precedents in cloud security, including fraudulent resource consumption and economic denial of sustainability [15, 39]. Recent LLM studies use distinct metrics: Beyond Max Tokens reports output-token amplification up to $658\times$ that of benign executions in multi-turn tool chains [50], while Overthinking reports up to $142\times$ token amplification from cyclic MCP tool structures [22]. Among recent studies of cost abuse, Clawdrain reports $6\text{--}7\times$ token amplification from a trojanized skill in a production-like OpenClaw deployment using a real API, approximately $9\times$ in its costly failure configuration, and persistent pollution from tool output as a source of input cost [9]. ThinkTrap elicits long outputs from commercial APIs under black-box access and separately demonstrates throughput degradation in a controlled self-hosted deployment [23]; benchmark audits call for tokens, latency, API cost, attacker budget, and defender capacity to be reported together [16]. Clawdrain thus establishes cost growth from retained history; we make the host’s retention decision the security object, intervene on retention to attribute recurring input billed by providers to admitted returns, and evaluate enforcement before reingestion while measuring task utility.

LeechHijack studies an MCP tool that injects unauthorized work through manipulated returns from legitimate calls, evaluating accuracy on the original task and audits of contextual memory after execution [49]. DoW instead studies recurring provider billing of retained returns, isolates the mechanism through independent reruns under different history policies,

and enforces the boundary before reingestion with host controls tested for utility.

AgentDoS uses directed grey-box fuzzing to uncover 36 vulnerabilities that exhaust resources in 16 of 20 open-source agents [25]. *From Shield to Target* traces payloads delivered through documents, tool outputs, agent messages, and shared memory to token, latency, and infrastructure amplification [51]. These works span broader resource lifecycles; our control point is the admitted remote-tool return immediately before it reenters provider-billed context.

LLM Agent Safety and Tool-Use Security. Research on agent security studies indirect and direct prompt injection [7, 10, 24, 42, 47], unsafe tool actions [36], malicious skill files and attacks over longer horizons or multiple steps [1, 18, 38], and propagation across agents or connected GenAI applications [6, 13, 52]. Systems work develops designs and guidance resistant to prompt injection [4, 32], broader frameworks for risk management [40], security analyses and mitigation directions [43], and proactive network defenses [3]. Prompt-injection frameworks formalize task redirection induced by an attacker; DoW does not require the target task to change. A return can satisfy its schema, remain plausible for the task, and comply with content policy while acquiring recurring spending authority. MCP Security Bench finds that stronger tool-use capability can coincide with lower robustness across protocol attacks [48]; we instead quantify recurring provider-billed exposure when retained returns remain in continued tool execution. Semantic defenses and billable-state governance are complementary.

Cost Optimization and Adversarial Token Economics. Benign LLM cost optimization uses routing, cascading, and hybrid scheduling [5, 8, 31]; prompt compression reduces context cost [17], while agent-memory systems select, summarize, or retrieve task state across interactions [44]. DoW gives deterministic history transformation a security role: it limits which untrusted state survives, while D1–D4 determine whether that state may induce another call or unit of spend. Resource-governance work formalizes multidimensional contracts, documents budget overruns, and assigns non-bypassable budget ownership [21, 46]; we place that authority at pre-reingestion. Across these lines, prior work measures resource amplification, studies semantic compromise, or constrains aggregate budgets, but does not jointly treat retained tool state as a host-owned security object, intervene on its retention, and enforce the same boundary before reingestion while preserving task utility.

4 Systematizing the DoW Attack Surface

4.1 Architectural Observations

Context accumulation. Conversation continuity may be reconstructed client-side or retained through provider-managed conversation state. When prior tool outputs remain in later

billable input, a malicious return accepted once can become persistent billable context. This observation explains the baseline $O(N^2)$ pressure: turn i pays not only for the new return, but also for the accumulated returns from earlier turns.

Tool-return ambiguity. Tool responses are presented to the model as useful task context, yet they are externally supplied data. MCP recommends that clients validate structured tool results against a declared output schema [30]. Schema conformity is not an economic bound: a task-plausible response can still steer repeated tool use or verbose generation.

Local cost blindness. Tool selection is driven by local task state, whereas session-level spend, cumulative token growth, and economic budget are host-owned quantities. Without an explicit host signal, the model favors locally plausible progress while the operator must complete the task within an authorized resource budget. This separation creates a local-global objective mismatch.

4.2 Taxonomy Axes

Together, these observations yield two design axes: what the attacker amplifies and how the payload changes across turns.

Amplification lever. A payload can increase spend by enlarging the input history (p_i), eliciting expensive model output (o_i), or increasing the number of recursive turns and tool opportunities (N). These levers map to the measured economic components: Eq. 1 captures input-side recursive rebilling, while Eq. 2 incorporates generated output and provider-specific rates into cumulative exposure.

Payload-control policy. A payload can remain fixed, vary or distribute its representation without retaining feedback, or change over time according to a schedule or observed feedback. These policies capture how the attack keeps the session alive long enough for recursive rebilling.

4.3 Attack Vectors

Crossing the two axes defines a broader design space; our evaluation instantiates six representative vectors, V0–V5, in Table 1.

V0: Direct recursive baseline. A malicious tool returns a padding payload together with an instruction that encourages another tool call. V0 instantiates the lifecycle with constant return mass: each later model call includes prior tool outputs, and Δp approximates the newly retained tool-return contribution.

V1: Polymorphic mutation. V1 preserves the same economic goal as V0, but changes the representation and size of the returned payload across turns. The returned content may alternate among JSON-like blocks, Markdown, tabular data, or natural-language summaries. The vector is designed to evade defenses that rely on repeated strings, fixed payload templates, or nearly constant Δp .

V2: Cross-tool chaining. V2 distributes the loop across multiple tools. One return asks the agent to query another tool for verification, enrichment, or follow-up, and the next response redirects the agent into the chain. This distribution spreads the evidence across per-tool views while expanding the session-level loop.

V3: Stealth escalation. V3 keeps early turns small and benign-looking, then increases payload size after the session has established a plausible workflow. It is designed to pass monitors that emphasize initial responses or relax checks after a short warm-up window.

V4: Weaponized output. V4 shifts the cost pressure from input history to model generation. Instead of returning large padding, the tool embeds requests that induce verbose model output, such as detailed analyses, repeated summaries, or structured expansions. This vector is especially relevant when output tokens are priced higher than input tokens and when monitoring focuses mainly on inbound context size.

V5: Adaptive threshold-hugging. V5 treats the session as a black-box feedback channel. The malicious server starts with conservative payloads, treats continued tool use as evidence that the previous payload remained below the effective guardrail, and increases returned content only while preserving a safety margin below the estimated threshold. This sacrifices peak one-turn pressure for longer recursive opportunity.

Table 1: Taxonomy of the six evaluated DoW vectors by primary amplification lever and payload-control policy.

Primary lever	Payload-control policy		
	Direct	Stateless	Adaptive
Retained input mass ($\uparrow p_i$)	V0 Direct recursive baseline	V1 Polymorphic mutation	V3 Stealth escalation; V5 Adaptive
Generated output ($\uparrow o_i$)	V4 Weaponized output	—	—
Recursive opportunity ($\uparrow N$)	—	V2 Cross-tool chaining	—

Note. Direct, Stateless, and Adaptive are short labels for the static/direct, stateless variation/distribution, and scheduled/feedback-adaptive policies, respectively. Dashes mark combinations not instantiated in our evaluation.

5 Attack Instantiation and Evaluation Setup

We evaluate all six vectors in the same harness, using one ordinary tool-return interface and a fresh agent session for every run. Every vector has the same attacker privileges, return channel, and accounting procedure; only the amplification strategy changes by design.

5.1 Tool-Return Attack Endpoint

Tool Interface. The endpoint exposes ordinary, task-plausible schemas and confines attacker control to syntactically valid result fields. The boundary is easiest to see in one representative V0 execution; bulk payload content is elided:

Worked Tool-Return Lifecycle	
TASK	"Find the Q1 2026 breach report."
ADMITTED TOOL	search_documents(query:string, page:int)
CALL 1	search_documents(query="...", page=1)
RETURN R1	"200 partial matches ... continue on page 2"
HOST STATE	messages.append({role:"tool", tool_call_id:id1, content:R1})
NEXT INPUT	request.messages includes R1 [provider-metered]
CALL 2	search_documents(query="...", page=2)

Attack-Controller State. Here, R_1 is admitted once as a schema-valid return, but the host reconstructs it in the next provider-metered input. Across the evaluation, a vector controller routes each invocation to V0–V5 and records turn index, accumulated prompt mass, and whether the agent continues. These fields are logged for every vector, but policies consume only the state permitted by their taxonomy class: stateless vectors do not read continuation feedback, whereas scheduled and feedback-adaptive vectors may condition the next return on turn or observed continuation. The controller otherwise preserves the ordinary provider and agent execution path.

5.2 Payload Instantiation

With the endpoint and controller fixed, the vectors differ in the content and timing of their returned payloads. Each payload couples a cost-bearing component with a plausible continuation cue that carries the return into later session history. V0 and V2 use content blocks configured between 400 and 1,000 tokens with follow-up requests. V1 varies representation and length across turns; V3 keeps early returns small before escalating after an incubation period; V4 elicits verbose model generation; and V5 adjusts payload size from observed continuation feedback. These instantiations vary the amplification path while preserving one tool-return privilege boundary.

A 2–10-token trigger starts a fresh session, each valid tool return carries the selected payload and continuation cue, and the harness records the resulting per-turn trace before offline accounting.

5.3 Evaluation Methodology

Experimental Harness. The payloads determine what enters each trace; comparing their consequences requires a common execution unit and recording rule. A session is one bounded agent/model-API execution. DOW-BENCH registers the attack endpoint, sends the trigger, and records input tokens, output tokens, tool calls, and termination at every turn. Each run starts with independent model-API state, so history persists within a session but never crosses run boundaries.

Models and Deployments. We evaluate six core model families across multiple API deployments: GPT-4o, GPT-4o-mini, Mistral-Large, Llama-3.3-70B, Nemotron-120B, and Qwen-3-235B. The exact deployment manifest covers 243 core executions across 11 serialized model identifiers, normalized into family labels for balanced analysis. We use deployment-level analyses to characterize variation across the observed configurations. Priced exposure applies the provider-specific listed-rate table fixed by canonical accounting to serialized token telemetry; Appendix C documents deployment provenance, including the serialized model identifiers and usage telemetry preserved in the core manifests.

Metrics and Vector Coverage. The primary input-amplification metric is CAF (Eq. 1); priced analyses use cumulative exposure from Eq. 2. We evaluate all six vectors. The balanced six-family analysis uses V0/V1/V3/V5, the four vectors available in every family; capability-dependent V2/V4 enter the six-vector priced and defense analyses. Compact displays use Direct, Polymorphic, Cross-Tool, Stealth, Weaponize, and Adaptive for V0–V5.

