PACE: PROVENANCE-AWARE CAPABILITY ENFORCEMENT FOR TOOL-USING LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tool-using large language model (LLM) agents turn generated text into real side effects, so poisoned tool metadata, retrieved pages, memory, and reusable skills can steer the next call. Vetting an artifact before admission does not settle this. A safe variant and a leaking variant can produce the same admission evidence, and a sound gate then cannot relax that site for either. We make that condition precise, which leaves the last boundary a deployment can still act on. We present Provenance-Aware Capability Enforcement (**PACE**), which mediates every tool call immediately before it executes. Path confinement proposes an executable cut of represented influence paths, while capability and effect verification checks schema-defined effects against authority compiled from the authenticated request. We distinguish the certified execution contract from the evaluated configuration, which can restore an authorized call after a proposed block or apply a declared repair. Confinement requires the final action to preserve the certified cut. On eight executable agent-security benchmarks with three target-model families, the evaluated configuration gives strictly lowest attack success in **62 of 79 eligible attack columns** and ties in 14; full-benchmark native utility loses at most three points relative to the undefended agent. A complete ablation over 1167 paired cases attributes most security gains to effect verification and refusal control to boundary adaptation. A reduced-scale adaptive search succeeds on 0/30 out-of-authority targets against the defense. Source code is in the supplemental material.

1 INTRODUCTION

Tool-using LLM agents plan, call tools, and act inside one loop (Yao et al., 2023; Schick et al., 2023; Qin et al., 2024; Patil et al., 2024). The same loop reads retrieved pages, tool returns, files, and third-party skill descriptions, so instructions and data share the context that picks the next action (Zverev et al., 2025). Prompt injection exploits this across tool suites, web tasks, and multi-step workflows (Perez & Ribeiro, 2022; Greshake et al., 2023; Zhan et al., 2024; Debenedetti et al., 2024; Zhang et al., 2025; Evtimov et al., 2025; Liu et al., 2024; Yi et al., 2025; Ruan et al., 2024), with payloads arriving through MCP schemas (Wang et al., 2026c; Yang et al., 2025; Hou et al., 2026), reusable skills (Duan et al., 2026), or poisoned memory and knowledge bases (Chen et al., 2024; Zou et al., 2025; Wang et al., 2026b). Once the agent can send mail, write files, or spend money, that text can cause a real effect. We target deployments where it also reaches the planning context, as MCP tool descriptions and marketplace skills do today. Section B surveys related work.

Defenses fall into four families: flow control anchored in a trusted plan or a reconstructed trace (Debenedetti et al., 2025; Costa et al., 2025; Zhong et al., 2025; Wang et al., 2025), declarative privilege and task alignment (Shi et al., 2025; Jia et al., 2024), detection and re-execution (Zhu et al., 2025; Liu et al., 2025; Chennabasappa et al., 2025; Rebedea et al., 2023; Inan et al., 2023), and instruction/data separation (Wallace et al., 2024; Chen et al., 2025a;b; Hines et al., 2024). Their trusted inputs differ: plans, request-derived policies, instrumented provenance, or learned distinctions between instructions and data (Section B). *Trust inversion* arises when the authority for admission is itself attacker-controlled. An admission gate may review a skill by reading its specification and the agent later follows that same specification, so the object under review and the authority used to review it are one attacker-written string. Published attacks on tool metadata exhibit this confused deputy (Wang et al., 2026c); adversarial prompting of reusable skills exposes a related surface (Duan

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

et al., 2026). Dynamic labels track instrumented flows but do not certify behavior over attacker-controlled contexts (Enck et al., 2010; Kemerlis et al., 2012). Underneath sits a mismatch of evidence: confidentiality compares two runs and needs the semantics of both (Clarkson & Schneider, 2010; Terauchi & Aiken, 2005; Barthe et al., 2004), while a reviewer holds evidence about one (Necula, 1997; Sekar et al., 2003). Section 3 makes this precise over the *behavior family*, the trajectory distributions an artifact induces across attacker-supplied contexts, and gives the condition under which no admission-time evidence licenses relaxing a site.

We therefore decide at the last tool boundary before an external effect. Four things meet there: the concrete proposal, the authenticated request, the observed provenance, and a hook that can still change the outcome. **PACE** pairs path confinement with capability and effect verification. The first proposes actions that sever represented influence paths; the second checks request authority over concrete effects. Their composition depends on the final action: restoring a blocked call can discard separation, and rewriting an authorized call can change its effects. The certified contract preserves the cut and rechecks the final call; the evaluated configuration permits restoration and declared repairs. Recent provenance and execution monitors already check argument provenance and fine-grained authority (Fan et al., 2026; Wang et al., 2026a; Santos-Grueiro, 2026). Our analysis identifies the graph-separation condition an intervention must retain alongside effect authority (Section 4.4), within the flows and effects the schemas represent.

We make the following contributions.

- The security-relevant object is an artifact's behavior family over attacker-controlled contexts. We show when admission-time evidence cannot license relaxing an egress site.
- We propose **PACE**, pairing path confinement with capability and effect verification at the last enforceable tool boundary, and distinguish its certified contract from the restoration and repair behavior of the evaluated configuration.
- We prove represented-path separation, authorized-effect soundness, and a conditional grounded-execution theorem, and specify which final interventions retain its path-separation premise. These guarantees concern represented chains.
- On eight executable benchmarks with three target-model families, the evaluated configuration achieves strictly lowest ASR in 62 of 79 eligible columns and ties in 14, with a paired component study and a reduced-scale adaptive evaluation.

2 Preliminaries

We define the agent execution model and the security property used throughout the analysis.

2.1 Agent Execution and Threat Model

A frozen policy M alternates free text with tool calls. An *artifact* K is a skill, a tool description, or an MCP schema; paired with a context $c \in \mathsf{Ctx}$ carrying the request, injected text, tool returns, state, and possibly a secret s, it induces a trajectory measure $\mu_{K,c}$ (Section C.1). The security-relevant object is therefore not K but the *behavior family* $\mathsf{Beh}(K) = \{\mu_{K,c} : c \in \mathsf{Ctx}\}$, which no audit of K enumerates. The attacker authors K and any untrusted span of c, adapts to each defense outcome, and succeeds on an unrequested irreversible effect or on a disclosure of s. The defender acts only at the tool boundary, reading proposals and returns and weakening or blocking a call, and it cannot retrain M, instrument a tool's interior, or enumerate Ctx.

2.2 The Security Property and Why It Is Hard

Probabilistic noninterference (Goguen & Meseguer, 1982; Sabelfeld & Myers, 2003) is the yardstick we measure against, not a property we claim. Section 3 uses it to state what admission-time evidence cannot license, and Section 4.4 says how far short of it **PACE** stops. Let $\mathsf{egress}(\tau)$ list the egress events of a trajectory in order over a finite site set Ω, with the answer r as one site, and let $\mu^{\Gamma}_{K,c}$ be the measure induced when M runs behind a mediator Γ. Two contexts differing only in s should then give egress no observer can tell apart beyond δ:

$$\Phi_{\mathrm{NI}}(K, \Gamma; \delta): \quad \sup_{\bar{c} \in \overline{\mathsf{Ctx}}} \sup_{s_0, s_1} d\big(\mathsf{egress}_{\#}\mu^{\Gamma}_{K,\bar{c}[s \leftarrow s_0]},\ \mathsf{egress}_{\#}\mu^{\Gamma}_{K,\bar{c}[s \leftarrow s_1]}\big) \ \leq\ \delta. \tag{1}$$

The projection keeps raw bytes, so base64, hashes, paraphrase, and fragmentation sit inside the comparison and need no separate leak predicate. Finite observations alone do not discharge Equation (1) over an unrestricted behavior family. It compares induced measures rather than isolated traces, so a finite sample is not a conclusive witness of a distributional violation. Hyperproperty reasoning (Clarkson & Schneider, 2010) and self-composition (Terauchi & Aiken, 2005; Barthe et al., 2004) require an adequate semantic model of the paired executions; an artifact's code slice need not model the agent's choice of inputs.

3 Limits of Admission-Time Certification

Consider a mail skill whose code always calls `send_email(`*body*`)`, with a natural-language specification that tells the model how to build *body*. A benign specification uses a fixed template, a malicious one asks the model to copy a credential out of context, and the code and interface are identical either way. That isolates the question: *when does evidence collected before execution justify removing mediation from an egress site?*

Fix an egress site $\omega \in \Omega$ and let $O_\omega(\tau)$ be the subsequence of $\mathsf{egress}(\tau)$ tagged ω, or $\perp$ if the site is never reached. It is measurable (Lemma D.1), so restricting Equation (1) to it gives a local criterion:

$$\Phi_{\mathrm{site}}(K, \Gamma, \omega; \delta): \quad \sup_{\bar{c} \in \overline{\mathsf{Ctx}}} \sup_{s_0, s_1} d\big((O_\omega)_{\#}\mu^{\Gamma}_{K,\bar{c}[s \leftarrow s_0]},\ (O_\omega)_{\#}\mu^{\Gamma}_{K,\bar{c}[s \leftarrow s_1]}\big) \ \le\ \delta. \tag{2}$$

Total variation contracts under a measurable map, so Φ_{NI} forces Φ_{site} everywhere, and relaxing a violating site already forfeits the global criterion. A *gate* sends admission-time evidence about K to $\{\mathsf{mediate}, \mathsf{relax}\}$, and is δ-*sound* when Equation (2) holds at every site for every artifact in its declared domain, a class the always-mediate gate shows is nonempty. Write $\mathcal{V}(K, \omega)$ for the variants in that domain sharing code, interface, and site but carrying different attacker-written specifications; Section D.1 fixes the reference profile and the conservative transformation it applies.

Admission decisions may use three kinds of artifact evidence: facts established over code, as in proof-carrying code (Necula, 1997); a sound abstraction of the artifact against its interface (Sekar et al., 2003) or of agent behavior reconstructed from traces (Wang et al., 2025); and dependencies read off the specification, which is what a registry reviewer must do when an artifact ships as prose (Wang et al., 2026c; Yang et al., 2025; Duan et al., 2026). Two conditions on such a route carry the argument. C1 says the route assigns one judgment to a safe and a leaking member of $\mathcal{V}(K, \omega)$. C2 says the attacker can exhibit a context in which the leaking member violates Equation (2). Neither is a claim about every admission system; Section D.1 states both formally.

Theorem 1 (No expansion under judgment ambiguity)**.** *Fix δ, a site ω, a route i, and a profile Γ_0, and suppose C1 and C2 hold at (K, ω). Let g be a δ-sound gate whose decision depends on K only through $\mathcal{J}_i(K, \omega)$ and whose induced mediator agrees with Γ_0 off ω at every artifact in its domain. Then g returns* mediate *for both variants of C1.*

Equal judgments force equal decisions, and C2's context then makes the dependent variant violate Equation (2). The work sits in C1, which for code and code-interface evidence reduces to the variant class holding one safe and one leaking member, and for text extraction to a semantics-changing rewrite the extractor cannot see (Section D.1). Evidence that separates the variants is untouched, including the code-resolved case (Proposition D.1). The theorem also locates where the missing evidence first appears. Immediately before a tool runs, the concrete arguments, the resources already read, the authenticated request, and a hook that can still change the effect coexist, none of it available at admission time. Reachability alone is not an authorization decision there, since a represented path says only that an effect may occur and cannot tell a requested, taint-free call from an attacker-induced one. The boundary therefore needs two checks, each landing on an action the runtime implements. The next section therefore has three requirements: represented unsafe paths must be severable, allowed effects must be authorized, and every abstract verdict must correspond to a real intervention.

4 **PACE**: Provenance-Aware Capability Enforcement

As shown in Figure 1, **PACE** handles each mediated call in four phases. *Propose* freezes its arguments, extends the episode provenance graph, and expands schema-defined effects. *Cut and Certify* proposes a cut of represented influence paths and checks effects against request-derived capabilities; verifiers

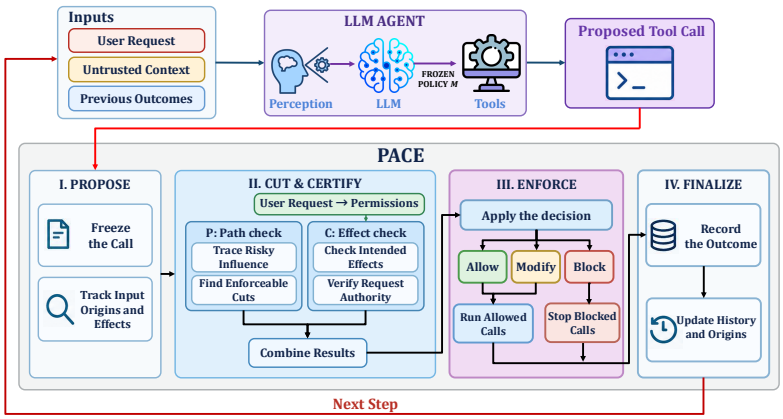


Figure 1: One mediated step of **PACE**. Phase I–PROPOSE (Section 4.1) freezes the call, extends the graph, and expands the effect atoms; phase II–CUT & CERTIFY (Section 4.2) cuts and certifies against committed evidence alone; phase III–ENFORCE (Section 4.3) installs the manifest and dispatches; phase IV–FINALIZE (Section 4.3) commits the outcome.

recompute both certificates. *Enforce* installs the selected action before dispatch, and *Finalize* records the outcome for the next call. We distinguish two configurations. **PACE**-C is the certified contract: the installed action preserves the cut and the final call passes effect verification. All experiments evaluate **PACE**-P, whose combiner can restore C-certified calls blocked by P and dispatch declared repairs without final-call recertification. Its fast paths and the conditions for retaining the certified guarantee are specified below.