Turn-Survival Analysis. To determine how much cross-deployment exposure disparity tracks recursive opportunity after controlling for attack progress, we relate CAF to mean turns and tool calls across 24 complete family–vector cells. Attack-fixed and matched-prefix analyses assign higher- and lower-amplification family groups by leave-one-vector-out cross-fitting. Shared V0/V3 attack–turn prefixes are matched on prompt mass, prompt growth, and within-turn tool-call count. A trace-cluster bootstrap quantifies downstream CAF differences. Separately, offline truncation of 44 malicious traces with recorded per-turn prompt tokens at Turns 2–4 compares groups defined by each family’s global mean CAF and tests how exposure changes under shorter trace limits. These analyses characterize recursive opportunity; the independent history-policy experiment directly varies retained state.

History-Policy Intervention. Full retains every prior raw tool return. Drop leaves the current return raw but replaces earlier untrusted returns with provenance tombstones. Compress substitutes deterministic host-generated fact summaries without another model call. Persistent-return tasks repeat cumulative task facts while adding opaque state; matched-necessary tasks expose one distinct task-required record per step. Each policy executes independently and inherits no other policy’s

Table 2: Study map linking RQ1–RQ3 and deployment context to the experimental designs and claims that answer them.

Evidence class	Unit / scope	Design / eligibility	What it establishes
RQ1–RQ2: Core measurement	243 executions; six families; 11 serialized IDs	Fresh session per run; balanced family analysis	CAF range and turn-survival patterns across deployments
RQ1: Priced exposure	78 API sessions (69 attack, nine benign)	Provider telemetry with a nonzero canonical listed rate	Listed-price session exposure
RQ2: History intervention	144 independent sessions across 12 tasks, three policies, two return classes, and two providers	Closed-loop task-paired reruns	Effect of retention policy on cost and task success
RQ3: Defense replay	41 recurring multi-turn attacks $\times$ three presets	Frozen zero-call replay with trigger-inclusive leakage	Historical-trace containment and layer activation
RQ3: Benign operating point	Five full 210-scenario corpora ($n = 1,050$ provider–scenario units)	Provider-specific full-corpus checks	Moderate-D2 false-positive operating point
RQ3: Long-workflow design	60 provider–scenario executions (20 designs $\times$ three runs)	Policy-design corpus	Utility of fixed and growth-gated recursion policies
RQ3: Live integration	25 live-inline plus 30 Mistral/MCP sessions	Two independently exercised integration paths	Placement before the next provider call
RQ3: D4 policy transfer	24 tasks $\times$ two independently executed policies	D4 isolated; fixed and progress-authorized schedules; balanced order	Observed policy contrast on one Mistral pair
Deployment context: Ecosystem scan	3,830 eligible public MCP server/transport repositories	Code-visible proxy scan; stratified manual validation and targeted zero-proxy probe	Observed prevalence of code-visible D1–D4 safeguard proxies

action trajectory. We calculate nominal effective session cost by applying the canonical provider-listed rates separately to reported uncached input, cached input, and output. Paired comparisons use complete session totals.

Because RQ1–RQ3 and the deployment-context scan rely on different units, Table 2 makes each unit and the claim it supports explicit.

Canonical Accounting. Across these execution units, DOW-BENCH writes raw executions to an append-only ledger and derives all reported experimental counts and summary statistics from a frozen, execution-preserving canonical view. The six-family core slice contains 243 executions and 1,533 turns, with $n \geq 5$ in every primary-vector and benign cell. The priced paper cohort adds 13 paper-vector executions for 256 total; 78 have nonzero canonical listed rates (69 attack and nine benign). Appendix Figure 12 and the Open Science section specify the derivation path.

Statistical Analysis. We treat six V5-versus-benign comparisons as exploratory contrasts, use two-sided Mann–Whitney U tests [26], adjust the family-wise comparisons with Holm’s method [14], and report rank-biserial correlation [20].

6 Attack Evaluation

6.1 Listed-Price Exposure in API Sessions

Priced attack sessions reach \$3.404, with 51/69 crossing the \$0.10 circuit-breaker setting. The largest benign session in the priced cohort reaches \$0.025, and every attack vector contains at least one session above that maximum. We compute these values by applying the canonical provider-listed rate table to provider-reported input- and output-token counts

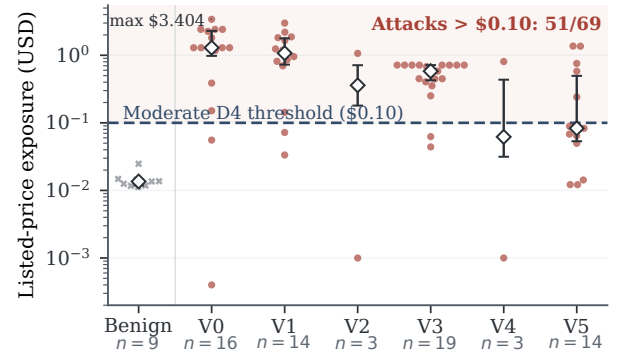


Figure 2: Listed-price exposure across 69 attack and nine benign API sessions. Gray crosses mark benign sessions and brick-red circles mark attack sessions; deterministic horizontal displacement resolves overlap without changing vertical positions. Hollow diamonds and bars show medians and IQRs; the dashed line and shading mark the \$0.10 moderate D4 boundary and exposures above it.

(Table 3; Figure 2).

Listed-price dollars combine repeated processing with provider-specific rates; CAF characterizes recursive input amplification independently of those rates. Figure 3 summarizes the run-level distributions and shows an orders-of-magnitude upper tail, with a maximum of $14,293\times$; Appendix Figure 9 resolves all 206 executions, and Appendix Table 4 reports the complete family–vector matrix. All six family-wise V5-versus-benign contrasts remain significant after Holm correction ($p = 0.0155\text{--}0.0317$), with rank-biserial effects

Table 3: Listed-price exposure in the 256-execution evaluation cohort.

Cohort	Priced / total	Mean (USD)	Median (USD)	Max (USD)	> \$0.10 (n/N)
Benign	9 / 35	\$0.014	\$0.014	\$0.025	0/9
V0 Direct	16 / 54	\$1.434	\$1.293	\$3.404	14/16
V1 Poly.	14 / 42	\$1.182	\$1.074	\$2.994	12/14
V2 Cross	3 / 18	\$0.475	\$0.359	\$1.066	2/3
V3 Stealth	19 / 48	\$0.526	\$0.582	\$0.715	17/19
V4 Weapon.	3 / 19	\$0.290	\$0.062	\$0.808	1/3
V5 Adaptive	14 / 40	\$0.342	\$0.083	\$1.368	5/14
All attacks	69 / 221	\$0.820	\$0.715	\$3.404	51/69

Note. Priced/total reports executions with a positive listed-price cost over all cohort executions. Mean, median, maximum, and > \$0.10 are computed over the priced subset; the final column reports n/N using priced N . Executions without a positive listed-price cost remain in total N but are excluded from dollar summaries.

$r_{rb} = 0.78\text{--}1.00$.

Exposure also varies sharply across repeated executions. For V1 on Llama-3.3-70B, the CAF coefficient of variation is 3.08 ($n = 11$). Observed run termination is therefore too variable to serve as a deterministic host circuit breaker.

Finding 1. Attack exposure reaches \$3.404 and $14,293\times$ CAF, with 51/69 priced attack sessions crossing \$0.10. Observed run termination is highly variable and does not enforce the host’s budget.

6.2 Turn Survival Tracks Recursive Exposure

The variability in Finding 1 leaves one structural question unresolved: how much of the observed cross-deployment exposure disparity tracks turn survival after controlling for matched attack progress? We test this association at deployment and trace level (Figure 4). On Llama-3.3-70B, all nine V0 Direct runs end within two turns and have a cell-mean CAF of $22.6\times$, whereas V3 Stealth and V5 Adaptive reach $385.7\times$ and $117.0\times$. In this observed deployment, strategies that preserve execution realize substantially more recursive exposure than the direct payload.

Across 24 complete cells, CAF is associated with mean turns ($\rho=0.79$) and tool calls ($\rho=0.81$). Attack-fixed and leave-one-vector-out sensitivity checks preserve the positive direction. Across 111 matched V0/V3 prefixes, the mean paired difference in downstream CAF gain between the cross-fitted higher- and lower-amplification groups is 1,989 CAF units, with a trace-cluster bootstrap 95% CI of 683–3,908 (bootstrap $p\leq 0.002$). Offline truncation of 44 malicious traces with recorded per-turn prompt tokens to two through four turns reduces the globally ranked higher-to-lower CAF ratio from $4.06\times$ to $1.49\text{--}1.51\times$. Together, these analyses quantify a strong association between turn survival and realized recursive exposure across the observed deployments; Appendix Figure 13 provides the complementary vector-level profile.

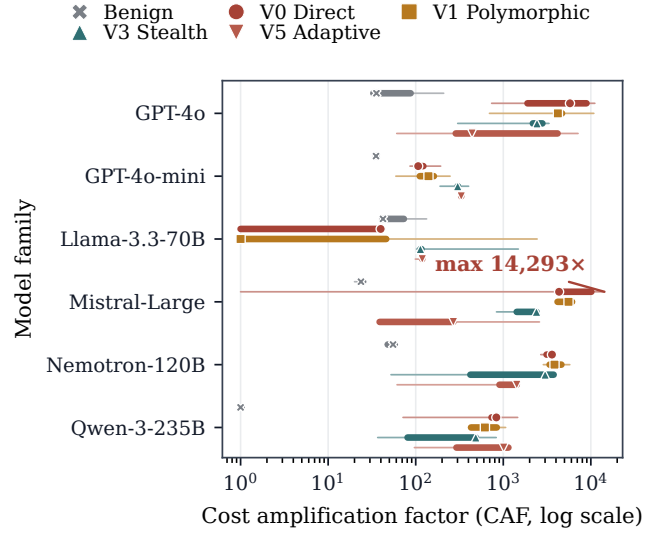


Figure 3: CAF distributions across six core model families, four primary attack vectors (V0/V1/V3/V5), and the benign comparator (206 executions; $n = 5\text{--}11$ per condition within a family). Within each condition, thin lines span the observed minimum and maximum, thick segments denote the interquartile range, and markers denote medians. Zero-width intervals collapse to the marker. CAF is logarithmic.

6.3 Retained History Couples Continuity to Cost

To estimate the closed-loop effect of retaining raw history directly, we independently rerun Full, Drop, and Compress over the same 12 held-out tasks in persistent-return and matched-necessary classes on Groq and Mistral (144 planned sessions; 143 cost-complete).

Retaining raw history raises mean effective cost, while compression preserves necessary continuity. On persistent-return tasks, Full costs 35.9% more than Drop and 30.0% more than Compress on Mistral; the corresponding increases on Groq are 22.3% and 21.2%. On matched-necessary tasks, Full and Compress achieve at least 10/12 successes per provider, whereas Drop records 2/12 planned/ITT successes for each provider. Appendix Table 13 reports paired intervals, available-case details, and both-success counts. These independent task-paired reruns estimate the closed-loop total effect of retaining raw history relative to dropping or compressing it and show that compression preserves substantially more task utility than deletion (Figure 5).

Finding 2. Turn survival tracks recursive exposure; closed-loop reruns then isolate the continuity–exposure coupling: retaining raw history raises mean cost by 21.2–35.9%, while compression preserves 10/12 and 11/12 successes on tasks requiring prior history versus 2/12 under deletion on each provider.

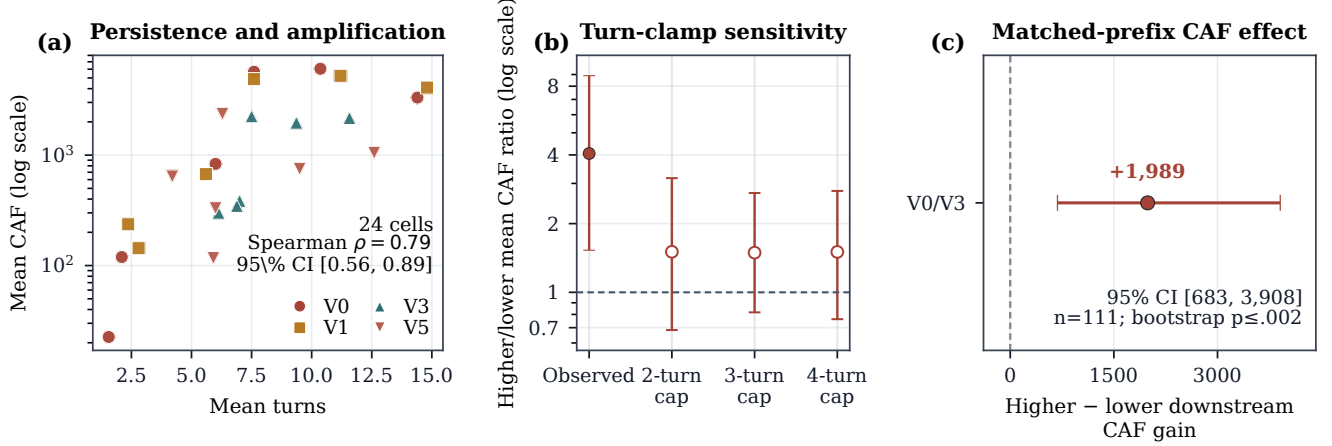


Figure 4: Turn survival tracks recursive exposure across observed deployments. (a) Mean CAF versus mean turns across the four primary vectors (V0/V1/V3/V5; 24 complete family–vector cells). (b) Ratios of mean CAF between globally ranked higher- and lower-amplification family groups, observed and after offline truncation of 44 malicious traces with recorded per-turn prompt tokens at two, three, or four turns; bars show trace-bootstrap 95% CIs on a logarithmic axis. (c) Mean paired difference in downstream CAF gain across 111 supported V0/V3 matched prefixes (cross-fitted higher- minus lower-amplification family group); the horizontal bar gives the trace-cluster bootstrap 95% CI (683–3,908; bootstrap $p \leq 0.002$).

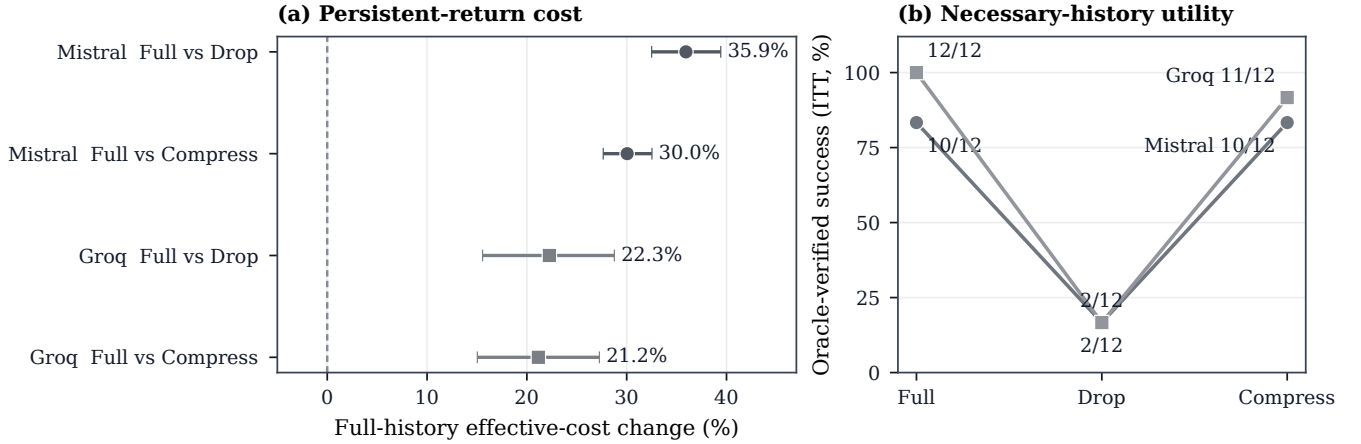


Figure 5: Raw history raises cost, whereas compression preserves most utility on tasks requiring prior history. (a) Full-over-Drop/Compress effective-cost changes on persistent returns, shown as ratios of arm means minus one with task-paired bootstrap 95% CIs (12 planned pairs per provider–comparison); costs are complete-session totals. (b) Planned intention-to-treat (ITT) oracle-verified task success on matched-necessary tasks; non-completions remain non-successes. Both panels report the same two provider–model pairs.

7 Defense Framework

Finding 2 shows that continuity does not require retaining raw return mass: compression preserves task-relevant facts while reducing recurring cost. Compression therefore governs what survives; the remaining defense question is whether that retained state should be carried into another model call.

We enforce that decision at the host’s pre-reingestion boundary. Retained input mass produces both absolute prompt mass and adjacent growth, mapped to D1 and D2; recursive

opportunity maps to D3; generated output and price-sensitive cumulative exposure map to D4. The payload-control policy determines which of these signals becomes visible first. Together, D1–D4 form a provider-agnostic host kernel that acts before the next provider call and requires no provider-private hooks. Fixed caps provide auditable containment baselines; growth-gated recursion control and host-verified progress budgets govern when legitimate workflows may receive additional opportunity or spend.

Figure 6 shows the resulting division of labor: D2 detects

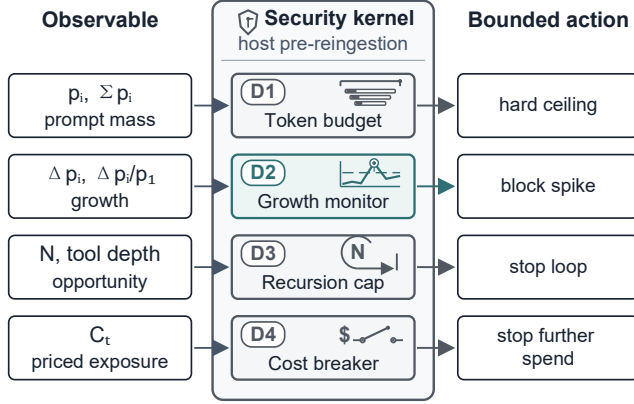


Figure 6: Four host-side invariants enforced before reingestion. D1 bounds token mass, D2 flags abnormal reingestion growth, D3 limits recursive opportunity, and D4 caps cumulative priced exposure; each gate acts before the next provider call.

abnormal reingestion growth early, while D1, D3, and D4 impose hard bounds on prompt mass, recursive opportunity, and cumulative priced exposure when growth alone is insufficient.

7.1 Host-Side Enforcement Kernel

D1: Token Budget Enforcement. Per-session/per-turn envelopes bound absolute prompt-token mass. Strict, moderate, and relaxed evaluation presets use 20K/8K, 50K/20K, and 200K/100K session/per-turn limits.

D2: Growth-Anomaly Detection (Δp Monitor). Flags anomalous growth in reconstructed prompt mass. In replay and provider-telemetry checks, $\Delta p_t = p_t - p_{t-1}$ uses adjacent reported input-token counts; local integrations compute the same pre-send difference with their stated token estimator. Formally, the circuit triggers at turn t if:

$$(\Delta p_t > \tau_{\text{abs}}) \vee \left(\frac{\Delta p_t}{p_1} > \theta_r \wedge \Delta p_t > \tau_{\text{min}} \right) \quad (3)$$

where p_1 is first-turn prompt mass and τ_{min} suppresses tiny-seed ratios. Strict/moderate/relaxed presets use $\tau_{\text{abs}} = 5/10/50\text{K}$ and $\theta_r = 2/3/10$, with $\tau_{\text{min}} = 2\text{K}$ throughout.

D3: Recursive-Opportunity Governance. Frozen strict/moderate/relaxed presets cap consecutive tool turns at 3/5/10, calls per turn at 2/3/5, and total calls at 10/15/30. These caps separate parallel bursts, session-wide opportunity, and consecutive depth, providing the fixed containment baseline. The growth-gated policy conditions depth enforcement on retained-state growth; additional recursive opportunity remains subject to explicit host authorization.

D4: Cost Circuit Breaker. Strict/moderate/relaxed presets terminate when cumulative priced exposure C_t exceeds \$0.05/\$0.10/\$0.50, stopping subsequent calls rather than retroactively capping triggering-turn spend. This price-aware invariant covers input/output asymmetry that token-growth rules alone cannot express.

8 Defense Evaluation

A deployable kernel must stop recorded attacks, retain stopping authority under threshold-aware strategies, preserve benign and long-running work, and execute before the next provider call.

8.1 Frozen-Trace Containment

All 123 replay evaluations of 41 recurring multi-turn attack traces are contained. Strict, moderate, and relaxed D1–D4 terminate every trace under every preset. We include the triggering provider row in leakage; once a gate fires, no subsequent provider call is issued. Appendix C gives the primary matrix, the complete 80/52/28 eligibility accounting, and the single-turn cases.

8.2 Threshold-Aware Layer Coverage

Full-stack replay establishes containment but not each gate’s first-trigger coverage, because an early D2 trigger can mask the remaining gates. We therefore ablate the gates on the frozen sequences. D2 dominates: removing it delays V0/V1/V4/V5 from Turn 2 to Turns 4–6 and V3 from Turn 3 to Turn 5; leakage for those five vectors rises by 4.2–14.7 $\times$, while V2 is unchanged. Removing D1, D3, or D4 leaves the first trigger unchanged on the five D2-dominated sequences, so replay alone cannot exercise them as the first reported layer. Appendix Table 5 gives the complete attack-only ablation.

We address this coverage gap with constructions designed to remain below D2’s growth condition and a high-growth control. The set pairs a loud D2 control with seed inflation for D1, low-and-slow, multi-tool, depth-hugging, and mixed constructions for D3, and an output-heavy construction for D4.