4.1 PHASE I: PROPOSE

Phase I constructs the evidence for each boundary decision. Dynamic tainting would need every transformation to carry a label (Enck et al., 2010; Kemerlis et al., 2012), which tool-using agents defeat by crossing opaque boundaries. **PACE** records declared structure instead. Each proposal yields a path view over the episode graph and an effect view over the effects the schema declares. Both are folded from an append-only audit record, so a verifier can recheck a decision without rerunning the agent.

Definition 1 (Evidence and Active Graphs)**.** The *evidence graph* $H_t = (V_t, E_t)$ is a finite directed multigraph over data, operation, resource, and terminal nodes, folded from the episode record. Proposals, outcomes, blocks, cancellations, and uncertain dispatches are appended and never rewritten, so the committed part of H_t only grows within an episode. Reachability uses only the flow subgraph

$$E_t^{\mathrm{fl}} = \big\{\, e \in E_t \;:\; \mathrm{type}(e) \in \{\, \texttt{resource_access},\ \texttt{data_flow},\ \texttt{output_flow},\ \texttt{resource_write},\ \texttt{terminal} \,\} \big\}. \tag{3}$$

The *active graph* G_t^- is what one decision reads: the committed edges of H_t at σ_t together with the proposed edges of a_t, excluding refused and cancelled proposals. Two counters index that history. The execution version σ_t advances on any change to committed edges, labels, or live values. The policy epoch ε_t advances on any change to schemas, effect rules, costs, contracts, capability grants, or adapter code. Each record advancing σ_t stores the live-value set with an origin label per entry.

Snapshot and Provenance. Against that state, the host first freezes the call under a fresh attempt identifier q_t, storing each argument by value or as a versioned resource handle. Later phases read the snapshot, never a value the agent controls. A resource that is neither snapshottable nor versioned is schema-undefined, so rule R1 refuses the call. The host then matches each frozen argument against a provenance map ν of live values. A hit reuses the existing node, linking this call to committed history and to any protected value behind it. A miss mints a new data node. Provenance is kept per slot, so arguments, memory records, artifacts, and output fields carry separate origins.

Emission and Authority. With every argument now a node, the host wires the proposal into the path view. One edge runs from each schema-defined argument node to a fresh operation node $\mathrm{op}(q_t)$.

A fresh terminal $T(q_t)$ closes it, and the versioned unsafe-effect rule labels $\mathrm{op}(q_t) \to T(q_t)$. Only these *proposed* edges remain severable by the executor. A position with no schema gets no edge, and an operation outside the abstraction becomes an unknown transition, wired to an unsafe terminal if it may be irreversible. The effect view expands the same frozen call through the effect schema into *effect atoms*. Each names an operation, a target resource, a persistence class, and the argument slots that determine it. The authenticated request compiles separately into a capability bounded by purpose, resource, arguments, budget, expiry, and delegation depth. It reads only the trusted request channel, so model-generated text can neither mint nor widen a capability.

Phase II reads G_t^-, the effect atoms, the capability set, and the pinned counters without modifying the frozen decision state.

4.2 Phase II: Cut and Certify

Phase II runs two checks over that frozen material. The path chain decides which proposed edges to sever and binds each to an executor action; the effect chain decides whether the concrete effects are authorized. Their certificates answer different questions; whether both guarantees survive depends on the action the combiner actually installs.

Sources and the unsafe terminal. The path chain starts from two source sets. P_t holds only `Data` and `Resource` nodes labeled as secrets, personal data, credentials, source code, or protected payloads. A derivation check also scans proposals and observations for registered values over a finite family Ψ of deterministic re-encodings, each canonicalizing in its own alphabet:

$$\mathrm{match}(y, x) = 1 \iff \exists \psi \in \Psi : \ \psi(x) \text{ occurs in } \mathrm{canon}_\psi(y). \tag{4}$$

A match adds that argument's data node to P_t for this step, as a label and not an edge, so encodings are caught without treating substring matching as taint. The check is a bounded approximation: paraphrase, translation, and fragmentation lie outside Ψ, and Section C.6 lists the misses and false positives. The other set, B_t, over-approximates where attacker influence may sit. Every committed node carries a deployment-assigned origin, untrusted covers anything an artifact, tool return, retrieved page, or memory record supplied, and a derived node inherits its inputs'. The verifier closes the untrusted-origin live values at σ_t over committed flow edges only (Section C.2), so a call built from the user request and the host alone needs no cut. At the far end, the registry runs $\mathrm{unsafe}_{\varepsilon_t}(a_t)$ on the canonical proposal, so $U_t = \{T(q_t)\} \cap T_{\mathrm{bad}}$ holds the terminal of an unsafe call and nothing else. It is disjoint from P_t, so the cut problem is well posed and a safe call takes the empty cut.

Contracts and binding. With the endpoints fixed, binding runs before optimization, so the cut ranges only over edges some adapter can sever. For each proposed edge e the trusted adapter declares the actions it implements, $\mathsf{Impl}(e)$, and the deployment declares the subset $\mathsf{Sev}(e)$ whose warrants rule out every real link through e once the action is installed. Section C.5 states those warrants and shows why they leave redaction rarely licensed for model-chosen calls, so an argument edge with no severing action pushes the cut back to the terminal, which admits `block_call`. Both lists are finite and canonically ordered, so the binding takes the least declared cost among the survivors,

$$\beta_t(e) \in \arg \min_{a \in \mathsf{Sev}(e) \cap \mathsf{Impl}(e)} \mathrm{cost}(a, e), \qquad \beta_t(e) = \bot \text{ if the intersection is empty.} \tag{5}$$

The cut and the manifest. Those bindings become capacities on the flow subgraph of G_t^-:

$$\kappa(e) = \begin{cases} \mathrm{cost}\big(\beta_t(e), e\big), & \beta_t(e) \neq \bot \text{ and } e \text{ is } \texttt{proposed} \text{ for } a_t, \\ \infty, & \text{otherwise,} \end{cases} \qquad e \in E_t^{\mathrm{fl}}. \tag{6}$$

A super-source joins $B_t \cup P_t$ and a super-sink follows U_t, and $\mathcal{C}_t$ is the canonical minimum cut (L. R. Ford & Fulkerson, 1956): cheapest, lexicographically least at equal cost. Committed edges are therefore uncuttable, since the present executor cannot change a past call. Control and intent edges never enter the network, and strictly positive rational costs bar padding with free edges. Under the emission rules the optimum reduces to a two-way choice that one reachability pass settles in linear time (Proposition D.2). What the executor consumes is not that cut but the edge-indexed *manifest* $\mathcal{M}_t = \{(e, \beta_t(e), \mathrm{arg}(e), \mathrm{hook}(e)) : e \in \mathcal{C}_t\}$, where $\mathrm{arg}(e)$ and $\mathrm{hook}(e)$ name the registry-bound argument position and hook, one per cut-edge instance. It is *admissible* only if its entries do not conflict, every hook runs in time, and the deployment declares that each severance contract survives

under the complete set (Section C.4). Only that declaration is checked here, and its truth is a premise no automatic check reaches. A conflict triggers refusal, not a search for a costlier cut.

The effect chain and the decision rule. The effect chain checks nine obligations over every schema-defined call: authorization coverage, typed binding, operation and principal control, confidentiality, persistence, delegation attenuation, budget, freshness against request and event state, and hook availability. Write $\mathrm{eff}(a) = 1$ when all atoms of a pass. For a nonblocking manifest, let a'_t be its resulting call, and $\mathrm{low}(a_t) = 1$ when every effect is schema-labelled reversible, has an empty sink, and has no persistent side effect. The influence test $\mathrm{infl}(P_t, a_t)$ is one when a protected node reaches $\mathrm{op}(a_t)$ or the derivation check matches a protected form; $\mathrm{auth}(a_t)$ comes from a channel the artifact cannot write. The certified decision contract uses the following ordered rules with R2 and R3 disabled; the evaluated fast paths are shown explicitly:

$$\mathrm{dec}(a_t) = \begin{cases} \texttt{refuse} & \textbf{R1}\ \mathrm{schema}(\mathrm{op}(a_t), j) = \bot \text{ for the operation or some argument } j, \\ \texttt{allow} & \textbf{R2}\ \mathrm{low}(a_t) = 1, \\ \texttt{allow} & \textbf{R3}\ \mathrm{auth}(a_t) = 1 \wedge \mathrm{infl}(P_t, a_t) = 0 \wedge \mathrm{eff}(a_t) = 1, \\ \texttt{refuse} & \textbf{R4}\ \mathrm{eff}(a_t) = 0, \\ \texttt{refuse} & \textbf{R5}\ \text{no finite cut, or } \mathcal{M}_t \text{ inadmissible}, \\ \texttt{refuse} & \textbf{R6}\ \mathcal{M}_t \text{ nonblocking} \wedge \mathrm{eff}(a'_t) = 0, \\ \mathcal{M}_t & \text{otherwise, with } \mathcal{C}_t \text{ possibly empty.} \end{cases} \tag{7}$$

R1 rejects undefined schemas. In **PACE**-C, R4 checks the original call, R5 requires an executable cut, and R6 checks the final effects of a nonblocking manifest; a terminal block dispatches no call. In **PACE**-P, R3 requires all nine obligations to pass, whereas R2 records their results and can release a low-impact call despite failure of AUTH or BUDGET. Beyond these fast paths, its combiner can restore the original call when P selects a terminal block and C certifies it, or apply a registry-declared repair to a repairable verdict without rerunning R6. Thus this equation specifies the certified contract, not a claim that every evaluated dispatch satisfies it. With sound request compilation, effect labels, validators, and complete mediation, an out-of-authority irreversible or outward effect can execute only through a declared repair: R2 excludes it and unrepaired release requires C's acceptance. Sections C.6 and D.6 give the premises and argument.

Two certificates. Following proof-carrying code (Necula, 1997; Sekar et al., 2003), **PACE** ships that candidate decision as evidence rather than as a trusted verdict. The path certificate $\mathsf{cert}_t = (q_t, L_t^{\mathrm{P}}, \sigma_t, \varepsilon_t, B_t \cup P_t, U_t, \mathcal{C}_t, \mathcal{M}_t)$ carries the state it was computed against and every input to the decision. Its verifier rebuilds the graph, the sources, and the terminal from the episode record and the versioned registry, then rechecks separation and every binding; Section C.4 lists the ten conditions in implementation order. The effect certificate binds the atoms to the request, the schema, the capability witnesses, and the provenance slice a check needs, and a deterministic verifier reruns the nine obligations over it. The generator solves the cut and the verifier checks the one it is handed, so we claim recomputed separation and enforceability, not recomputed optimality. Acceptance certifies the candidate; it does not show that the combiner executed its manifest.

4.3 Phases III and IV: Enforce and Finalize

Definition 2 (Grounded action)**.** A cut edge's bound action is *grounded* at step t when the adapter exposes the corresponding hook, the executor installs it before the represented dependency can be traversed, and its severance contract holds under the complete manifest, not in isolation. A manifest is grounded when all entries are. An ungrounded entry makes the certified branch refuse or cancel.

Enforce. The executor consumes the combined decision before it invokes the tool. Three actions suppress an effect, blocking, refusal, and a dry run that commits nothing; four constrain one, a read-only view, an argument rewrite, sanitization at a mediated boundary, and a budget cap; and allow passes the call through. A severance contract may name only these, which places the mediator among the editing enforcement mechanisms rather than the truncating ones (Ligatti et al., 2005; Hamlen et al., 2006). In the certified contract, installation precedes dispatch and the call carries exactly the manifest-produced bytes, with R6 checked on those bytes. Restored or repaired calls in **PACE**-P follow the combiner's selected action. A recheck of σ_t and ε_t cancels a stale attempt. That freshness is process-local. Durable reservation, one-shot token spending, and crash reconciliation are deployment hardening, specified in Section C.3 and outside the artifact we evaluate.