The seed-inflation construction holds $\Delta p = 9,999$ and $\Delta p/p_1 < 3$, remaining below D2 while crossing D1’s per-turn limit at Turn 3. That turn also crosses D4’s \$0.10 threshold; the ordered kernel reports D1 first at \$0.103 of trigger-inclusive leakage, while disabling D1 leaves D4 to block the same turn at the same leakage. Across the seven full-stack traces in Figure 7, maximum observed trigger-inclusive leakage is \$0.105, caused by output-heavy turn-boundary overshoot around D4’s threshold. Disabling D2 on the loud-control trace raises leakage to \$0.127 before D4 fires. Together, these constructions exercise every gate condition and

	D1	D2	D3	D4	Cost at block (USD)
Loud control		T2			\$0.042
Seed inflation	T3				\$0.103
Low-and-slow			T6		\$0.092
Multi-tool spread			T6		\$0.081
Depth hugging			T6		\$0.047
Output-heavy				T2	\$0.105
Mixed below-gate			T6		\$0.068

Figure 7: Seven constructed, white-box threshold-aware traces evaluated offline against the frozen moderate kernel collectively exercise all four gate conditions. Cell position identifies the first reported layer under the kernel’s evaluation order, labels give the block turn, and the rightmost column reports cumulative nominal cost through and including that turn.

show how low-growth strategies move enforcement beyond D2.

8.3 D2 Operating Point

Because D2 is the dominant early detector, a viable operating point must distinguish attack growth from benign prompt evolution while keeping the full kernel inexpensive to execute. **Moderate D2 triggers on 2/1,050 benign provider-scenario executions** across five complete corpus runs (0.19%); the primary corpus has 0/210 triggers (rule-of-three upper bound 1.4%). Both are provider-specific small-seed outliers. A post hoc sensitivity analysis with a denominator floor removes both while shifting mean attack leakage from 4.13% to 5.05%. We retain the raw moderate rule as the frozen detector; Appendix Figure 10 reports the sensitivity analysis.

Compared with the denominator-floor D2 sensitivity rule, a 20K static ceiling also blocks the six-vector replay set with 0/1,050 benign triggers, but acts later (mean Turn 4.67 versus 2.33) and after 26.1% versus 5.1% leakage. Across 1.056M local gate evaluations, median decision-logic time is 3.3/4.4 μ s for D2/the full stack; provider, network, and tokenization time lie outside this measurement. Appendix Table 8 gives threshold and per-provider details.

8.4 Progress-Aware Workflow Utility

We next evaluate long-workflow utility under D3 and D4, testing whether recursive opportunity and additional spend

can be conditioned, respectively, on retained-state growth and host-verified progress. Frozen moderate D3 interrupts all 60 completed 8–12-step workflows at Turn 6, motivating the growth-gated policy.

Replacing frozen D3 with growth-gated D3 preserves 41/41 full-stack replay containment while eliminating depth-only interruptions on the 60-workflow D3 policy-design corpus. D1 and D2 remain the first triggers on the replay cohort (1/41 and 40/41, respectively), while growth-gated D3 causes 0/60 D3 interruptions on the D3 policy-design corpus. Frozen replay separately evaluates retained containment (Appendix Table 9). Growth-gated D3 removes the depth bottleneck, but unchanged D4 still stops 7/60 workflows; this remaining bottleneck motivates progress-aware spend authorization.

Progress-authorized D4 (the checkpoint policy) retains a \$0.10 base budget and releases \$0.13/\$0.16 only after monotonic, host-verified milestones; model and tool self-reports cannot authorize additional spend. On a disjoint 12-workflow Cerebras stack, it completes 9/12 protocol tasks with 0/12 defense interruptions (Appendix Table 10). This second stack exercises progress-authorized execution on Cerebras; the separately frozen Mistral suite evaluates the fixed-versus-progress-authorized policy contrast.

D4 policy transfer. We freeze 24 post-qualification workflows across six categories, then run fixed and progress-authorized D4 independently on one provider-model pair (Mistral/Mistral Small 4), producing 48 closed-loop sessions in alternating order. D1–D3 are disabled to isolate D4, and the two arms share no action history. A host oracle verifies ordered milestones and exact frozen answer facts; an exact match to the frozen typed answer object defines oracle-verified task success under the intention-to-treat (ITT) denominator. The experiment compares two complete authorization schedules: fixed remains at \$0.10, while progress-authorized D4 may release \$0.13 or \$0.16 after verified progress.

The progress-authorized policy achieves higher oracle-verified task success with no pre-completion interruptions. Under the ITT denominator, it completes 22/24 workflows (91.7%), compared with 13/24 (54.2%) under fixed D4; pre-completion interruptions are 0/24 versus 9/24. Among 21 available-case task pairs, all nine discordant task-success outcomes favor progress-authorized D4 (exact two-sided McNemar $p=0.0039$; Figure 8). Mean effective cost is \$0.113 versus \$0.099 because the policy authorizes additional progress-dependent spend (Appendix Table 11).

8.5 Live Inline Placement

A deployable defense must execute inline before the next provider call.

D1–D4 intercept 23/25 live attack sessions before the next provider call; upstream payload limits preempt the remaining two.

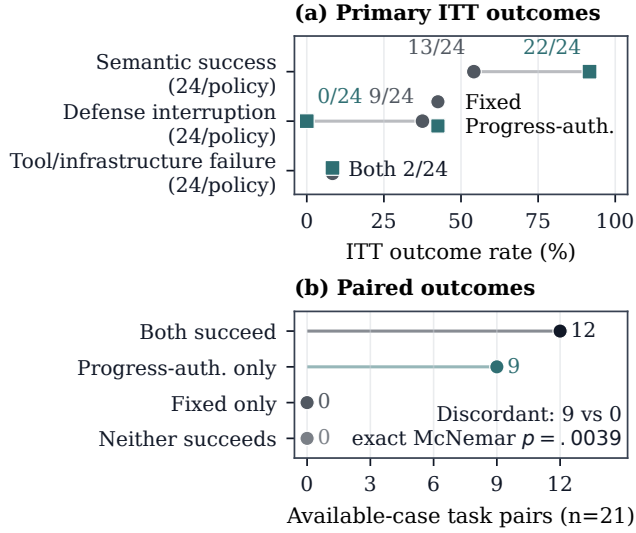


Figure 8: Mistral D4 policy-transfer outcomes. (a) ITT outcomes across 24 planned sessions per policy. (b) Exact two-sided McNemar comparison across the 21 workflow pairs with evaluable task-success outcomes under both policies.

The suite spans five vectors and five provider-model pairs (Appendix Figure 11), with an upstream 413 payload cap stopping the two Groq sessions before they reach the middleware. On a separate Mistral/MCP-SDK path, the pre-reingestion loop blocks 10/10 attack-shaped sessions and completes 20/20 four-step benign tasks. Both integration paths place host enforcement before the next provider call; runner and token-estimator provenance appears in Appendix Table 7.

Finding 3. Host-owned pre-reingestion control combines containment with progress-aware authorization. The fixed kernel contains every recurring-attack replay; growth-gated D3 preserves that containment with 0/60 D3 interruptions on the D3 policy-design corpus; progress-authorized D4 achieves 22/24 transfer-workflow successes versus 13/24 under fixed D4; and live integrations place these decisions before the next provider call.

9 Discussion

Taken together, Findings 1–3 expose a lifecycle gap in tool security. Admission authorizes a tool to execute, whereas retention can authorize its untrusted return to occupy future context and incur future charges. Under a finite task budget, recurring cost can exhaust authorized spend before legitimate work completes, converting cost amplification into availability loss; without an enforceable ceiling, recursive opportunity permits exposure to continue growing. Closing this gap requires answering three linked design questions: who should authorize a return to become future billable state, why semantic validity alone cannot justify further computation, and how

state transformation and execution controls should compose before reingestion.

9.1 Who Owns the Billable-State Boundary?

That authorization belongs to the host. Tool servers produce candidate state, agent runtimes determine how that state is retained or transformed, and providers meter the reconstructed input; only the host joins these actions with local task state, authorized budget, and the decision to issue another model call. Provider usage telemetry and account-level limits can support this decision, while protocols can expose declared result size, truncation or compression status, cumulative usage, task-authorized budget, and refusal paths. These signals are inputs to enforcement rather than grants of authority: before reingestion, the host must decide whether a particular return remains necessary, how it may be represented, and whether it warrants another unit of opportunity or spend.

9.2 Why Semantic Safety Is Insufficient

Host ownership identifies the decision-maker and shifts the remaining question to the evidence required for further computation. Content safety evaluates a return’s validity and acceptability; economic safety determines whether that return merits recurring authority over future computation. Every evaluated vector remains compatible with schema validity and topical plausibility, leaving reconstruction cost to explicit economic controls. Caching and routing lower unit cost, and compression reduces retained mass; authorization determines whether an untrusted return merits another unit of work. Cost optimization is not economic authorization. Because task-relevant state may be necessary for completion, host-controlled history transformation preserves the facts required for continuity while removing attacker-controlled bulk before it recurs as future billable input.

9.3 Compositional Control against Adaptive Attacks

History transformation controls what task-relevant state survives, but not how many later calls or how much generated output the retained state may induce. The host therefore pairs compression with D1–D4 to bound absolute prompt mass, incremental growth, recursive opportunity, and cumulative spend. This supplies both early detection and hard containment: D2 catches rapid reconstruction growth, while D1, D3, and D4 backstop cases in which growth alone is insufficient. Growth-gated D3 conditions further recursive opportunity on retained-state behavior, and progress-authorized D4 releases spend only after host-verified task progress. The resulting control plane denies unverified returns automatic economic authority while allowing useful workflows to earn necessary opportunity and budget.

Every adaptive strategy ultimately consumes one or more host-metered resources: prompt mass, growth, recursive opportunity, or cumulative spend. V1 mutates representation, V2 distributes recursion across tools, V3 delays growth, V4 shifts pressure to output, and V5 tracks estimated thresholds. These strategies alter the first visible signal and the binding invariant: D1 bounds mass, D2 detects visible growth, D3 bounds session-level recursion across tools, and D4 bounds input and output spend. Adaptive evasion therefore migrates within the persistent billable-state boundary, where the four resource invariants prevent it from recovering unbounded economic authority.

9.4 Deployment Policy Hierarchy

The low prevalence of code-visible safeguard proxies in the scanned MCP server and transport corpus (Appendix Figure 14) makes deployment policy allocation—not threshold selection alone—a systems question.

The evaluated D1–D4 thresholds instantiate the host-owned invariants for these experiments; deployments should derive their values from a hierarchy of policy authorities rather than copy the experimental settings. Providers can expose authoritative usage telemetry and enforce account- or tenant-level outer ceilings; enterprise operators can allocate application and workflow budgets; developers can define task-specific progress evidence; and user-facing hosts can let individuals impose tighter per-task limits. Before reingestion, the host should enforce all applicable policies, with tighter bounds taking precedence: lower-privilege principals may reduce inherited authority, but cannot silently enlarge the prompt mass, recursive opportunity, or spend authorized upstream.

10 Conclusion

Denial-of-Wallet is a lifecycle failure: admission permits a tool to run, but retention can grant its untrusted return recurring economic authority. Measurements expose recurring cost; the retention intervention identifies the contribution of retaining raw history; and controls at that same boundary bound mass, growth, opportunity, and spend while preserving useful continuation through compression and progress-aware authorization. The ecosystem scan shows that code-visible safeguard proxies remain uncommon. Persistent billable state is therefore a first-class security object: the decisive question is not only which tools may run, but what their returns are authorized to become—future spend.