Finalize. After execution **PACE** records a `ToolEvent`: whether the action ran, the resolved resource, the observable output, artifact and network effects, the executor mode, and the guard's reason. A blocked or dry-run call is recorded as not executed, an allowed or rewritten one as the effect observed. That event updates the graph, the provenance map, and the taint state the next proposal reads. A confirmed outcome commits its output edges and mints a fresh provenance identifier per returned value, unless the adapter certifies a pass-through; a dispatch with no observed output commits the proposal alone, since the call may still have taken effect. Either way the returned value first appears at step $t + 1$ and never enters its own certificate. The final response crosses the same boundary under the same state, and its terminal is registered unsafe whenever the influence test fires.

4.4 Security Guarantees

The certificates concern represented-path separation and authority over schema-exposed effect atoms; executor grounding connects cuts to real links. Neither establishes noninterference. A **PACE**-P step inherits the guarantees below only when its final action meets their premises (proofs in Section D).

Proposition 1 (Certificate soundness)**.** (a) Represented-path separation. *If the path verifier accepts* cert_t*, removing* $\mathcal{C}_t$ *makes* U_t *unreachable from* $B_t \cup P_t$ *in* G_t^-*. Each cut edge is a proposed flow-edge instance of* q_t *with a re-derived severing action;* $\mathcal{M}_t$ *binds each exactly once and is* declared *jointly admissible;* $\kappa(\mathcal{C}_t)$ *equals the stated cost; and the named state and epoch are current at acceptance.* (b) Authorized-effect soundness. *Assume capabilities issue only from authenticated requests, the effect schema and slot provenance cover the concrete effect, declared validators and declassifiers meet their contracts, and every required hook is applied. If the effect verifier certifies* a_t*, every effect atom is covered by a valid capability and satisfies the nine obligations of Section 4.2.*

Minimality remains declared, and omissions shared with the builder remain possible (Section C.6). Rewrites can change effects, so **PACE**-C rechecks (b) on a'_t under R6.

For a real causal chain $\gamma = x_0 \to \cdots \to x_{m-1} \to \zeta_t$ ending at the pending effect, a link is *covered* if its adapter schema exposes the relevant value position and a timely hook, and *emitted* if its flow edges appear before dispatch. Under maps α_{D} to data nodes and α_{T} to a terminal, γ is *simulated* if its endpoints match, its per-link segments concatenate into one path in E_t^{fl}, and each real link requires traversing every edge of its segment.

Theorem 2 (Conditional grounded execution)**.** *Let both verifiers accept with grounded* $\mathcal{M}_t$ *(Definition 2). Let* γ *connect a registered protected source to the pending unsafe effect, with every link covered, emitted in time, and simulated, and complete source and terminal labels (H3a and H3c of Section D). If the proposal, state, and epoch remain current through dispatch,* γ *cannot complete at this call. Replacing H3a with H3b gives integrity (Corollary D.1).*

Simulation maps γ to a P_t–U_t path crossing the cut. Grounded severance prevents the corresponding real link; freshness aligns certified and dispatched proposals. Proposition 2 gives sufficient adapter conditions for emission and simulation.

Proposition 2 (Transparent chains are simulated)**.** *Let* γ *have* $m \geq 1$ *and start at a registered value with a host provenance identifier. Suppose each call reads, writes, and returns only at declared positions, issues no hidden secondary effect, and preserves pass-through value identifiers; each* x_{i+1} *reaches the next call as a host-held handle or bytes* ν *matches to its own node; and every earlier call producing a value or resource version in* γ *was finalized before* q_t *was proposed. Under the host emission rules of Sections C.2 and 4.1,* γ *is emitted and simulated in* G_t^-*.*

The transparent fragment covers typed handles, host-managed values, pass-through chains, and host-versioned resources, including internal computation (Remark D.1). It excludes model-recomposed strings without host provenance, unexposed effects, tool-internal secondary calls, and writes bypassing versioned adapters. Observable refusal remains a channel, with no bound of the form Equation (2) proved at any δ. Final actions must preserve separation: in Proposition D.2, removing $S \subseteq A \cup \{e_{\mathrm{T}}\}$ separates sources from the unsafe terminal exactly when $e_{\mathrm{T}} \in S$ or $A^\star \subseteq S$, by each unsafe path's last two edges (Section D.3). Restoring a blocked call removes neither, losing separation when $A^\star \neq \emptyset$ even if C authorizes it. Complete mediation and the remaining theorem premises extend the per-call result to an episode invariant for **PACE**-C (Corollary D.2), with no such general claim for the evaluated configuration.

Table 1: Main results on benchmarks with target-model families. ASR is the worst attack group for that benchmark and model. Each block shows the undefended agent, **PACE**, and the three strongest published defenses, ranked by mean worst-group ASR. Util. is the benchmark's own score and is higher better except on MCPTox, where it is the refusal rate. Best ASR per block and model in bold.

	Qwen		DeepSeek		ChatGPT	
Method	ASR ↓	Util.	ASR ↓	Util.	ASR ↓	Util.
AgentDojo						
Baseline	48.0	62.59	99.84	75.68	100.0	73.77
CaMel	0.32	35.14	28.14	60.04	36.09	57.02
DTA	0.95	29.84	33.33	54.0	55.01	56.01
DataSent.	35.29	73.77	78.06	85.0	84.1	82.95
PACE	**0.0**	59.67	**5.09**	75.25	**8.11**	72.05
AgentDyn						
Baseline	32.5	63.39	19.5	65.0	19.5	74.82
Repeat Prompt	34.5	61.25	31.5	61.43	31.5	70.54
CaMel	**0.0**	0.0	5.5	3.75	**0.0**	0.0
DRIFT	0.56	29.82	2.78	13.39	4.5	14.64
PACE	**0.0**	60.54	**0.06**	63.57	**0.0**	73.57
ASB						
Baseline	88.5	30.29	74.08	64.0	81.08	80.25
Sandwich	88.5	32.5	81.83	69.07	71.21	87.86
Direct Para.	71.5	62.0	79.0	81.64	81.75	81.57
PoT Shuffle	99.0	28.57	65.42	57.43	59.25	67.04
PACE	**0.42**	29.57	**10.5**	64.64	**8.75**	81.63
WASP						
Baseline	19.05	78.57	9.52	79.76	4.76	86.9
Prompt Filter	14.29	82.14	7.14	75.0	11.9	83.33
Simple Static	20.37	72.64	16.67	64.29	19.04	84.52
FIDES	40.48	0.0	30.95	32.14	23.81	59.52
PACE	**7.14**	92.86	**0.0**	91.67	**0.0**	95.24

	Qwen		DeepSeek		ChatGPT	
Method	ASR ↓	Util.	ASR ↓	Util.	ASR ↓	Util.
InjecAgent-E (no util.)						
Baseline	79.07	–	52.29	–	50.5	–
Sandwich	0.59	–	50.62	–	73.12	–
Instr. Prevention	58.82	–	67.29	–	56.63	–
PI-Det.	20.0	–	56.08	–	57.36	–
PACE	**0.0**	–	**3.12**	–	**30.77**	–
MCPTox (Util.: refusal ↓)						
Baseline	32.52	0.16	56.54	0.42	86.93	27.96
Allowlist	0.72	2.46	15.18	7.66	16.29	55.78
MCIP	0.51	54.23	9.95	76.72	10.04	81.22
Progent	8.8	45.88	20.42	66.85	46.4	78.43
PACE	**0.0**	1.97	**2.62**	0.58	**2.65**	28.55
PASB (IPI util.)						
Baseline	85.0	98.47	92.5	99.24	97.5	100.0
Delimiters	45.0	99.23	55.0	98.47	35.0	99.24
Instr. Prevention	17.5	97.67	27.5	96.95	27.5	98.47
Progent	80.0	100.0	77.5	100.0	62.5	98.47
PACE	**0.0**	97.69	**5.0**	99.24	**2.5**	100.0
MSB (Util.: PUA)						
Baseline	38.75	46.34	100.0	91.24	98.75	89.08
MCIP	50.0	59.15	77.5	96.55	76.25	92.34
PI-Guard	37.5	56.42	57.5	91.98	66.25	89.97
Tool Filter	30.81	47.34	59.35	93.65	52.5	91.04
PACE	**2.5**	60.14	**3.75**	98.83	**3.75**	94.87

5 EXPERIMENTS

In this section, we give the experimental settings and show some compared results with more detailed in Section F.1. All experiments evaluate **PACE**-P, denoted **PACE** in the tables.

5.1 EXPERIMENTAL SETUP

Models and measurement. Three frozen target families, `gpt-5.6-luna`, `DeepSeek-V4-Flash`, and local vLLM-served `Qwen-3.8-27B`, decode deterministically at temperature 0 with a 1024-token completion limit. Target, attack source, and judge roles are separate; a model from the target's family never judges alone. Native metric point estimates are unpooled across benchmarks. Rates record numerators, valid denominators, invalid and technical-failure counts; missing required defense-hook invocations invalidate defended episodes (More information can be found in Sections E.1, E.2 and E.4).

Benchmarks and baselines. Eight executable benchmarks retain native attack construction: AgentDojo (Debenedetti et al., 2024), AgentDyn (Li et al., 2026a), WASP (Evtimov et al., 2025), InjecAgent (Zhan et al., 2024), ASB (Zhang et al., 2025), PASB (Wang et al., 2026b), MCPTox (Wang et al., 2026c), and MSB (Zhang et al., 2026). Baselines span five families: trusted flow and information-flow control (CaMeL (Debenedetti et al., 2025), FIDES (Costa et al., 2025), DRIFT (Li et al., 2025a)); tool-use safeguards (Progent (Shi et al., 2025), MCIP Guardian (Jing et al., 2025), ToolShield (Li et al., 2026b), AgentDojo tool filter, MCPTox allowlist); re-execution (MELON (Zhu et al., 2025)); detection (DataSentinel (Liu et al., 2025), PIGuard (Li et al., 2025b), PromptGuard-2 (Llama Team, 2025), AgentDojo injection detector, MCPTox metadata sanitizer); and ASB and WASP prompt-level defenses, including spotlighting (Hines et al., 2024). Fully instrumented DTA (Enck et al., 2010; Kemerlis et al., 2012) is a diagnostic taint-tracking reference excluded from main-table comparator selection (Section E.5). Comparisons run locally at pinned revisions. Capabilities compile deterministically from the trusted request channel, excluding attack labels and judge outputs. Benchmark-specific policies and schemas were frozen before final runs (Section E.2).

5.2 MAIN RESULTS

Within benchmarks, Table 1 compare the worst eligible attack group per model; Section F.1 gives all groups and the comparator-selection rule. Among 79 eligible per-group ASR columns,

PACE-P is strictly lowest in 62, tied lowest in 14, and higher in three, excluding the derived InjecAgent data-stealing aggregate, its three conditional DS-S2 columns, and two columns with zero undefended ASR. On `Qwen-3.8-27B`, four of five ASB groups have zero ASR. Residuals are 4.76% and 7.14% on WASP, 0.42% on ASB-DPI, and 2.50% on MSB-Call. On AgentDojo, **PACE** attains lower worst-group ASR and higher utility than CaMeL, MELON, and Progent on all three models. Relative to undefended agents, full-benchmark utility drops at most 2.92 points, gains 14.29, 11.91, and 8.34 points on WASP, and improves in MSB PUA and NRP; MCPTox refusal differs by at most 1.81 points. The three lower-ASR comparator columns have much lower companion utility (Section F.1). Reduced-scale adaptive search on 30 out-of-authority AgentDojo pairs yields 21/30 (70.0%) undefended success, including nine first successes from search-generated payloads, versus 0/30 for **PACE**-P (two-sided 95% Wilson interval [0, 11.4%]). All 157 target calls at its boundary were blocked before dispatch without R2 release. Sections E.2 and F detail the one-model, three-suite protocol and controlled checks.

5.3 Ablation Study

A complete $2 \times 2 \times 2$ factorial on 1167 frozen cases isolates path confinement (P), capability and effect verification (C), and execution-boundary adaptation (B). Eight arms share logical case identifiers, decoding, and judge, with metric-specific eligibility: 9336 scored outcomes and no technical failure. A0 is the wrapper-matched no-defense arm; A7 re-aggregates complete **PACE**-P on these cases. C jointly enables capability installation, effect expansion, verification, and the combiner; its estimate measures that bundle. The subset's task mixture differs from benchmarks.

Table 2: Ablation study with the local `Qwen-3.8-27B` endpoint.