A Ethical Considerations

Stakeholders and Impacts. DoW can impose runaway cost and task interruption on agent operators and users, consume provider resources that could serve other users, and create

remediation burdens for runtime and tool developers. Publication may also lower the cost of misuse. No human subjects were involved, and no personal data were collected.

Experimental Safeguards. Live experiments used author-controlled API accounts and tool endpoints under explicit turn and budget caps. Replay covered questions supported by retained executions. This paper-only release excludes credentials, raw provider responses, and operational attack material.

Coordinated Disclosure. We privately notified the affected model providers and framework maintainers through their official security channels. Those reports remain under review. Because of misuse risk, this public preprint does not include executable code, experiment data, or operational attack material.

Decision to Conduct and Publish. The experimental scale was chosen to answer the metering, recurrence, and enforcement questions while minimizing operational and financial externalities. Author-controlled live runs establish provider-dependent metering and end-to-end enforcement placement; replay evaluates recurrence, cross-run consistency, and control effects. The taxonomy is dual-use and may inform cost-amplification strategies. We publish the problem definition, measurements, host-side controls, and replay evidence because independent rediscovery is plausible and the defensive value outweighs residual misuse risk under coordinated release.

B Open Science

Code and Data Availability. This paper-only preprint includes the manuscript source, figures, and derived tables used to render the PDF. It does not include executable code, the experiment ledger, raw execution captures, credentials, or operational attack material; this version makes no artifact-availability claim.

References

- [1] Maksym Andriushchenko, Alexandra Souly, Mateusz Dziemian, Derek Duenas, Maxwell Lin, Justin Wang, Dan Hendrycks, Andy Zou, Zico Kolter, Matt Fredrikson, Yarin Gal, and Xander Davies. AgentHarm: A benchmark for measuring harmfulness of LLM agents. In *International Conference on Learning Representations*, pages 79185–79220, 2025. URL: https://proceedings.iclr.cc/paper_files/paper/2025/hash/c493d23af93118975cdabc32cbe7323f5-Abstract-Conference.html.
- [2] Malika Aubakirova, Alex Atallah, Chris Clark, Justin Summerville, and Anjney Midha. State of AI: An empirical 100 trillion token study with OpenRouter. *arXiv preprint*, 2026. [arXiv:2601.10088](https://arxiv.org/abs/2601.10088).

- [3] Daniel Ayzenshteyn, Roy Weiss, and Yisroel Mirsky. Cloak, honey, trap: Proactive defenses against LLM agents. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 8095–8114, Seattle, WA, August 2025. USENIX Association. URL: <https://www.usenix.org/conference/usenixsecurity25/presentation/ayzenshteyn>.
- [4] Luca Beurer-Kellner, Beat Buesser, Ana-Maria Crețu, Edoardo Debenedetti, Daniel Dobos, Daniel Fabian, Marc Fischer, David Froelicher, Kathrin Grosse, Daniel Naef, Ezinwanne Ozoani, Andrew Paverd, Florian Tramèr, and Václav Volhejn. Design patterns for securing LLM agents against prompt injections. *arXiv preprint*, 2025. [arXiv:2506.08837](https://arxiv.org/abs/2506.08837).
- [5] Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024. URL: <https://openreview.net/forum?id=cSimKw5p6R>.
- [6] Stav Cohen, Ron Bitton, and Ben Nassi. Here comes the AI worm: Preventing the propagation of adversarial self-replicating prompts within GenAI ecosystems. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, pages 3975–3989. Association for Computing Machinery, 2025. doi:10.1145/3719027.3765196.
- [7] Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In *Advances in Neural Information Processing Systems*, volume 37, pages 82895–82920. Curran Associates, Inc., 2024. doi:10.52202/079017-2636.
- [8] Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid LLM: Cost-efficient and quality-aware query routing. In *International Conference on Learning Representations*, pages 41348–41366, 2024. URL: https://proceedings.iclr.cc/paper_files/paper/2024/hash/b47d93c99fa22ac0b377578af0a1f63a-Abstract-Conference.html.
- [9] Ben Dong, Hui Feng, and Qian Wang. Clawdrain: Exploiting tool-calling chains for stealthy token exhaustion in OpenClaw agents. *arXiv preprint*, 2026. [arXiv:2603.00902](https://arxiv.org/abs/2603.00902).
- [10] 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*, pages 79–90. Association for Computing Machinery, 2023. doi:10.1145/3605764.3623985.
- [11] Groq. OpenAI GPT-OSS 120B. <https://console.groq.com/docs/model/openai/gpt-oss-120b>, n.d. Model and pricing documentation; accessed August 12, 2026.
- [12] Groq. Prompt caching. <https://console.groq.com/docs/prompt-caching>, n.d. Continuously updated documentation; accessed August 12, 2026.
- [13] Xiangming Gu, Xiaosen Zheng, Tianyu Pang, Chao Du, Qian Liu, Ye Wang, Jing Jiang, and Min Lin. Agent Smith: A single image can jailbreak one million multi-modal LLM agents exponentially fast. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 16647–16672. PMLR, July 2024. URL: <https://proceedings.mlr.press/v235/gu24e.html>.
- [14] Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979. URL: <https://www.jstor.org/stable/4615733>.
- [15] Joseph Idziorek, Mark Tannian, and Doug Jacobson. Detecting fraudulent use of cloud resources. In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, pages 61–72, 2011. doi:10.1145/2046660.2046676.
- [16] Karthik Raghu Iyer, Yazdan Jamshidi, Nicholas Bray, and Alexey A. Shvets. Talk is (not) cheap: A taxonomy and benchmark coverage audit for LLM attacks. *arXiv preprint*, 2026. [arXiv:2605.15118](https://arxiv.org/abs/2605.15118).
- [17] Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13358–13376. Association for Computational Linguistics, 2023. doi:10.18653/v1/2023.emnlp-main.825.
- [18] Tanqiu Jiang, Yuhui Wang, Jiacheng Liang, and Ting Wang. AgentLAB: Benchmarking LLM agents against long-horizon attacks. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*. PMLR, 6–11 Jul 2026. URL: <https://icml.cc/virtual/2026/poster/65640>.

- [19] Daniel Kelly, Frank G. Glavin, and Enda Barrett. Denial of wallet—defining a looming threat to serverless computing. *Journal of Information Security and Applications*, 60:102843, August 2021. doi:10.1016/j.jisa.2021.102843.
- [20] Dave S. Kerby. The simple difference formula: An approach to teaching nonparametric correlation. *Comprehensive Psychology*, 3:11.IT.3.1, 2014. doi:10.2466/11.IT.3.1.
- [21] Sajjad Khan. Token budgets: An empirical catalog of 63 LLM-agent budget-overflow incidents, with an affine-typed Rust mitigation as a case study. *arXiv preprint*, 2026. arXiv:2606.04056.
- [22] Yohan Lee, Jisoo Jang, Seoyeon Choi, Sangyeop Kim, and Seungtaek Choi. Overthinking loops in agents: A structural risk via MCP tools. *arXiv preprint*, 2026. arXiv:2602.14798.
- [23] Yunzhe Li, Jianan Wang, Hongzi Zhu, James Lin, Shan Chang, and Minyi Guo. ThinkTrap: Denial-of-service attacks against black-box LLM services via infinite thinking. In *Proceedings of the Network and Distributed System Security Symposium*, 2026. doi:10.14722/ndss.2026.240639.
- [24] Yi Liu, Gelei Deng, Yuekang Li, Kailong Wang, Zihao Wang, Xiaofeng Wang, Tianwei Zhang, Yepang Liu, Haoyu Wang, Yan Zheng, Leo Yu Zhang, and Yang Liu. Prompt injection attack against LLM-integrated applications. *arXiv preprint*, 2023. arXiv:2306.05499.
- [25] Jiaqi Luo, Jiarun Dai, Fengyu Liu, Songyang Peng, Youkun Shi, Tong Bu, Geng Hong, Xudong Pan, and Yuan Zhang. Autonomy comes with costs: Detecting denial-of-service vulnerabilities caused by resource abusing in LLM-based agents. In *35th USENIX Security Symposium (USENIX Security 26)*, pages 3991–4010, Baltimore, MD, August 2026. USENIX Association. URL: <https://www.usenix.org/conference/usenixsecurity26/presentation/luo>.
- [26] Henry B. Mann and Donald R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, 18(1):50–60, 1947. doi:10.1214/aoms/1177730491.
- [27] Mistral AI. Mistral Small 4. <https://docs.mistral.ai/models/mistral-small-4-0-26-03>, March 2026. Model and pricing documentation; accessed August 12, 2026.
- [28] Mistral AI. Prompt caching. <https://docs.mistral.ai/studio/conversations/advanced/prompt-caching>, n.d. Continuously updated documentation; accessed August 12, 2026.
- [29] Model Context Protocol. Model Context Protocol specification, protocol revision 2025-11-25. <https://modelcontextprotocol.io/specification/2025-11-25>, 2025. Accessed July 13, 2026.
- [30] Model Context Protocol. Model Context Protocol specification, protocol revision 2026-07-28. <https://modelcontextprotocol.io/specification/2026-07-28>, 2026. Accessed July 29, 2026.
- [31] Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs from preference data. In *International Conference on Learning Representations*, pages 34433–34448, 2025. URL: https://proceedings.iclr.cc/paper_files/paper/2025/hash/5503a7c69d48a2f86fc00b3dc09de686-Abstract-Conference.html.
- [32] OWASP GenAI Security Project. OWASP Top 10 for LLM applications 2026, August 2026. Accessed August 24, 2026. URL: <https://genai.owasp.org/resource/owasp-genai-llm-top-10-2026/>.
- [33] Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive APIs. In *Advances in Neural Information Processing Systems*, volume 37, pages 126544–126565. Curran Associates, Inc., 2024. doi:10.52202/079017-4020.
- [34] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations*, pages 9695–9717, 2024. URL: <https://openreview.net/forum?id=dHng200Jjr>.
- [35] Christian Rossow. Amplification hell: Revisiting network protocols for DDoS abuse. In *Proceedings of the 21st Annual Network and Distributed System Security Symposium*, 2014. doi:10.14722/ndss.2014.23233.
- [36] Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. Identifying the risks of LM agents with an LM-emulated sandbox. In *The Twelfth International Conference on Learning Representations*, pages 27031–27098, 2024. URL: https://proceedings.iclr.cc/paper_files/paper/2024/hash/7274ed909a312d4d869cc328ad1c5f04-Abstract-Conference.html.