	A0	A1	A2	A3	A4	A5	A6	A7
Path confinement (P)		✓			✓	✓		✓
Capability and effect (C)			✓		✓		✓	✓
Boundary adaptation (B)				✓		✓	✓	✓
Attack success (%), lower better								
WASP, plain text	8.3	8.3	0.0	0.0	8.3	0.0	8.3	**0.0**
WASP, URL injection	8.3	8.3	8.3	0.0	0.0	0.0	16.7	**0.0**
AgentDojo, overall	37.4	0.0	0.0	33.3	0.0	0.0	0.0	**0.0**
InjecAgent-Base, valid	14.7	0.0	0.0	14.6	0.0	1.0	0.0	**0.0**
InjecAgent-Enhanced, valid	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**0.0**
ASB, overall	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**0.0**
AgentDyn, overall	25.0	0.0	1.7	23.3	1.7	0.0	0.0	**0.0**
PASB, overall	41.4	31.0	0.0	41.4	0.0	39.7	0.0	**0.0**
MCPTox, valid	5.2	0.8	0.0	5.4	0.0	0.8	0.0	**0.0**
MSB, official macro	24.8	7.1	4.5	0.0	5.8	0.0	0.0	**0.0**
Native utility and refusal (%)								
WASP, utility ↑	83.3	83.3	83.3	95.8	79.2	66.7	87.5	83.3
AgentDojo, utility ↑	76.8	13.6	64.6	76.8	65.2	13.6	66.2	67.2
AgentDyn, utility ↑	66.7	0.0	36.7	66.7	36.7	0.0	31.7	33.3
ASB, utility ↑	0.0	0.0	0.0	0.0	30.0	0.0	0.0	30.0
ASB, refusal ↓	39.3	41.8	38.6	39.6	37.1	43.2	41.8	40.7
PASB, IPI utility ↑	100.0	0.0	100.0	100.0	100.0	0.0	100.0	100.0
MCPTox, refusal ↓	42.2	78.4	99.3	0.8	98.5	46.5	1.6	2.3
MSB, PUA ↑	48.3	52.5	55.8	58.2	56.6	54.6	59.0	**62.0**
MSB, NRP ↑	36.3	48.8	53.3	58.2	53.3	54.6	59.0	**62.0**

In the ablation study, every C-containing arm closes PASB, unlike arms without C. B alone leaves residuals on AgentDojo, AgentDyn, InjecAgent-Base, and MCPTox, but every B-containing arm closes MSB-Call, against 53.8% to 61.5% without B. P alone reduces ASRs through suppression but leaves PASB and MSB macro residuals. ASB and InjecAgent-Enhanced are zero across arms and identify no layer. Several reduced arms tie A7's floor and none beats it in any security row; the full combination has no demonstrated unique security benefit. Adding B to C or P+C lowers MCPTox refusal at unchanged zero valid-case ASR. C-containing arms partly recover P-only utility losses on AgentDojo, AgentDyn, and PASB IPI. Paired A0-to-A7 AgentDyn utility falls from 66.7% to 33.3%; the full-benchmark three-point bound does not hold on this subset. ASB utility is implementation-sensitive (Section F). AgentDyn has 5842 nonempty candidate cuts in 6647 guarded-call records but no executed nonterminal rewrite; utility recovers through C restoring P's terminal blocks.

6 Conclusion

When admission evidence cannot distinguish behavior-changing specifications, reviewing an agent artifact before execution cannot justify relaxing mediation at the affected egress site. The calls it induces still require a runtime decision. **PACE** moves the decision to the last enforceable tool boundary, where a concrete proposal, authenticated authority, runtime provenance, and a working hook are all available at once. Its certified contract combines represented-path separation, final-effect authorization, and executor grounding. The evaluated configuration retains effect authorization on restored calls, while path separation depends on the final intervention. On eight full benchmarks it lowers attack success while losing at most three points of native utility relative to the undefended agent, and the paired ablation shows the two checks are not interchangeable: effect verification carries most of the security, boundary adaptation is what prevents indiscriminate refusal. The reduced-scale adaptive search obtains no success on 30 out-of-authority targets, within its 50-episode budget per case. Extending coverage to semantic restatement, fragmented flows, hidden secondary effects, and durable distributed execution is future work.

AI USE STATEMENT

We used generative AI tools to check grammar and spelling, revise wording and organization, and examine the manuscript's logical consistency. These tools also assisted in reviewing mathematical derivations, proof arguments, and theoretical analyses for internal consistency. No generative-AI system is a component of **PACE**'s mediation logic; the evaluated agents and several baselines use models. We take responsibility for the final text, mathematical derivations, theoretical claims, experimental results, and artifacts.

ETHICS STATEMENT

This work studies attacks on deployed LLM agents and therefore carries dual-use risk. The attacks use published benchmarks and search-generated variants, executed only inside the corresponding benchmark sandboxes. The released artifact replaces harmful payloads with inert local sinks and deterministic fixtures that preserve path structure without providing working exfiltration targets (Section E.6). No human subjects, no personal data, and no production system were involved: every "secret" in our evaluation is a synthetic credential minted for the run. The graph construction makes the verifier's represented channels explicit; Section 4.4 identifies the dependencies it does not cover. This distinction matters when interpreting the security claims because a path outside the schema cannot enter a certified cut. We report the coverage boundary alongside the formal guarantees, and the released artifact uses inert local sinks and deterministic fixtures so readers can inspect the graph construction without access to working exfiltration targets or production systems.

REPRODUCIBILITY STATEMENT

The security model, the mediator, and the guarantee are specified so that a reader can check them independently. Sections 2.1 and 2.2 fix the execution model, the attacker, and the target property; Section 4 gives the four phases, the binding rule, the capacity function, and the decision rule. The runtime results carry numbered assumptions: H1 to H5 open Section D, and the protocol assumptions A1 to A9 sit in Section C.3. The admission-time results of Section 3 instead carry named conditions: the δ-achieving conservative transformation, C1, and C2 in Section 3; almost-sure halting in Section C.1; and the hypotheses of Proposition D.1 with that proposition. Proposition 1 carries its own four assumptions in its statement. Section C.4 lists the ten verifier conditions in the order the implementation checks them, and Section D contains proofs of the formal claims in the main text. Section C.6 states the premises no automatic check reaches, and Section C.3 marks the hardening protocol that lies outside the evaluated artifact. For the empirical side, Section E.1 fixes every configuration value before the primary run, Sections E.2 and E.3 record scaffolds, evaluators, denominators, and the ablation arms, Section E.5 records baseline port fidelity, Section E.4 fixes the statistical protocol and the multiple-comparison policy, and Section E.6 describes the machine-readable manifest, the deterministic replay procedure, and the release checklist. Anonymized source code and the frozen manifest accompany the submission as supplementary material.

REFERENCES

Hengyu An, Jinghuai Zhang, Tianyu Du, Chunyi Zhou, Qingming Li, Tao Lin, and Shouling Ji. IPIGuard: A novel tool dependency graph-based defense against indirect prompt injection in LLM agents. In *Proceedings of EMNLP*, pp. 1023–1039, 2025.

James P. Anderson. Computer security technology planning study. Technical Report ESD-TR-73-51, Electronic Systems Division, Air Force Systems Command, 1972.

Gilles Barthe, Pedro R. D'Argenio, and Tamara Rezk. Secure information flow by self-composition. In *Proceedings of Workshop on CSF*, pp. 100–114, 2004.

Sizhe Chen, Julien Piet, Chawin Sitawarin, and David Wagner. Struq: Defending against prompt injection with structured queries. In *Proceedings of USENIX Security*, pp. 2383–2400, 2025a.

Sizhe Chen, Arman Zharmagambetov, David Wagner, and Chuan Guo. Meta SecAlign: A secure foundation LLM against prompt injection attacks. *arXiv preprint arXiv:2507.02735*, 2025b.

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

Sahana Chennabasappa, Cyrus Nikolaidis, Daniel Song, David Molnar, Stephanie Ding, et al. LlamaFirewall: An open source guardrail system for building secure AI agents. *arXiv preprint arXiv:2505.03574*, 2025.

Michael R. Clarkson and Fred B. Schneider. Hyperproperties. *J. Comput. Secur.*, 18(6):1157–1210, 2010.

Manuel Costa, Boris Köpf, Aashish Kolluri, Andrew Paverd, Mark Russinovich, Ahmed Salem, Shruti Tople, Lukas Wutschitz, and Santiago Zanella-Béguelin. Securing AI agents with information-flow control. *arXiv preprint arXiv:2505.23643*, 2025.

Edoardo Debenedetti, Jie Zhang, Mislav Balunović, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In *Proceedings of NeurIPS, Datasets and Benchmarks Track*, 2024.

Edoardo Debenedetti, Ilia Shumailov, Tianqi Fan, Jamie Hayes, Nicholas Carlini, Daniel Fabian, Christoph Kern, Chongyang Shi, Andreas Terzis, and Florian Tramèr. Defeating prompt injections by design. *arXiv preprint arXiv:2503.18813*, 2025.

Dorothy E. Denning. A lattice model of secure information flow. *Commun. ACM*, 19(5):236–243, 1976.

Zenghao Duan, Yuxin Tian, Zhiyi Yin, Liang Pang, Jingcheng Deng, Zihao Wei, Shicheng Xu, Yuyao Ge, and Xueqi Cheng. SkillAttack: Automated red teaming of agent skills through attack path refinement. *arXiv preprint arXiv:2604.04989*, 2026.

William Enck, Peter Gilbert, Byung-Gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N. Sheth. TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones. In *Proceedings of OSDI*, pp. 393–407, 2010.

Ivan Evtimov, Arman Zharmagambetov, Aaron Grattafiori, Chuan Guo, and Kamalika Chaudhuri. WASP: Benchmarking web agent security against prompt injection attacks. *arXiv preprint arXiv:2504.18575*, 2025.

Linfeng Fan, Ziwei Li, Yuan Tian, Yichen Wang, Rongsheng Li, and Xiong Wang. The granularity mismatch in agent security: Argument-level provenance solves enforcement and isolates the LLM reasoning bottleneck. *arXiv preprint arXiv:2605.11039*, 2026.

Joseph A. Goguen and José Meseguer. Security policies and security models. In *Proceedings of S&P*, pp. 11–20, 1982.

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 ACM Workshop on AIS*, pp. 79–90, 2023. doi: 10.1145/3605764.3623985.

Kevin W. Hamlen, Greg Morrisett, and Fred B. Schneider. Computability classes for enforcement mechanisms. *ACM Trans. Program. Lang. Syst.*, 28(1):175–205, 2006.

Keegan Hines, Gary Lopez, Matthew Hall, Federico Zarfati, Yonatan Zunger, and Emre Kiciman. Defending against indirect prompt injection attacks with spotlighting. *arXiv preprint arXiv:2403.14720*, 2024.

Xinyi Hou, Yanjie Zhao, Shenao Wang, and Haoyu Wang. Model context protocol (MCP): Landscape, security threats, and future research directions. *ACM Trans. Software Eng. Method.*, 35(10):1–37, 2026.

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

Feiran Jia, Tong Wu, Xin Qin, and Anna Squicciarini. The task shield: Enforcing task alignment to defend against indirect prompt injection in LLM agents. *arXiv preprint arXiv:2412.16682*, 2024.

Huihao Jing, Haoran Li, Wenbin Hu, Qi Hu, Heli Xu, Tianshu Chu, Peizhao Hu, and Yangqiu Song. MCIP: Protecting MCP safety via model contextual integrity protocol. *arXiv preprint arXiv:2505.14590*, 2025.

Vasileios P. Kemerlis, Georgios Portokalidis, Kangkook Jee, and Angelos D. Keromytis. libdft: Practical dynamic data flow tracking for commodity systems. In *Proceedings of ACM SIGPLAN/SIGOPS*, pp. 121–132, 2012.

Jr. L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Ca. J. Math.*, 8:399–404, 1956.

Hao Li, Xiaogeng Liu, Hung-Chun Chiu, Dianqi Li, Ning Zhang, and Chaowei Xiao. DRIFT: Dynamic rule-based defense with injection isolation for securing LLM agents. *arXiv preprint arXiv:2506.12104*, 2025a.

Hao Li, Xiaogeng Liu, Ning Zhang, and Chaowei Xiao. PIGuard: Prompt injection guardrail via mitigating overdefense for free. In *Proceedings of ACL*, pp. 30420–30437, 2025b.

Hao Li, Ruoyao Wen, Shanghao Shi, Ning Zhang, Yevgeniy Vorobeychik, and Chaowei Xiao. AgentDyn: Are your agent security defenses deployable in real-world dynamic environments? *arXiv preprint arXiv:2602.03117*, 2026a.

Xu Li, Simon Yu, Minzhou Pan, Yiyou Sun, Bo Li, Dawn Song, Xue Lin, and Weiyan Shi. Unsafer in many turns: Benchmarking and defending multi-turn safety risks in tool-using agents. *arXiv preprint arXiv:2602.13379*, 2026b.

Jay Ligatti, Lujo Bauer, and David Walker. Edit automata: Enforcement mechanisms for run-time security policies. *Int. J. Inf. Secur.*, 4(1–2):2–16, 2005.

Yupei Liu, Yuqi Jia, Runpeng Geng, Jinyuan Jia, and Neil Zhenqiang Gong. Formalizing and benchmarking prompt injection attacks and defenses. In *Proceedings of USENIX Security*, pp. 1831–1847, 2024.

Yupei Liu, Yuqi Jia, Jinyuan Jia, Dawn Song, and Neil Zhenqiang Gong. DataSentinel: A game-theoretic detection of prompt injection attacks. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2025. doi: 10.1109/SP61157.2025.00250.

Llama Team. Llama prompt guard 2 model card, 2025. URL `https://github.com/meta-llama/PurpleLlama/blob/main/Llama-Prompt-Guard-2/86M/MODEL_CARD.md`.

Andrew C. Myers. JFlow: Practical mostly-static information flow control. In *Proceedings of ACM SIGPLAN-SIGACT*, pp. 228–241, 1999.

Praneeth Narisetty, Shiva Nagendra Babu Kore, Uday Kumar Reddy Kattamanchi, and Jayaram Kumarapu. Adaptive evaluation of out-of-band defenses against prompt injection in LLM agents. *arXiv preprint arXiv:2606.26479*, 2026.

Milad Nasr, Nicholas Carlini, Chawin Sitawarin, Sander V. Schulhoff, Jamie Hayes, Michael Ilie, Juliette Pluto, Shuang Song, Harsh Chaudhari, Ilia Shumailov, Abhradeep Guha Thakurta, Kai Yuanqing Xiao, Andreas Terzis, and Florian Tramèr. The attacker moves second: Stronger adaptive attacks bypass defenses against LLM jailbreaks and prompt injections. In *Proceedings of USENIX Security*, pp. 1467–1486, 2026.

George C. Necula. Proof-carrying code. In *Proceedings of ACM SIGPLAN-SIGACT*, pp. 106–119, 1997.

Nils Palumbo, Sarthak Choudhary, Jihye Choi, Guy Amir, Prasad Chalasani, and Somesh Jha. Formal policy enforcement for real-world agentic systems. *arXiv preprint arXiv:2602.16708*, 2026.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive APIs. In *Proceedings of NeurIPS*, 2024.

Fábio Perez and Ian Ribeiro. Ignore previous prompt: Attack techniques for language models. In *Proceedings of NeurIPS Workshop*, 2022.

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 *Proceedings of ICLR*, 2024.

Traian Rebedea, Razvan Dinu, Makesh Narsimhan Sreedhar, Christopher Parisien, and Jonathan Cohen. NeMo guardrails: A toolkit for controllable and safe LLM applications with programmable rails. In *Proceedings of EMNLP*, pp. 431–445, 2023.

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 *Proceedings of ICLR*, 2024.

Andrei Sabelfeld and Andrew C. Myers. Language-based information-flow security. *IEEE J. Select. Areas Commun.*, 21(1):5–19, 2003.

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

Igor Santos-Grueiro. Context-to-execution integrity for LLM agents. *arXiv preprint arXiv:2607.06000*, 2026.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Proceedings of NeurIPS*, 2023.

Fred B. Schneider. Enforceable security policies. *ACM Trans. Inf. Syst. Secur.*, 3(1):30–50, 2000.

R. Sekar, V. N. Venkatakrishnan, Samik Basu, Sandeep Bhatkar, and Daniel C. DuVarney. Model-carrying code: A practical approach for safe execution of untrusted applications. In *Proceedings of SOSP*, pp. 15–28, 2003.

Tianneng Shi, Jingxuan He, Zhun Wang, Hongwei Li, Linyu Wu, Wenbo Guo, and Dawn Song. Progent: Securing AI agents with privilege control. *arXiv preprint arXiv:2504.11703*, 2025.

Tachio Terauchi and Alex Aiken. Secure information flow as a safety problem. In *Proceedings of SAS*, pp. 352–367, 2005.

Lillian Tsai and Eugene Bagdasarian. Contextual agent security: A policy for every purpose. In *Proceedings of the Workshop on Hot Topics in Operating Systems (HotOS)*, pp. 8–17, 2025.

Eric Wallace, Kai Xiao, Reimar Leike, Lilian Weng, Johannes Heidecke, and Alex Beutel. The instruction hierarchy: Training LLMs to prioritize privileged instructions. *arXiv preprint arXiv:2404.13208*, 2024.

Peiran Wang, Yang Liu, Yunfei Lu, Yifeng Cai, Hongbo Chen, Qingyou Yang, Jie Zhang, Jue Hong, and Ye Wu. AgentArmor: Enforcing program analysis on agent runtime trace to defend against prompt injection. *arXiv preprint arXiv:2508.01249*, 2025.

Peiran Wang, Ying Li, and Yuan Tian. Aligning provenance with authorization: A dual-graph defense for LLM agents. *arXiv preprint arXiv:2605.26497*, 2026a.

Yuhang Wang, Feiming Xu, Zheng Lin, Guangyu He, Yuzhe Huang, Haichang Gao, Zhenxing Niu, Shiguo Lian, and Zhaoxiang Liu. From assistant to double agent: Formalizing and benchmarking attacks on OpenClaw for personalized local AI agent. *arXiv preprint arXiv:2602.08412*, 2026b.

Zhiqiang Wang, Yichao Gao, Yanting Wang, Suyuan Liu, Haifeng Sun, Haoran Cheng, Guanquan Shi, Haohua Du, and Xiangyang Li. MCPTox: A benchmark for tool poisoning on real-world MCP servers. In *Proceedings of AAAI*, pp. 35811–35819, 2026c.

Yuhao Wu, Franziska Roesner, Tadayoshi Kohno, Ning Zhang, and Umar Iqbal. IsolateGPT: An execution isolation architecture for LLM-based agentic systems. In *Network and Distributed System Security Symposium*, 2025.

Yixuan Yang, Cuifeng Gao, Daoyuan Wu, Yufan Chen, Yingjiu Li, and Shuai Wang. MCPSecBench: A systematic security benchmark and playground for testing model context protocols. *arXiv preprint arXiv:2508.13220*, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *Proceedings of ICLR*, 2023.

Jingwei Yi, Yueqi Xie, Bin Zhu, Emre Kiciman, Guangzhong Sun, Xing Xie, and Fangzhao Wu. Benchmarking and defending against indirect prompt injection attacks on large language models. In *Proceedings of ACM SIGKDD*, 2025.

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

Qiusi Zhan, Richard Fang, Henil Shalin Panchal, and Daniel Kang. Adaptive attacks break defenses against indirect prompt injection attacks on LLM agents. In *Proceedings of NAACL (Findings)*, pp. 7116–7132, 2025.

Dongsen 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 *Proceedings of ICLR*, 2026.

Hanrong Zhang, Jingyuan Huang, Kai Mei, Yifei Yao, Zhenting Wang, Chenlu Zhan, Hongwei Wang, and Yongfeng Zhang. Agent security bench (ASB): Formalizing and benchmarking attacks and defenses in LLM-based agents. In *Proceedings of ICLR*, 2025.

Wei Zhao, Zhe Li, Peixin Zhang, and Jun Sun. ClawGuard: A runtime security framework for tool-augmented LLM agents against indirect prompt injection. *arXiv preprint arXiv:2604.11790*, 2026.

Peter Yong Zhong, Siyuan Chen, Ruiqi Wang, McKenna McCall, Ben L. Titzer, Heather Miller, and Phillip B. Gibbons. RTBAS: Defending LLM agents against prompt injection and privacy leakage. *arXiv preprint arXiv:2502.08966*, 2025.

Kaijie Zhu, Xianjun Yang, Jindong Wang, Wenbo Guo, and William Yang Wang. MELON: Provable defense against indirect prompt injection attacks in AI agents. In *Proceedings of ICML*, 2025.

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

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

Appendix of PACE

CONTENTS

A Notations **17**

B Related Work **19**

B.1 Agent Attack Surfaces 19

B.2 Agent Defense Paradigms 19

B.3 Formal Foundations 20

C Protocol and Verification **20**

C.1 Measure-Theoretic Setup 20

C.2 Graph Fold and Emission Rules 20

C.3 Deployment Hardening: Authenticated Execution Protocol 21

C.4 The Recomputing Verifier 22

C.5 Details Deferred from the Main Text 23

C.6 Scope and Residual Assumptions 24

D Proofs **24**

D.1 Admission-Time Results 25

D.2 Certified Separation 26

D.3 The Cut: Closed Form 26

D.4 Coverage and Simulation on the Transparent Fragment 27

D.5 Freshness, Authentication, and Anti-Replay 28

D.6 Grounded Confinement and the Episode Invariant 29

E Experimental Details **30**

E.1 Frozen Hyperparameters 30

E.2 Benchmarks, Adapters, and Denominators 30

E.3 Ablation Design 32

E.4 Statistical Analysis 32

E.5 Baseline Port Fidelity 33

E.6 Reproducibility Manifest and Replay 33

E.7 Cost, Release Checklist, and Artifact Safety 34

F Extra Experiments **34**

F.1 Per-Group Benchmark Results 34

F.2 Ablation Notes 34

A NOTATIONS

Table 3: List of notations and their descriptions used in the paper.

Notation	Description
Agent execution and threat model (Section 2.1)	
M	Frozen policy that alternates free text with tool calls
$\mathcal{T}$	Registered tool set
a_t	Tool call proposed at step t
o_t	Observation returned by a_t
r	Answer returned to the user, itself an egress site
$\tau = \big((a_t, o_t)_t, r\big)$	Trajectory of one episode
$\mathcal{X}$	Standard Borel space of trajectories
$K = (NL, code)$	Artifact: a skill, tool description, or MCP schema
$c \in \mathsf{Ctx}$	Context: request, injected text, tool returns, state, secret
s	Secret carried by the context
$\bar{c} \in \overline{\mathsf{Ctx}}$	Public context with a designated secret slot
$\mu_{K,c} = \mathsf{Traj}(M, K, c)$	Trajectory measure induced by K under context c
$\mu^{\Gamma}_{K,c}$	Deployed trajectory measure, with M run behind mediator Γ
$\mathsf{Beh}(K) = \{\mu_{K,c}\}$	Behavior family: the security-relevant object, not K itself
Security property (Section 2.2, Section C.1)	
$\mathsf{egress}(\tau)$	Egress projection: ordered pairs (ω, bytes) leaving the deployment
Ω	Finite set of egress sites
$\mathcal{B}$, $\mathcal{B}^*$	Standard Borel byte space; $\mathcal{B}^* = \bigsqcup_{n\geq 0} \mathcal{B}^n$
$\mathcal{E}$	Codomain of egress, $\bigsqcup_{n\geq 0}(\Omega \times \mathcal{B})^n$
ω	A single egress site
pr_ω	Subsequence extractor selecting the entries tagged ω
$O_\omega = \mathrm{pr}_\omega \circ \mathsf{egress}$	Egress observed at site ω, or $\perp$ if never reached
$\hat{O}_\omega$	Value the code emits at ω before the mediator acts
$d(P, Q)$	Total variation, $\sup_A \lvert P(A) - Q(A)\rvert$ over measurable A
δ	Indistinguishability budget
$\Phi_{\mathrm{NI}}(K, \Gamma; \delta)$	Global probabilistic-noninterference criterion (Equation (1))
$\Phi_{\mathrm{site}}(K, \Gamma, \omega; \delta)$	Per-site restriction of Φ_{NI} (Equation (2))
$\Delta_\Gamma(K, \bar{c}, \omega)$	Secret-induced distinguishability at ω under context $\bar{c}$
Gates and admission-time evidence (Section 3)	
Γ	Mediator: the deployed transformation applied at each egress site
g, Γ_g	Gate mapping a judgment to $\{\mathsf{mediate}, \mathsf{relax}\}$; the mediator it induces
Γ_0, Γ_0^{r}	Reference profile mediating every site but ω; its relaxing extension
$\mathcal{J}_i(K, \omega)$	Judgment held by admission route $i \in \{\mathrm{I}, \mathrm{II}, \mathrm{III}\}$
$\mathcal{V}(K, \omega)$	Variant class: same code, interface, and site; different specifications
C1, C2	Judgment-ambiguity and attacker-witness conditions of Theorem 1
ϕ	Feature map through which a Route-III text extractor factors
v	Slice inputs supplied to site ω along a trajectory
F, F_0	Code-slice map $\hat{O}_\omega = F(v, s)$; its secret-independent common value
med	Mediator action at ω, a measurable map of the call alone
Evidence graph and phase 1 (Section 4.1, Section C.2)	
H_t	Evidence graph: multigraph folded from the append-only log
G_t^-	Active graph: committed edges at σ_t plus the proposed edges of a_t
E^{fl}	Flow subgraph; the only edge types reachability uses
E1–E5	The five flow-edge emission rules
q_t	Fresh, globally unique attempt identifier for a_t
ι_t	External idempotency key, preserved across retries
$\mathrm{op}(q_t)$, $T(q_t)$	Operation node and call-specific terminal of attempt q_t
T_{bad}	Set of terminals labeled unsafe by the versioned effect rule