- [37] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Tool-former: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, pages 68539–68551. Curran Associates, Inc., 2023. doi:10.52202/075280-2997.
- [38] David Schmotz, Luca Beurer-Kellner, Sahar Abdelnabi, and Maksym Andriushchenko. Skill-Inject: Measuring agent vulnerability to skill file attacks. *arXiv preprint*, 2026. arXiv:2602.20156.
- [39] Gaurav Somani, Manoj Singh Gaur, Dheeraj Sanghi, Mauro Conti, and Rajkumar Buyya. DDoS attacks in cloud computing: Issues, taxonomy, and future directions. *Computer Communications*, 107:30–48, 2017. doi:10.1016/j.comcom.2017.03.010.
- [40] Elham Tabassi. Artificial intelligence risk management framework (AI RMF 1.0). Technical Report NIST AI 100-1, National Institute of Standards and Technology, 2023. doi:10.6028/NIST.AI.100-1.
- [41] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024. doi:10.1007/s11704-024-40231-1.
- [42] Peiran Wang, Xinfeng Li, Chong Xiang, Jinghui Zhang, Ying Li, Lixia Zhang, Xiaofeng Wang, and Yuan Tian. The landscape of prompt injection threats in LLM agents: From taxonomy to analysis. *arXiv preprint*, 2026. arXiv:2602.10453.
- [43] Fangzhou Wu, Ning Zhang, Somesh Jha, Patrick McDaniel, and Chaowei Xiao. A new era in LLM security: Exploring security concerns in real-world LLM-based systems. *arXiv preprint*, 2024. arXiv:2402.18649.
- [44] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huang, Qi Zhang, and Tao Gui. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2):121101, 2025. doi:10.1007/s11432-024-4222-0.
- [45] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL: https://openreview.net/forum?id=WE_vluYUL-X.
- [46] Qing Ye and Jing Tan. Agent Contracts: A formal framework for resource-bounded autonomous AI systems. *arXiv preprint*, 2026. arXiv:2601.08815.
- [47] Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10471–10506, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi:10.18653/v1/2024.findings-acl.624.
- [48] Dongsun Zhang, Zekun Li, Xu Luo, Xuannan Liu, Peipei Li, and Wenjun Xu. MCP Security Bench (MSB): Benchmarking attacks against Model Context Protocol in LLM agents. In *International Conference on Learning Representations*, pages 58341–58372, 2026. URL: https://proceedings.iclr.cc/paper_files/paper/2026/hash/5fc47800ee5b30b8777fdd30abcaaf3b-Abstract-Conference.html.
- [49] Yuanhe Zhang, Weiliu Wang, Zhenhong Zhou, Kun Wang, Jie Zhang, Li Sun, Yang Liu, and Sen Su. Leech-Hijack: Covert computational resource exploitation in intelligent agent systems. *arXiv preprint*, 2025. arXiv:2512.02321.
- [50] Kaiyu Zhou, Yongsun Zheng, Yicheng He, Meng Xue, Xueluan Gong, Yuji Wang, Xuanye Zhang, and Kwok-Yan Lam. Beyond max tokens: Stealthy resource amplification via tool calling chains in LLM agents. *arXiv preprint*, 2026. arXiv:2601.10955.
- [51] Yuguang Zhou, Xunguang Wang, Pingchuan Ma, Zhan-tong Xue, Zhaoyu Wang, and Shuai Wang. From shield to target: Denial-of-service attacks on LLM-based agent guardrails. *arXiv preprint*, 2026. arXiv:2606.14517.
- [52] Zhenhong Zhou, Zherui Li, Jie Zhang, Yuanhe Zhang, Kun Wang, Yang Liu, and Qing Guo. CORBA: Contagious recursive blocking attacks on multi-agent systems based on large language models. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 6899–6908. Association for Computational Linguistics, July 2026. doi:10.18653/v1/2026.findings-acl.342.

C Supplementary Tables and Figures

Replay Audit and Layer Ablation.

Primary and expanded replay accounting. The primary matrix contains 19 source traces (15 attack and four benign) evaluated without defense and under three presets, yielding 76 trace-mode rows. All 13 non-trivial multi-turn attacks terminate in all 39 preset evaluations: V0/V1/V2/V4/V5 at Turn 2 and V3 at Turn 3. Two additional attacks are single-turn CAF=1 cases. The expanded immutable audit covers 80 retained JSONL files; 52 contain replay-eligible turns and 28 are summary-only `no_turn_records` exclusions. Across the eligible files, all 41 non-trivial multi-turn attacks terminate under all three presets (123/123 evaluations) using serialized per-turn tool-call counts.

Live Calibration and Cache Effects.

Cerebras live calibration. On the Cerebras provider-model pair, V0 short-burst and staggered sizing reach \$0.116/min and \$0.010/min, matched D1–D4 V0 leakage is \$0.00320, and five seed-inflation sessions reach max $\Delta p = 8,486$, $\Delta p/p_1 = 1.17$, and \$0.00819 mean leakage (Table 6).

Second-provider seed-inflation check. In a second-provider seed-inflation live check, Mistral Small Latest completed 5/5 staggered sessions with 10 live tool calls. With first-turn seed

9,046, max $\Delta p = 11,431$, and $\Delta p/p_1 = 1.26$, full D1–D4 stopped every session at the D1 per-turn budget on Turn 2 with \$0.00446 mean leakage. This second-provider full-stack live calibration exercises a different first-trigger path from Cerebras.

Cache-aware cost calibration. Cache rows cover V0/V3/V5/benign with $N = 5$ each. Effective cost applies the fixed nominal rate table used by the canonical accounting: Mistral Small 4 and Groq `openai/gpt-oss-120b` use \$0.15/\$0.015/\$0.60 and \$0.15/\$0.075/\$0.60 per million uncached-input/cached-input/output tokens, respectively [11, 12, 27, 28]. For each call, we multiply provider-reported token categories by their corresponding fixed rates and normalize cumulative cost by the standard-input-rate cost of the first-turn prompt. Caching changes Mistral/Groq cost by 50.2%/0.9%, while attack effective-cost amplification remains $16.0\text{--}20.8\times/26.1\text{--}37.1\times$ and attack-/benign ratios remain $0.88\text{--}1.31\times$. Cache behavior changes price but does not separate attack from benign sessions in these matrices.

MCP protocol provenance. A local stdio compatibility check used Python MCP SDK 1.23.3 and protocol revision 2025-11-25. The corresponding protocol semantics are defined by the

Table 4: CAF distributions across six core LLM families, four primary attack vectors, and the benign comparator.

Model	Metric	Benign	Direct (V0)	Polymorphic (V1)	Stealth (V3)	Adaptive (V5)	V5 vs. benign adj. p / r_{rb}
GPT-4o	Median CAF	35.6	5,778.3	4,204.9	2,427.8	440.2	0.0317* $r_{rb} = 0.89$
	IQR	32.8–87.5	1,889.8–8,921.7	3,997.1–4,692.2	2,177.7–2,798.8	286.5–4,169.9	
	Max	208.7	11,116.0	10,820.5	3,328.6	7,143.9	
	Valid n	5	5	5	6	7	
GPT-4o-mini	Median CAF	35.2	106.7	139.9	301.6	331.1	0.0317* $r_{rb} = 1.00$
	IQR	35.1–35.5	106.7–122.3	111.3–161.3	301.5–303.4	330.9–331.2	
	Max	37.8	192.8	247.0	402.1	331.3	
	Valid n	5	10	5	8	5	
Llama-3.3-70B	Median CAF	42.3	39.8	1.0	113.2	118.8	0.0155* $r_{rb} = 0.78$
	IQR	41.4–73.3	1.0–39.8	1.0–46.0	113.2–113.2	118.7–118.8	
	Max	133.2	39.8	2,436.5	1,486.6	118.9	
	Valid n	10	9	11	5	11	
Mistral-Large	Median CAF	23.7	4,336.6	5,561.1	2,394.3	269.2	0.0317* $r_{rb} = 1.00$
	IQR	23.2–25.1	4,336.6–10,136.3	4,167.1–6,108.3	1,423.7–2,394.3	38.7–269.3	
	Max	26.4	14,293.4	6,277.1	2,394.3	2,588.3	
	Valid n	5	11	5	11	5	
Nemotron-120B	Median CAF	55.1	3,599.8	3,847.3	3,007.2	1,426.2	0.0317* $r_{rb} = 1.00$
	IQR	47.9–57.7	3,120.3–3,599.9	3,380.7–4,612.2	420.9–3,761.4	905.0–1,426.5	
	Max	59.3	3,602.0	5,750.2	3,772.7	1,426.6	
	Valid n	5	5	5	7	5	
Qwen-3-235B	Median CAF	1.0	833.9	616.9	485.2	1,010.7	0.0269* $r_{rb} = 1.00$
	IQR	1.0–1.0	734.1–833.9	428.0–843.4	80.7–485.2	291.7–1,144.7	
	Max	1.0	1,460.5	1,064.5	830.8	1,144.8	
	Valid n	5	10	5	9	6	

Note. The final column reports two-sided Mann–Whitney p -values Holm-adjusted across the six family-wise comparisons and rank-biserial effects r_{rb} for V5 versus benign. Positive r_{rb} indicates higher CAF under V5 (* $p < 0.05$, ** $p < 0.01$).