Table 3 (continued)

Notation	Description
R_{v+1}	Resource version node minted by a confirmed declared write
ν	Provenance map over live values
σ_t	Execution version: committed edges, labels, and live-value set
ε_t	Policy epoch: schemas, effect rules, costs, contracts, adapter code
L_t^{p}	Canonical `PROPOSE` record of attempt q_t
Cut and certification, phase 2 (Section 4.2, Section C.4)	
P_t	Protected sources: data and resource nodes with protection labels
Ψ	Finite family of deterministic re-encodings used by the derivation oracle
$\mathrm{match}(y, x)$, canon_ψ	Oracle match predicate (Equation (4)) and per-form canonicalization
S_t	Untrusted-origin entries of the live-value set at σ_t
B_t	Untrusted frontier: closure of S_t over committed flow edges
$U_t = \{T(q_t)\} \cap T_{\mathrm{bad}}$	Unsafe terminal of the pending call, empty for a safe call
$\mathrm{unsafe}_{\varepsilon_t}(a_t)$	Registry predicate placing $T(q_t)$ in T_{bad}
$\mathsf{Impl}(e)$, $\mathsf{Sev}(e)$	Actions the adapter implements on e; the severing subset
$\mathrm{cost}(a, e)$	Declared per-edge cost of action a, a strictly positive rational
$\beta_t(e)$	Canonical binding: least-cost action in $\mathsf{Sev}(e) \cap \mathsf{Impl}(e)$, or $\perp$
$\kappa(e)$	Edge capacity (Equation (6)); finite only on bound proposed edges
A, e_{T}	Argument edges of a_t; its terminal edge $\mathrm{op}(q_t) \to T(q_t)$
R, $A^\star$	Nodes reachable from $B_t \cup P_t$ in the committed flow subgraph; $\{e \in A : \mathsf{src}(e) \in R\}$
$\mathcal{C}_t$	Canonical minimum cut: cheapest, then lexicographically least
$\mathcal{M}_t$	Enforcement manifest, one bound hook per cut-edge instance
$\mathrm{arg}(e)$, $\mathrm{hook}(e)$	Registry-bound argument position and executor hook for edge e
$\mathrm{infl}(P_t, a_t)$	Influence test: a protected node reaches $\mathrm{op}(a_t)$, or the oracle matches
$\mathrm{rev}(a_t)$, $\mathrm{auth}(a_t)$	Adapter reversibility flag; externally authenticated authorization flag; $\mathrm{low}(a_t)$ additionally requires empty sinks and no persistent effects
$\mathrm{dec}(a_t)$	Decision rule (Equation (7))
R1–R6	Refuse on undefined schema; R2 low-impact release; R3 authenticated, uninfluenced, C-certified release; refuse unauthorized original effects; refuse uncuttable or inadmissible manifests; refuse unauthorized final effects in the certified contract
$\mathrm{eff}(a_t)$	Effect check: one when every effect atom of a_t is certified
cert_t	Path certificate $(q_t, L_t^{\mathrm{p}}, \sigma_t, \varepsilon_t, B_t \cup P_t, U_t, \mathcal{C}_t, \mathcal{M}_t)$
n_t, n^*, h^{auth}	Certificate sequence number; verifier counter and authorization head
tok_t	One-shot authorization token of the hardened protocol (Equation (9))
MAC, h, k, k'	MAC and hash; log key and token key, held by disjoint component pairs
Enforcement protocol, phases 3–4 (Section 4.3, Section C.3)	
z_j	Tag of log record j in the append-only MAC chain (Equation (8))
$\mathrm{res} \in \{\perp\} \cup \mathcal{Q}$	Global single-flight reservation; $\mathcal{Q}$ is the space of attempt identifiers
$\mathrm{Apply}_{\varepsilon_t}$	Deterministic canonical rewrite of the proposal under $\mathcal{M}_t$
$E^{\mathrm{o}}(q_t, o_t)$	Canonical output edge set bound into an outcome receipt
A1–A9	Protocol assumptions: cryptography, atomicity, durability, state currency
Causal chains and guarantees (Section 4.4, Section D)	
$\gamma = x_0 \to \cdots \to \zeta_t$	Real causal chain ending at the pending effect
x_i, ζ_t	Values along the chain; the pending unsafe effect
w_i, w_i'	Calls realizing link $x_i \to x_{i+1}$ (writer and reader in the resource case)
α_{D}, α_{T}	Simulation maps from chain values to data nodes and from the effect to a terminal
π_i, Π	Emitted segment of link i; their concatenation, a flow path in G_t^-
H1–H5	Implementation conditions; H3 splits into H3a, H3b, H3c by label kind
Configurations	
PACE-C	Certified contract: no R2/R3 releases or restoration of blocks; the installed action preserves the cut and every dispatched final call passes R6

Table 3 (continued)

Notation	Description
PACE-P	Evaluated configuration: evaluates all nine obligations; R3 requires all to pass, while R2 can release effects labelled reversible with empty sinks and no persistence despite failures. C can restore P's blocks, and declared repairs are dispatched without R6

B Related Work

B.1 Agent Attack Surfaces

Tool-integrated agents process external text in the same autoregressive context that produces actions, so an observation can redirect execution. InjecAgent separates direct user harm from private-data exfiltration (Zhan et al., 2024), AgentDojo supplies executable user tasks with security targets and adaptive interfaces (Debenedetti et al., 2024), ASB broadens the surface to system, user, tool, and memory stages (Zhang et al., 2025), and WASP separates partial compromise from terminal attacker success in multi-step web tasks (Evtimov et al., 2025). Those distinctions are why we report native metrics instead of one pooled attack-success label.

Reusable artifacts differ from a single malicious page: the artifact persists, can carry code, and defines a workflow. MCPTox poisons the tool metadata of real MCP servers and reports high attack success with near-zero refusal (Wang et al., 2026c), MCPSecBench organizes the protocol's attack surface into a reproducible benchmark (Yang et al., 2025), the MCP lifecycle survey describes post-trust server changes (Hou et al., 2026), while SkillAttack probes reusable skills through adversarial prompts (Duan et al., 2026). Trust inversion arises when an attacker-authored description informs admission and shapes later execution. We therefore evaluate prose, code-carried, and cross-turn payloads separately.

Persistent state defeats a single-turn monitor, because source and sink can fall in different turns. AgentPoison optimizes triggers that retrieve poisoned memories (Chen et al., 2024), ASB includes a memory-poisoning stage (Zhang et al., 2025), and PASB exercises prompt, tool, and memory-retrieval attacks in a personalized local agent (Wang et al., 2026b). Our reduced-scale adaptive search tests out-of-authority AgentDojo goals (Sections E.2 and F). Adaptive evaluation can change conclusions drawn from native attacks (Zhan et al., 2025; Nasr et al., 2026); out-of-band defenses also have a dedicated protocol and a small-scale adaptive evaluation of Progent (Narisetty et al., 2026).

B.2 Agent Defense Paradigms

CaMeL extracts control and data flow from a trusted request and wraps a system layer around a susceptible model (Debenedetti et al., 2025). FIDES formalizes the properties dynamic taint tracking can enforce and builds an IFC planner with confidentiality and integrity labels (Costa et al., 2025). RTBAS adapts IFC to tool-based agents and falls back on user confirmation when automated screening cannot establish integrity or confidentiality (Zhong et al., 2025). CaMeL derives its plan from a trusted request, while FIDES uses trusted wrappers and flow labels. The deployments we target also expose attacker-authored artifact descriptions to the planner. Our comparison uses instrumented IFC implementations where their required hooks are available, preserving the scope of their flow-control mechanisms.

Progent expresses least privilege as symbolic rules over tool names and arguments (Shi et al., 2025), Task Shield checks whether instructions and calls serve the user's stated task (Jia et al., 2024), and ClawGuard enforces user-confirmed rules at tool boundaries across web, local, MCP, and skill channels (Zhao et al., 2026). IPIGuard uses a tool dependency graph (An et al., 2025), Conseca derives contextual policies (Tsai & Bagdasarian, 2025), and FORGE enforces formal policies over tool operations (Palumbo et al., 2026). These approaches share **PACE**'s concern with action authorization and differ in the evidence used. Their rules range over names, arguments, and task descriptions; **PACE** derives a separating cut from an execution graph and then demands a concrete hook binding for every cut edge.

MELON detects indirect injection by comparing the original action with a masked-prompt re-execution (Zhu et al., 2025), DataSentinel trains an adaptive detector through a minimax objective (Liu et al., 2025), and LlamaFirewall combines prompt, agent-alignment, and code guardrails (Chennabasappa et al., 2025). Structured-query and model-level defenses separate instructions from data or train the model to respect the distinction (Chen et al., 2025a;b). All of these compose with **PACE**, and our evaluation keeps a pre-filter's AUROC, TPR, and FPR apart from end-to-end agent security.

AgentArmor reconstructs runtime traces as program graphs and applies a type system to security metadata (Wang et al., 2025). More recent monitors operate at argument and execution granularity: PACT supports fine-grained provenance and scoped discharge certificates (Fan et al., 2026), AuthGraph separates observed provenance from request-derived authorization (Wang et al., 2026a), and CXI binds executed effects to a canonical manifest and requires fresh authority after relevant edits or repairs (Santos-Grueiro, 2026). Thus fine-grained authorization and re-authorization alone do not distinguish **PACE**. Its analysis couples a candidate graph cut to the intervention actually installed: separation survives exactly when that intervention blocks the terminal or severs every reachable argument edge, alongside authorization of the final call (Sections D.3 and 4.4). The evaluated restoration and repair paths make this distinction observable in the component study. IsolateGPT establishes isolation between LLM applications and the host (Wu et al., 2025), a separate boundary from these call-level checks.

B.3 Formal Foundations

Complete mediation requires every security-relevant access to pass through a tamper-resistant, analyzable mechanism (Saltzer & Schroeder, 1975; Anderson, 1972), and runtime policy enforcement characterizes which trace properties execution monitors can enforce (Schneider, 2000). **PACE** is an edit automaton rather than a truncation automaton, since its actions rewrite and suppress arguments as well as halting the call, and that class is strictly more expressive (Ligatti et al., 2005; Hamlen et al., 2006). Lattice-based confidentiality (Denning, 1976) and language-level information-flow control (Myers, 1999; Sabelfeld & Myers, 2003) give the basis for source and sink labels, and TaintDroid and libdft show that labels do propagate through instrumented execution (Enck et al., 2010; Kemerlis et al., 2012). Our opaque-boundary ablation is built to respect that capability, not to deny it.

Proof-carrying code attaches machine-checkable evidence that code satisfies a policy (Necula, 1997), and model-carrying code checks a model of an untrusted program against a safety property (Sekar et al., 2003). Both apply when the code slice is the whole security-relevant semantics *and* the model's choice of that slice's inputs is itself secret-independent, which is the positive case of Proposition D.1. **PACE**'s runtime guarantee is deliberately weaker and more operational. The max-flow/min-cut theorem gives exact minimum-capacity separation for a fixed capacitated graph (L. R. Ford & Fulkerson, 1956), and our solver applies it directly; the registry costs are deterministic inputs, not calibrated probabilities.

C Protocol and Verification

C.1 Measure-Theoretic Setup

One episode gives a trajectory $\tau = \big((a_t, o_t)_t, r\big)$: the calls a_t proposed against the registered tool set $\mathcal{T}$, the observations o_t they return, and the answer r, which is itself an egress site. Trajectories live in a standard Borel space $\mathcal{T}$ and episodes halt almost surely, so egress is defined off a null set. Its codomain is $\mathcal{E} = \bigsqcup_{n\geq 0}(\Omega \times \mathcal{B})^n$, for the finite site set Ω and a standard Borel byte space $\mathcal{B}$, carrying the disjoint-union σ-algebra; egress is measurable for it. The per-site projection of Section 3 maps into $\mathcal{B}^* \sqcup \{\bot\}$, where $\mathcal{B}^* = \bigsqcup_{n\geq 0} \mathcal{B}^n$ is again standard Borel, and Lemma D.1 proves it measurable. The artifact K paired with a context c induces $\mu_{K,c} = \mathsf{Traj}(M, K, c)$ on $\mathcal{T}$, so the dependence on the frozen policy and the artifact is explicit; running M behind a mediator Γ gives $\mu^{\Gamma}_{K,c}$.

C.2 Graph Fold and Emission Rules

Parallel edge instances stay distinct in the multigraph. Besides the nodes of Definition 1 the fold also records the agent's stated intentions and plans, and the control and intent edges among them stay in

the audit record and never reach a cut. Values and effects connect through an operation node, so the construction has no direct `Data`-to-`Data` and no `Data`-to-`Terminal` edge; Section D.4 uses both facts.

The host maintains a provenance map ν for live values and applies five rules, which are the complete set of flow-edge emitters. (E1) Before certification, each schema-defined proposal argument emits an edge from its data or resource node to the fresh operation node. (E2) A valid outcome receipt emits an edge from the operation to every schema-defined returned value, minting a fresh provenance identifier unless the adapter certifies the value as an unchanged pass-through. (E3) Such a pass-through keeps its identifier at the next declared position. (E4) Every proposal creates a fresh terminal $T(q_t)$ and emits $\mathrm{op}(q_t) \to T(q_t)$ before dispatch, and the versioned unsafe-effect rule labels that terminal. (E5) A confirmed outcome of a schema-declared write mints a new resource version node R_{v+1}, commits $\mathrm{op}(q) \to R_{v+1}$, and gives R_{v+1} the union of the origin and protection labels of that call's committed inputs. Rules E2 and E5 fire at finalization, so their edges are committed and not proposed. These rules emit structure, not a claim about the semantics of hidden computation.

Instance status follows from the log: a refusal expires the current proposal edges, `OUTCOME` commits them together with the authenticated output edges, `UNKNOWN` commits the proposal alone, and `BLOCKED` or `CANCEL` commits no flow. The frontier B_t is folded as in Section 4.2 from the origin-labeled live-value set, over committed flow edges only, so $B_t \cap T_{\mathrm{bad}} = \emptyset$ and no node enters B_t through a current proposed edge; in particular the fresh operation and terminal of q_t stay out. The disjointness $P_t \cap U_t = \emptyset$ recorded in Section 4.2 follows for the same reason, since protection labels apply only to data and resource nodes.

Lemma C.1 (Every unsafe path crosses a current edge)**.** *Let $E^{\mathrm{p}}(q_t)$ be the proposed flow-edge instances of a_t. Every directed path from $B_t \cup P_t$ to nonempty U_t contains an edge in $E^{\mathrm{p}}(q_t)$. If the terminal in-edge admits `block_call`, that edge alone is a finite-capacity cut.*

Proof. Nonempty U_t contains only $T(q_t)$, which is fresh, disjoint from the source set, and by (E4) has the single proposed in-edge $\mathrm{op}(q_t) \to T(q_t)$. Every path to it therefore ends with that current edge, and removing it separates the terminal. When it has a nonempty binding to `block_call`, Equation (6) gives it a finite value. □

Declared costs are strictly positive rationals with a fixed denominator, which the implementation scales to integers. Positivity matters: with a zero-cost action a minimum-cost separating set could be padded arbitrarily, and the verifier's set equality in Section C.4 would be ill posed. The infinity sentinel exceeds the sum of every finite capacity, and acceptance separately rejects a cut containing it. Nodes and edge instances carry a canonical order with the terminal edge first, which is what makes the selection in Proposition D.2 single-valued. Builder and verifier therefore agree without floating-point arithmetic and without depending on which minimum cut a flow solver happens to return.

C.3 Deployment Hardening: Authenticated Execution Protocol

The evaluated prototype mediates one in-process execution path and invalidates a decision when the episode state it names changes. This subsection specifies the stronger protocol a distributed deployment needs, with durable records, a global reservation, and one-shot token spending. It is what discharges the freshness condition H5 of Section D outside a single process, and it is not part of the artifact we evaluate in Section 5.

Every record belongs to one append-only MAC chain in the style of tamper-evident audit logs. With a canonical, length-delimited encoding,

$$z_j = \mathsf{MAC}_k\big(z_{j-1} \| \mathrm{enc}(\mathrm{type}_j, q_j, \mathrm{payload}_j)\big). \tag{8}$$

A record contains its type, attempt identifier, payload, previous tag, and tag. Types are `PROPOSE`, `DECIDE`, `DISPATCH`, `OUTCOME`, `BLOCKED`, `CANCEL`, `UNKNOWN`, and `RECONCILE`. AcceptIfFresh is the primitive that checks Section C.4, moves the reservation from $\perp$ to q_t, advances the verifier sequence and authorization head, appends `DECIDE(accept)`, and returns

$$\mathsf{tok}_t = \mathsf{MAC}_{k'}\big(q_t \| n_t \| \sigma_t \| \varepsilon_t \| \mathsf{h}(L_t^{\mathrm{p}}) \| \mathsf{h}(\mathcal{M}_t)\big), \tag{9}$$

where n_t is the verifier's next sequence number and k' is shared only by the verifier and the executor. The log key k and the token key k' sit with different pairs of components, so neither the graph builder nor the certificate producer holds either one. The executor spends tok_t once by compare-and-swap, rechecks the versions, and holds each resource lease through the access. Its permissive counterpart RESERVEIFFRESH checks the same state and epoch for R2 and R3, takes the same reservation, and appends `DECIDE(release)` without a path certificate or token; R3 still requires C's nine obligations to pass. A stale proposal is refused, never silently rebuilt. For a nonblocking manifest, $\mathrm{Apply}_{\varepsilon_t}$ is the deterministic canonical adapter function that applies every certified rewrite to the call bytes in L_t^{p}; it preserves ι_t and every versioned resource identity, and it is undefined for a terminal block or an inadmissible manifest.

The remaining protocol content is the assumption list the proofs cite. Nine assumptions carry the protocol obligations of H5 in Section D, together with the currency obligation A9 supplies.

Cryptography. (A1) MAC is existentially unforgeable under chosen-message attack, with k held by the host and log store and k' only by the verifier and the executor, so neither the graph builder nor the certificate producer holds either key. (A2) h is collision resistant. (A3) Calls, manifests, and log payloads use a canonical length-delimited serialization, so the concatenations in Equations (8) and (9) parse uniquely.

Identity and atomicity. (A4) Attempt identifiers q_t are globally unique and never reused, while ι_t is a separate field that may repeat; the `PROPOSE` record stores the immutable call bytes, all resource versions, and ι_t, so ι_t is covered by $\mathsf{h}(L_t^{\mathrm{p}})$ and cannot change after certification. (A5) Log append, ACCEPTIFFRESH, RESERVEIFFRESH, FINALIZE, RECONCILE, epoch updates, and the executor's compare-and-swap and LEASEIFCURRENT are linearizable, and the appender and the four mediation primitives share one critical section. A successful reservation atomically appends its decision, and a failure records refusal only while the attempt is undecided. (A6) Five objects are distinct and rollback-protected: the raw log tip, which moves on every append and is therefore never the freshness anchor; the execution version σ_t, naming committed edges, execution-derived labels, and the live-value set; the verifier store (n^*, h^{auth}), which advances only on certificate acceptance; the policy epoch ε_t, naming an immutable snapshot of labels, schemas, effect rules, costs, contracts, and adapter code; and a live or halted mode together with one global reservation $\mathrm{res} \in \{\bot\} \cup \mathcal{Q}$, where $\mathcal{Q}$ is the space of attempt identifiers.

Durability and executor. (A7) Log appends are durable and write-ahead, so `DISPATCH` commits to stable storage before any externally visible send of that call and no send is reordered ahead of that commit. A trusted adapter returns `BLOCKED` only for a terminal `block_call` after enforcing non-dispatch, and an outcome receipt only when bound to $(q_t, \iota_t, o_t, \mathsf{h}(E^{\mathrm{o}}(q_t, o_t)))$, with E^{o} the canonical output edge set. (A8) PREPARE performs the dispatch sequence of Section 4.3 under one further discipline: it acquires the policy and resource leases and spends the token only after checking q_t, σ_t, ε_t, $\mathsf{h}(L_t^{\mathrm{p}})$, and $\mathsf{h}(\mathcal{M}_t)$ against the current stores and the immutable `PROPOSE` record. It sends nothing before that check, rolls back partial installation, and returns only `CANCEL`, a $(q_t, \mathsf{h}(\mathcal{M}_t), \mathtt{BLOCKED})$ receipt, or a $(q_t, \mathsf{h}(\mathcal{M}_t), \mathsf{h}(\mathrm{Apply}_{\varepsilon_t}(\mathcal{M}_t, L_t^{\mathrm{p}})), \mathtt{READY})$ receipt whose branch dispatches only those bytes and holds every resource lease through the certified access. (A9) Every event that could add a committed flow edge, change a label, or change the live-value set either passes the mediator or advances σ_t before the next certificate is issued. A resource adapter must supply a snapshot or a lease held through the certified access, or the position is schema-undefined. A9 is what makes σ_t a statement about the world and not only about the store, and an out-of-band write through a channel the deployment never registered violates it.

Finalization follows Section 4.3 and is atomic, with reconciliation opening a new attempt $q_{t'}$ that carries the same ι_t. Crash recovery uses the same durable intent: without `DISPATCH`, write-ahead ordering proves no send occurred and the attempt can be cancelled, while a dispatch without a receipt becomes `UNKNOWN`. Rule (E5) fires only on a confirmed outcome, so an `UNKNOWN` write mints no version node, and reconciliation must re-read the resource before any later step may use it. The protocol therefore gives one-shot token use, not exactly-once delivery without adapter support.

C.4 THE RECOMPUTING VERIFIER

Inside ACCEPTIFFRESH, the verifier folds the authenticated log and the current registry snapshot. It reconstructs G_t^-, B_t, P_t, U_t, flow-edge instances, bindings, capacities, and statuses, and reads no

graph, capacity, or verdict from the certificate producer. It accepts cert_t only if every check below passes.

1. The mode is live and $\mathrm{res} = \bot$.
2. n_t is the next verifier sequence, σ_t is current, and L_t^{p} is the complete immutable `PROPOSE` record of q_t with no decision for that attempt.
3. ε_t is the current policy epoch.
4. The operation and every argument are schema-defined. Immutable proposal bytes and resource versions agree with L_t^{p}. The verifier rebuilds B_t from the origin-labeled live-value set stored with σ_t, and rebuilds P_t from committed protection labels together with a rerun of Equation (4) on the frozen proposal.
5. The verifier reruns the unsafe-effect predicate to construct U_t, and every supplied cut edge is a proposed flow-edge instance of q_t.
6. For every proposed flow edge, it recomputes the canonical binding in Equation (5) and the capacity in Equation (6).
7. Removing $\mathcal{C}_t$ separates U_t from $B_t \cup P_t$, its capacity is finite and equals the declared cost, and an empty target carries the empty cut. The verifier checks the cut it is given rather than re-solving the flow problem. Proposition D.2 shows that an independent recomputation would cost one reachability pass, which we note as an available strengthening rather than an implemented one.
8. $\mathrm{dom}(\mathcal{M}_t) = \mathcal{C}_t$, with exactly one canonically encoded entry per cut-edge instance and no repeated edge.
9. Every cut edge has a nonempty binding, and every manifest entry contains that binding with the registry's argument position and hook.
10. The adapter declares the complete manifest jointly admissible: rewrites do not conflict, every hook is timely, and each severance contract is declared preserved under the complete set.

Condition 6 blocks substitution of a costlier action, condition 7 blocks a cut that does not separate or that misstates its cost, and conditions 8 and 9 block an apparently separating cut with missing bindings. Condition 10 checks a registry declaration and not its semantic truth, an obligation H4 carries at execution level. Failure returns $\bot$ without touching the reservation or the verifier stores, except that an undecided attempt receives one refusal record.

C.5 Details Deferred from the Main Text

The untrusted frontier. Let S_t hold the untrusted-origin entries of the live-value set at σ_t. The verifier computes B_t by closing S_t over committed flow edges and dropping terminals, which loses nothing since a terminal has no out-edge. Proposed edges never advance the closure. If they did, one untrusted argument would collapse every call into terminal blocking, since the proposal itself would carry the frontier to the operation node it just created.