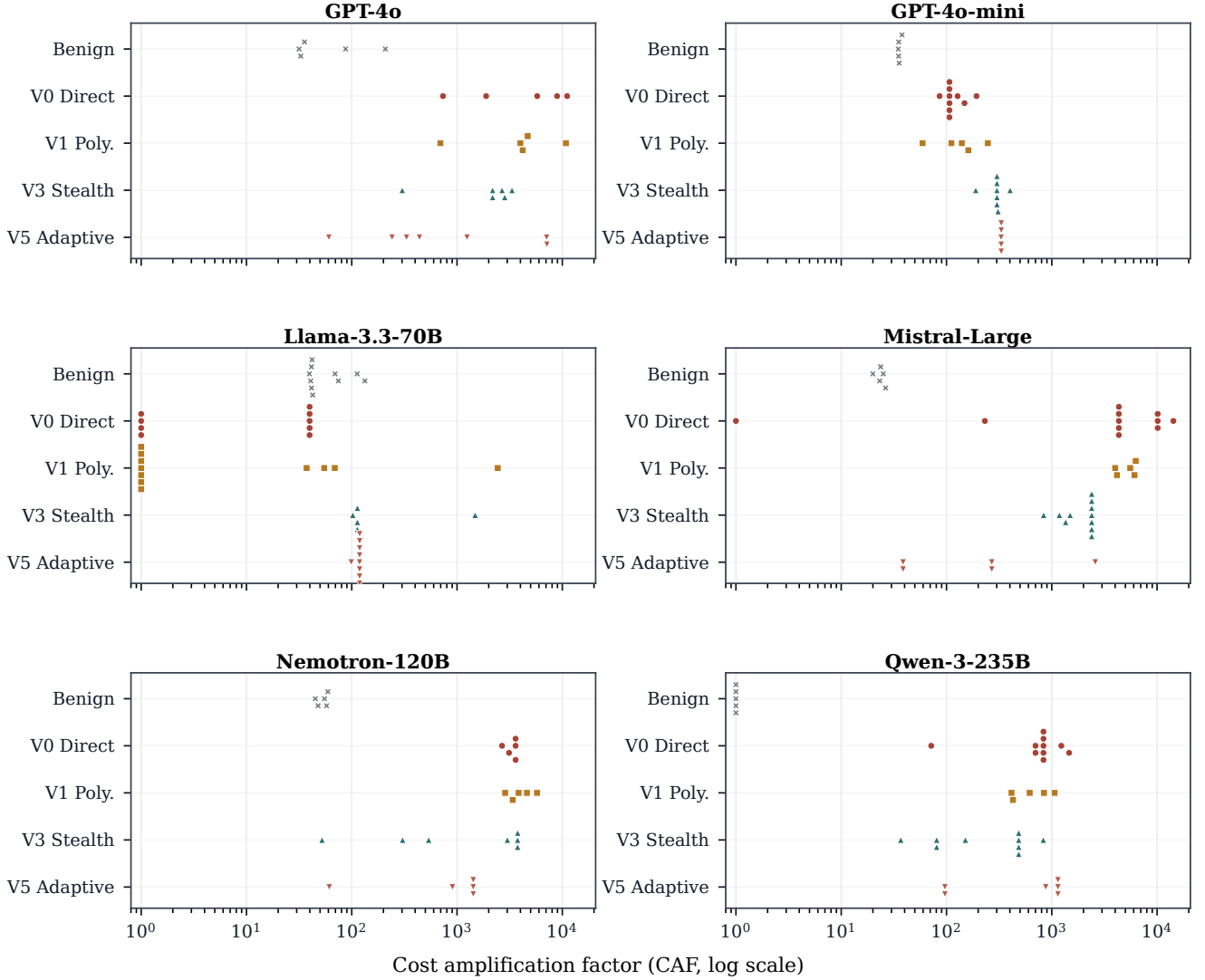


Figure 9: Execution-level CAF for the 206 runs summarized in Figure 3. Each marker represents one execution; deterministic vertical stacking separates coincident and near-coincident values within a condition row without changing CAF on the horizontal axis. Panels share a logarithmic CAF scale and retain the main figure’s condition colors and marker shapes.

MCP specification [29]. Archived Mistral live-run manifests omit the resolved SDK and negotiated protocol versions; we therefore treat those runs as version-unspecified. The 2026-07-28 revision is defined by the newer specification [30]; a separate zero-provider engineering check validated SDK 2.0.0 against that revision. These compatibility checks are not included in this paper-only source package.

Core deployment provenance. Historical core manifests preserve serialized model identifiers and usage telemetry but not temperature, maximum-turn, cache-policy, or pricing-version fields.

Long-Workflow and D4 Successors.

D3 successor diagnostic. Frozen moderate D3 interrupts

60/60; the versioned growth-gated successor has zero D3 interruptions, while the full stack completes 53/60 because D4 stops seven tasks early at median Turn 13. A seven-case idle/reset, burst, depth, and task-contract boundary suite passes.

D4 successor check. Progress-authorized D4 uses a \$0.10/\$0.13/\$0.16 checkpoint schedule and 10–12 exact host-verified milestones across 12 held-out workflows in six categories. The preregistered infrastructure-only repair preserves the protocol and policy hashes. Final ITT outcomes comprise 9/12 protocol completions, one infrastructure failure, and two model-incomplete sessions, with 0/12 defense interruptions. A fixed-\$0.10 replay of the recorded action histories yields

Table 5: Moderate-preset single-layer ablation on one attack sequence per vector.

Attack	All	No D1	No D2	No D3	No D4
V0	0.022	0.022	0.151	0.022	0.022
V1	0.024	0.024	0.100	0.024	0.024
V2	0.128	0.128	0.128	0.128	0.128
V3	0.020	0.020	0.128	0.020	0.020
V4	0.010	0.010	0.114	0.010	0.010
V5	0.008	0.008	0.117	0.008	0.008

Note. Cells report cumulative nominal USD through the first blocking turn, inclusive. **All** uses D1–D4; each other column removes the named layer. Bold values differ from **All**.

Table 6: Nominal burn-rate and first-trigger calibration on Cerebras gpt-oss-120b: two defense-off V0 schedules, a matched full-stack V0 schedule, and five full-stack seed-inflation sessions.

(a) Execution outcome.			
Schedule	Defense	Completed <i>n</i> / <i>N</i>	First stop
V0 burst (2 concurrent)	Off	2/2	max-turns T2
V0 staggered (2 concurrent)	Off	5/5	max-turns T2
V0 staggered (2 concurrent)	D1–D4	5/5	D2 at T2
Seed inflation	D1–D4	5/5	D4 at T2

(b) Nominal cost calibration.					
Schedule	Defense	Mean/leak (USD)	Batch (USD)	Observed USD/min	Linear \$10 (min)
V0 burst	Off	\$0.00321	\$0.00641	\$0.116	86.3
V0 staggered	Off	\$0.00320	\$0.01602	\$0.010	979
V0 staggered	D1–D4	\$0.00320	\$0.01601	\$0.007	1,441
Seed inflation	D1–D4	\$0.00819	\$0.04096	\$0.010	1,010

Note. Nominal costs use the fixed provider-listed rate table. Full-stack leakage includes the triggering turn. Seed inflation uses a calibration-specific \$0.006 D4 threshold, including trigger-turn overshoot; Section 8 evaluates the \$0.10 replay preset. Five-session rates include observed throttling and retries. The linear \$10 projection uses completed-session rates.

7/12 completions and 2/12 pre-completion interruptions. Success requires exact host-oracle completion under the frozen task contract.

D4 transfer sensitivity. We froze the protocol before the 48-session post-qualification cohort. Two invalid qualification protocols supplied provider-usage measurements for packet sizing and cache-state design; every reported transfer estimate uses the 48 formal sessions. Across 21 available-case pairs, all nine discordant task-success outcomes favor progress-authorized D4, yielding exact two-sided $p = 0.0039$; the paired median effective-cost increase is +\$0.00511. Each policy appears 12 times in each run position. Fixed succeeds in 4/12 first/cold-position sessions versus 9/12 second/warm-position sessions, and progress-authorized D4 succeeds in 11/12 at each position; eight of nine progress-authorized wins occur when fixed runs first/cold-position and one when

Table 7: Cache-aware cost calibration and live MCP utility over stdio.

(a) Cache-aware cost calibration.		
Metric	Groq GPT-OSS-120B	Mistral Small 2603
Sessions	20	20
Cached-input share	2.0%	57.9%
Total gross/effective cost	\$0.0190 / \$0.0188	\$0.0159 / \$0.0079
Cost saved	0.9%	50.2%
Effective-cost amplification	26.1–37.1×	16.0–20.8×
Attack / benign cost	0.88–1.25×	1.00–1.31×

(b) Live MCP utility (stdio).	
Metric	Observed value
Stack	Mistral Small 2603 + MCP stdio
Attacks blocked	10/10
Benign tasks completed	20/20 (0 blocked)
First-trigger layer	D1: 6; D2: 4
Session duration	4.21 / 6.15 s (median / p95)
MCP time/session	12.93 ms (median)
Defense time/session	0.031 ms (median)

Note. In (a), each provider contributes five V0, V3, V5, and benign sessions. Cached-input share is provider-reported cached prompt tokens divided by total prompt tokens, pooled across the 20 sessions; reported costs are total nominal USD under the fixed listed-price table. In (b), MCP and defense times are within-session cumulative times, summarized by the median across 30 sessions.

fixed runs second/warm-position. These post-hoc strata reveal order/cache-position sensitivity.

Mechanism and History-Intervention Details.

Related-mechanism comparison. Across two provider-model pairs, all 40/40 persistent-rebilling, structural-loop-proxy, cost-optimization-proxy, and matched-benign sessions satisfy the same four-turn/four-tool-call protocol oracle. Success-gated effective-cost ratios versus provider-matched benign are $1.249 \times / 1.308 \times$ for V0, $0.880 \times / 1.004 \times$ for V3, and $1.192 \times / 1.235 \times$ for V5 on Groq/Mistral (Table 12). V3 and V5 provide mechanism-matched proxy comparisons for structural-loop and cost-optimization mechanisms, respectively; Gross/projected reports the fixed-action projection defined in Table 12.

History-policy results. Table 13 reports the independently executed Full, Drop, and Compress policies defined in Section 5.3 across persistent-return and matched-necessary tasks.

Mistral completes all 72 sessions; Groq has 71 cost-complete sessions after one matched-necessary/Drop parse failure. That cell reports 2/11 available-case success and 2/12 planned/ITT success; Mistral Drop is 2/12. Intervals use 20,000 task-paired percentile-bootstrap resamples, and m counts pairs where both policies achieve exact host-oracle success. All four persistent mean effects are positive, with 47/48 individual task costs increasing. In the sole reversal, Full uses more input tokens but Drop emits enough output to cost slightly more. Among persistent both-success pairs

Table 8: Provider-specific benign false positives under post hoc D2 ratio-branch variants.

Provider/model	Turns	False-positive scenarios			
		Raw D2	Floor 512	Floor 768	Gate 3k
Primary Llama-3.1-8B	575	0/210	0/210	0/210	0/210
GitHub GPT-4o-mini	468	1/210	1/210	0/210	0/210
Cerebras GPT-OSS-120B	847	0/210	0/210	0/210	0/210
Mistral Small Latest	541	1/210	1/210	0/210	0/210
Groq GPT-OSS-120B	911	0/210	0/210	0/210	0/210
All five corpora	3,342	2/1,050	2/1,050	0/1,050	0/1,050

Note. False-positive entries report triggered scenarios / benign provider-scenario executions. Turns counts all recorded interaction turns, including the first-turn seed. Floor k replaces the ratio denominator p_1 with $\max(p_1, k)$; Gate 3k raises only the ratio branch’s minimum-growth gate from 2,000 to 3,000 tokens. Raw D2 is the original moderate rule.

Table 9: Growth-gated D3 outcomes on 41 replay attacks and 60 completed 8–12-step workflows.

Policy	Replay	Long tasks		
	Blocked ($N = 41$)	Tasks done ($N = 60$)	Pre-comp. D3 stops ($N = 60$)	Pre-comp. D4 stops ($N = 60$)
Frozen moderate	41/41	0/60	60/60	0/60
Successor + D4	41/41	53/60	0/60	7/60
Successor, D1–D3	41/41	60/60	0/60	0/60

($m=10$ in each comparison), Full/Drop and Full/Compress ratios are $1.37\times/1.31\times$ on Mistral and $1.25\times/1.22\times$ on Groq. **Operating-Point and Deployment Evidence.**

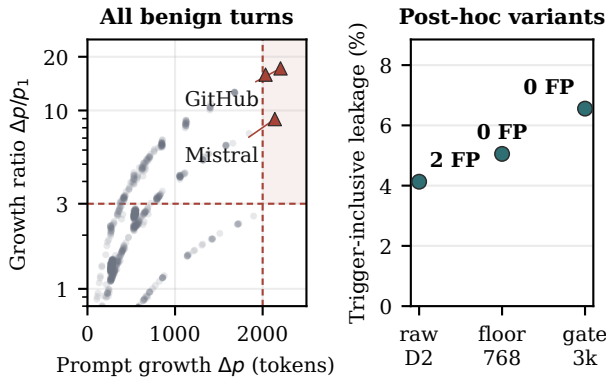


Figure 10: D2 operating point across 1,050 benign provider-scenario executions. Left: 2,292 post-seed turns from 984 multi-turn scenarios; three raw-trigger rows belong to two scenarios, and $\Delta p/p_1$ uses a logarithmic axis. Right: false-positive counts retain the 1,050-scenario denominator, while attack leakage averages trigger-inclusive fractions over six replay attacks.

Table 10: D3/D4 successor check on 12 Cerebras GPT-OSS-120B workflows.

Policy	Protocol outcomes ($N = 12$)			
	Tasks complete	Pre-comp. blocks	Infra. failures	Model incomplete
Fixed \$0.10 (counterfactual)	7/12	2/12	1/12	2/12
Checkpoint budget (live)	9/12	0/12	1/12	2/12

Note. Intention-to-treat (ITT) outcomes include all 12 workflows and partition them into task completion, pre-completion block, infrastructure failure, and model-incomplete execution. The checkpoint-budget row is live and releases budget at exact host-verified milestones; the fixed-\$0.10 counterfactual replays the same recorded action histories under a constant cap and counts task completion when it precedes the resulting block.

Table 11: D4-only Mistral Small 4 transfer: 24 post-qualification workflows independently executed under each policy (48 sessions).

Metric	Fixed	Progress-auth.
ITT task success	13/24 (54.2%)	22/24 (91.7%)
Pre-completion D4 stop	9/24	0/24
Tool failure	2/24	2/24
Mean cost/session (USD)	0.0992	0.1126
Median cost/session (USD)	0.1037	0.1143

Note. Fixed holds D4 at \$0.10; progress-authorized D4 releases \$0.13/\$0.16 at 7/10 and 10/10 host-verified milestones. ITT success requires all 10 milestones and an exact match to the typed JSON oracle. Cost rows report nominal cache-aware effective cost in USD per session over all 24 sessions per policy.

Table 12: Success-gated cost comparison of related mechanisms on matched workflows.

Treatment	Task success	Cost ratio	
		Effective / benign	Gross / projected
<i>Groq</i>			
Persistent rebilling	5/5	1.249×	1.675×
Structural-loop proxy	5/5	0.880×	1.344×
Cost-optimization proxy	5/5	1.192×	1.593×
Matched benign	5/5	1.000×	1.555×
<i>Mistral</i>			
Persistent rebilling	5/5	1.308×	1.786×
Structural-loop proxy	5/5	1.004×	1.386×
Cost-optimization proxy	5/5	1.235×	1.690×
Matched benign	5/5	1.000×	1.660×

Note. Each provider-treatment cell contains five sessions; task success requires the four-turn/four-tool-call oracle. Ratios use mean per-session costs over successful sessions. Effective/benign compares observed cache-aware effective cost with provider-matched benign. Gross/projected divides observed gross cost by a fixed-action no-history projection over the same tool-call and completion-token sequence; task success is measured on the observed execution. V3 and V5 are mechanism-matched proxies.

Table 13: Closed-loop history-policy cost effects.

Return	Comparison	n	m	Δ input	Δ cost [95% CI]	Median Δ \$ [IQR]	Both-success ratio
<i>Groq</i>							
Persistent	Full/Drop	12/12	10	37.88%	22.26% [15.56, 28.76]	\$0.00012 [\$0.00008, \$0.00014]	1.25×
Persistent	Full/Compress	12/12	10	33.10%	21.15% [15.04, 27.27]	\$0.00011 [\$0.00007, \$0.00013]	1.22×
Necessary	Full/Drop	11/12	2	35.50%	22.13% [0.46, 42.20]	\$0.00014 [\$0.00009, \$0.00020]	1.54×
Necessary	Full/Compress	12/12	11	31.38%	29.15% [20.26, 38.57]	\$0.00016 [\$0.00013, \$0.00019]	1.30×
<i>Mistral</i>							
Persistent	Full/Drop	12/12	10	45.97%	35.93% [32.50, 39.42]	\$0.00010 [\$0.00010, \$0.00012]	1.37×
Persistent	Full/Compress	12/12	10	40.30%	30.05% [27.65, 32.53]	\$0.00008 [\$0.00008, \$0.00009]	1.31×
Necessary	Full/Drop	12/12	2	51.60%	38.88% [36.51, 41.24]	\$0.00012 [\$0.00012, \$0.00012]	1.39×
Necessary	Full/Compress	12/12	10	46.18%	33.03% [29.83, 36.09]	\$0.00011 [\$0.00010, \$0.00012]	1.32×

Note. Persistent-return rows are the primary cost estimand; matched-necessary rows assess continuity when returns carry task-required records. n gives evaluable/planned pairs, and m gives both-success pairs. Δ input and Δ cost are Full/comparator ratios of arm means minus one; 95% CIs use 20,000 task-paired percentile-bootstrap resamples. Median Δ \$ is Full minus comparator. The Groq Necessary-Drop comparison uses 11 available pairs after one parse failure. Both-success ratios with $m = 2$ are descriptive summaries.

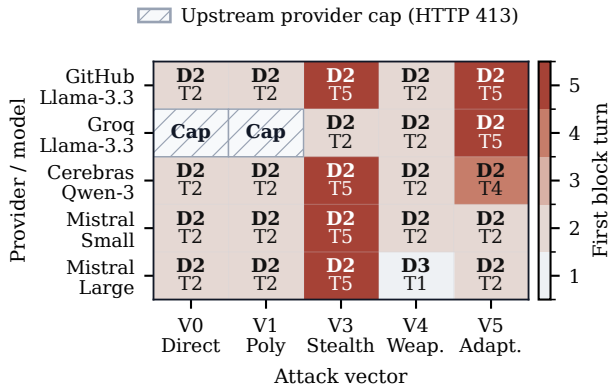


Figure 11: Live inline placement across five attack vectors and five provider-model pairs. Fill encodes first block turn; text gives interception layer/turn; hatching marks an upstream HTTP 413 cap. D2/D3 intercept 23/25 sessions before the next provider call; the other two stop at the provider cap.

Ecosystem-Scan Method. The ecosystem scan covers publicly available MCP server and transport repositories whose source contents were collected through March 30, 2026. Using explicit `mcp_server*` or `mcp_transport*` discovery signals, we identify 4,535 distinct candidates. The scan examines nonempty Python, JavaScript/TypeScript, and Go source files after excluding tests, examples, documentation, generated code, dependencies, and declaration files. Of these candidates, 3,830 contain at least one eligible source file and form the scanned corpus and prevalence denominator.

The remaining 705 contain no eligible code under these rules and appear only in collection-flow accounting. We map lexical patterns to D1 token/context limits, D2 rate/window controls, D3 iteration/turn/tool-call limits, and D4 cost/spend budgets, and apply family-specific path and contextual filters. Each accepted family match counts as a code-visible safe-guard proxy, with at most one proxy recorded per family per repository.

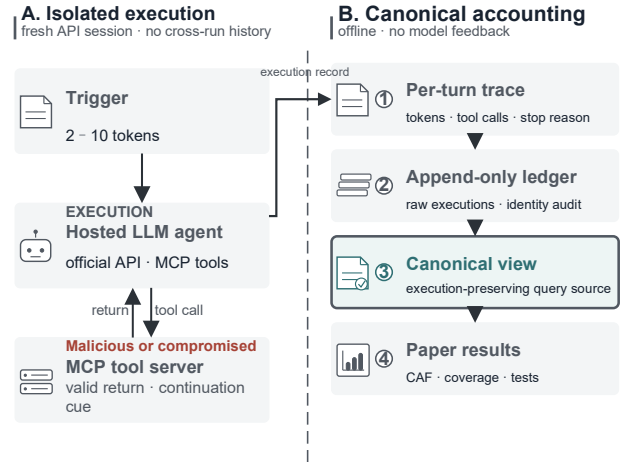


Figure 12: Evaluation pipeline from isolated execution to offline accounting. Each fresh API session emits one execution record to the append-only ledger, from which offline accounting builds the analysis view used for reported results.

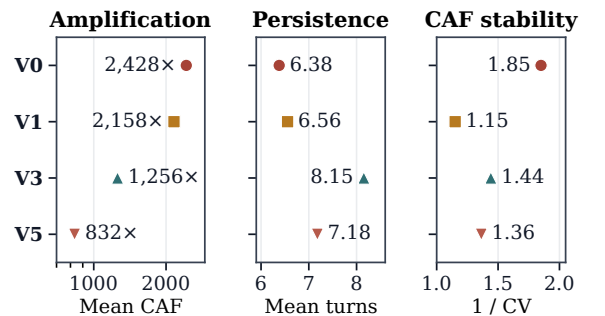
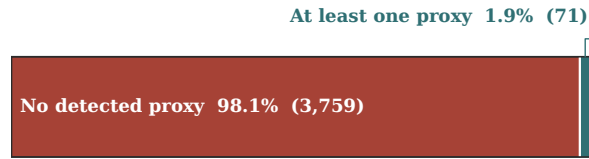


Figure 13: Attack-vector tradeoffs for V0, V1, V3, and V5 ($n = 50/36/46/39$ executions). Panels report mean CAF, mean turns, and CAF stability (inverse mean within-family CV across six family cells); higher stability is better. No vector leads on all three, and each panel retains its native scale.

a Safeguard-proxy prevalence

Real-world public repositories · n = 3,830



b Proxy-family depth

Proxy-positive repositories · n = 71

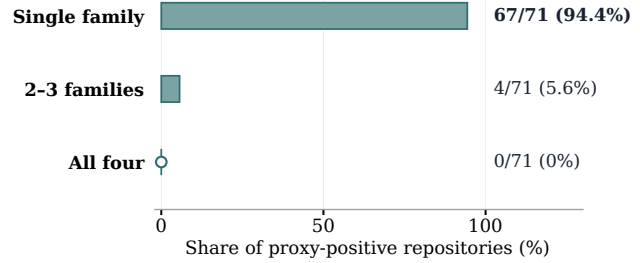


Figure 14: Prevalence of code-visible safeguard proxies across 3,830 public MCP server and transport repositories. (a) The scan detects no proxy in 3,759/3,830 repositories (98.1%) and at least one in 71/3,830 (1.9%). (b) Among those 71 positives, 67 (94.4%) match one family, 4 (5.6%) match two or three, and none match all four.

Scan Validation. Manual adjudication of a 60-row stratified set confirms all 20 sampled positive rows at the proxy-family level and all 20 sampled code-bearing zero rows as containing no visible proxy; 20 context-dependent boundary cases exercise the path and contextual filters.

We additionally probe 80 repositories classified as zero-proxy, comprising 60 broad-term candidates and 20 zero-broad controls. The probe surfaces three D2-like candidates; manual adjudication identifies all three as generic pacing/rate controls, leaving their zero-proxy classifications unchanged.