Severance contracts. An action lies in $\mathsf{Sev}(e)$ when its warrants guarantee that, once installed, no execution of this call takes a real link whose represented segment contains e. The contract is about the call and not about the graph, so observation and user confirmation sever nothing: they change what the operator sees, not what the call can do. This is also why Sev is declared by the deployment while Impl is declared by the adapter. The adapter knows which hooks exist; only the deployment can assert that installing one removes a real dependency.

The two redaction warrants. Redaction on a data edge lies in $\mathsf{Sev}(e)$ only under two declarations. The adapter warrants that the emitted value is independent of the original in content, length, and structure, so a length-preserving mask does not qualify. The registry warrants, per operation and argument position, that the call's occurrence does not depend on that argument. Its default is that occurrence may depend, since no adapter knows whether the model would have proposed this call with a different argument. Miss either warrant and the argument edge has no severing action. The terminal edge then admits `block_call`, and a read-only resource view only when the registry certifies that the call's single unsafe effect is the write. This default is why **PACE** blocks at most sites rather than editing them.

C.6 Scope and Residual Assumptions

Four cases carry no structural link between a real dependence and the graph: undeclared reads, secondary external calls made inside a tool, strings recreated by the model, and external changes to resources that bypass versioned adapters. R1 refuses an undefined operation or argument but does not detect a hidden read inside an otherwise schema-defined tool. Out-of-band resource changes violate A9. The transform family of Equation (4) is $\Psi = \{$raw text, case normalization, whitespace normalization, base64, base64url, hex, percent encoding, the common digests, fixed-length prefixes$\}$, which narrows the third case to deterministic re-encodings; paraphrase, translation, arithmetic encodings, and short fragments still escape. It also errs in the other direction: short low-entropy values match benign text, so P_t grows spuriously and R3 inherits both behaviors, which is one reason that rule exists only in **PACE**-P.

The cost in Equation (6) is a declared per-edge surrogate and not episode utility, so one call-level block can cover several edges while the sum charges each. Minimality is therefore over per-edge-groundable sets rather than over jointly admissible ones, and a cheaper conflicting cut may coexist with a costlier compatible one, where the certified contract fails closed. Canonical binding followed by R6 is not a complete search over authorized interventions: a cheapest rewrite can fail a final argument constraint while another declared, severing rewrite satisfies it. The cut's cost optimality does not resolve that action-selection problem.

Request compilation, effect labels, validators, source labels, and severance contracts are semantic premises; runtime invocation counts do not establish them. The verifier shares the graph-fold code, the label rules, T_{bad}, and the registry with the builder, so the two agree on every semantic omission and neither tests H1 to H4. A1 to A3 must hold, or the token of Equation (9) records its fields instead of binding them. Every severance contract must be true and not only declared. And nothing here defends against a compromised mediation layer: a party holding the log key k can fabricate the observation prefix that every later decision reads. Reversibility alone says nothing about disclosure. R2 additionally requires empty sinks and no persistent side effects. Semantic soundness here means that every irreversible effect is labelled irreversible and every outward effect is either labelled irreversible or declares a nonempty sink or persistence. Under that premise R2 releases no outward effect, but it still supplies no path certificate and can bypass authorization or budget failure. For fixed trusted capabilities, schema, provenance, and event state, the checks are deterministic. An injection can change subsequent proposals and provenance, but cannot directly mint or widen authority from the trusted request. Residual goals can fall inside the compiled authority, use declared repairs or the R2 class, exploit unrepresented channels, or violate these semantic premises.

D Proofs

The runtime proofs use five implementation conditions, H1 to H5, with the labeling condition H3 split by label kind. In a real causal chain, a value link means the later value semantically depends on the earlier one, and the final link means the effect's occurrence or content depends on the last value; both are defined without reference to the graph. H1: the chain is emitted and simulated in the sense of Section 4.4, so every covered link emits its directed segment before dispatch, the segments concatenate, and taking a link entails traversing its segment. H2: every relevant unsafe or state-changing effect passes the mediator. H3a: protected-source labels are complete for the chain under consideration; H3b: untrusted-origin labels are; and H3c: the unsafe-effect rule labels the terminal of the pending call, so $U_t \neq \emptyset$. H4: the severance contract of every installed manifest entry is true under the complete manifest, for the adapter content warrant and the registry occurrence warrant alike. H5: the decision is current at dispatch, so the proposal, the execution version, and the policy epoch are rechecked, the only dispatched call is the canonical rewrite of the frozen proposal, and each resource snapshot or version is held through the certified access. Definition 2 is H4 together with the single-dispatch clause of H5. The prototype checks freshness process-locally; restored and repaired actions must separately satisfy the final-action premises to instantiate H4 and H5. Assumptions A1 to A9 of Section C.3 discharge it in a distributed deployment, and Propositions D.3 and D.4 give the argument.

D.1 Admission-Time Results

The gate model. A gate g maps a judgment $\mathcal{J}_i(K,\omega)$ to $\{\mathsf{mediate}, \mathsf{relax}\}$ and induces a mediator Γ_g that applies its *conservative transformation* at every site it mediates and the identity elsewhere. That transformation is assumed δ-achieving, so Equation (2) holds at every mediated site for every artifact in the gate's declared domain; refusing the call is the canonical instance, since it makes O_ω constant. Both sides of Equation (2) use the deployed measure. We write Γ_0 for a reference profile that mediates every site but ω, and Γ_0^{r} for its extension that relaxes ω. The variant class $\mathcal{V}(K,\omega)$ collects the artifacts in the domain sharing code, interface, and egress site while carrying different attacker-written specifications.

The three routes. $\mathcal{J}_{\mathrm{I}}$ holds facts discharged by verification conditions over the code. It proves that the mail implementation never reads credentials and says nothing about how *body* was chosen. $\mathcal{J}_{\mathrm{II}}$ is a sound abstraction, either of the artifact against its interface or of agent behavior reconstructed from traces; the code-interface form inherits the same blind spot, while an abstraction of full agent behavior can succeed, but only by separating the benign completion from the malicious one. $\mathcal{J}_{\mathrm{III}}$ reads dependencies off the specification and fails whenever a semantics-changing rewrite leaves the extractor's output unchanged. Publisher attestation replaces such reasoning with identity and stays out of scope.

The two conditions. Write $\Delta_\Gamma(K,\bar{c},\omega)$ for the inner supremum of Equation (2), the secret-induced gap at ω under context $\bar{c}$. C1: route i assigns one judgment to a pair $K_0, K_1 \in \mathcal{V}(K,\omega)$ in the gate's domain with $\sup_{\bar{c}} \Delta_{\Gamma_0^{\mathsf{r}}}(K_0,\bar{c},\omega) = 0$, so K_0 is safe to relax. C2: the attacker can pick $\bar{c}^\star$ with $\Delta_{\Gamma_0^{\mathsf{r}}}(K_1,\bar{c}^\star,\omega) > \delta$, so K_1 is not. They are what a route and a threat model must supply before Theorem 1 says anything.

Lemma D.1 (The site criterion is necessary)**.** *Work in the setting of Section C.1, write $\mathcal{B}^* = \bigsqcup_{n\ge 0}\mathcal{B}^n$, and let $\mathrm{pr}_\omega : \mathcal{E} \to \mathcal{B}^* \sqcup \{\bot\}$ send a finite egress sequence to the subsequence of its entries tagged ω, or to $\bot$ when there is none, so that $O_\omega = \mathrm{pr}_\omega \circ \mathsf{egress}$. Then pr_ω is measurable, and for every mediator Γ, artifact K, public context $\bar{c}$, and pair s_0, s_1,*

$$d\big((O_\omega)_{\#}\mu^{\Gamma}_{K,\bar{c}[s\leftarrow s_0]},\ (O_\omega)_{\#}\mu^{\Gamma}_{K,\bar{c}[s\leftarrow s_1]}\big) \ \le\ d\big(\mathsf{egress}_{\#}\mu^{\Gamma}_{K,\bar{c}[s\leftarrow s_0]},\ \mathsf{egress}_{\#}\mu^{\Gamma}_{K,\bar{c}[s\leftarrow s_1]}\big).$$

Consequently $\Phi_{\mathrm{NI}}(K,\Gamma;\delta)$ implies $\Phi_{\mathrm{site}}(K,\Gamma,\omega;\delta)$ at every egress site.

Proof. On each summand $(\Omega\times\mathcal{B})^n$ of $\mathcal{E}$ the map pr_ω is a finite composition of coordinate projections with selection on the value of the discrete tag, hence measurable; a map on a countable disjoint union that is measurable on each summand is measurable, so pr_ω and therefore O_ω are measurable. Total variation does not increase under a measurable map: for any measurable f and any P, Q, $d(f_{\#}P, f_{\#}Q) = \sup_A |P(f^{-1}A) - Q(f^{-1}A)| \le \sup_B |P(B) - Q(B)| = d(P,Q)$, since the preimages $f^{-1}A$ form a sub-collection of the measurable sets. Apply this with $f = \mathrm{pr}_\omega$ to the two egress pushforwards. Taking the supremum over s_0, s_1 bounds $\Delta_\Gamma(K,\bar{c},\omega)$, and taking the supremum over $\bar{c}$ gives the second claim. The converse fails and we never use it. □

Proof of Theorem 1. Let $K_0, K_1 \in \mathcal{V}(K,\omega)$ witness C1 and let g be as in the statement. Both variants lie in g's declared input domain and carry the same judgment, and g has the form $g(\mathcal{J}_i(\cdot,\omega))$, so it decides them alike. Suppose that decision is relax. Because Γ_g agrees with Γ_0 off ω at every artifact in the domain, in particular at K_1, we have $\Gamma_g = \Gamma_0^{\mathsf{r}}$ there, and C2 supplies a public context $\bar{c}^\star$ with $\Delta_{\Gamma_g}(K_1,\bar{c}^\star,\omega) > \delta$. Hence $\Phi_{\mathrm{site}}(K_1,\Gamma_g,\omega;\delta)$ fails at an artifact inside the gate's domain, contradicting δ-soundness. Every such gate therefore mediates both variants, and by Lemma D.1 the same failure forecloses Φ_{NI} at level δ. The statement has content only because the class of δ-sound gates is nonempty: the gate that mediates everywhere is δ-sound by the δ-achieving assumption, not vacuously.

It remains to say when C1 holds. For Route I and a code-interface Route II abstraction the judgment is a function of the shared code and interface alone, so it is constant on $\mathcal{V}(K,\omega)$. C1 then reduces to the existence inside $\mathcal{V}(K,\omega)$ of a safe member K_0 and, with C2, of a leaking member K_1. That is a δ-separation condition on the induced families $\mathsf{Beh}(K_0)$ and $\mathsf{Beh}(K_1)$, hence a property of the frozen policy M and the context space rather than of the code. It fails whenever the hypothesis of Proposition D.1 holds, and it also fails when no safe member exists at all, so the two results do not exhaust the space between them. The argument does not cover a sound abstraction of full

agent behavior that separates the pair. For Route III, let the extractor factor through a feature map ϕ. If a semantics-changing rewrite preserves ϕ, preserves the code and interface, and carries one specification to $\sup_{\bar{c}} \Delta_{\Gamma_0'}(\cdot, \bar{c}, \omega) = 0$ while the other satisfies C2, then the two carry the same extracted judgment and C1 holds. This is a sufficient condition for a blind spot, not a claim about every text extractor. Identity whitelists backed by trusted attestation, and independently certified invariants, fall outside the route judgment considered here. □

Proposition D.1 (The code-resolved positive case)**.** *Fix ω and the deployed mediator Γ with that site relaxed and its policy at other sites held fixed. Let $v(\tau)$ collect the slice inputs supplied to ω along τ, a measurable function of the trajectory. Write $\hat{O}_\omega$ for the value the code emits at ω before the mediator acts on it. Suppose that for every $L \in \mathcal{V}(K, \omega)$ there is a measurable F with $\hat{O}_\omega(\tau) = F\big(v(\tau), s\big)$ for $\mu^{\Gamma}_{L,\bar{c}}$-almost every τ and every $\bar{c}$, so that v also determines how often ω is reached. Suppose a sound verification condition establishes $F(v, s_0) = F(v, s_1)$ for all v, s_0, s_1, including the site's occurrence, termination, and exposed metadata, and write F_0 for the common value, a measurable function of v alone. Suppose Γ acts at ω as a measurable function of the ω-call alone, and suppose finally that under the deployed measure the law of v is independent of the secret, $v_{\#}\mu^{\Gamma}_{L,\bar{c}[s\leftarrow s_0]} = v_{\#}\mu^{\Gamma}_{L,\bar{c}[s\leftarrow s_1]}$ for every $\bar{c}$ and s_0, s_1. Then $\Delta_\Gamma(L, \bar{c}, \omega) = 0 \le \delta$, so a Route I gate may relax the class without violating Φ_{site}.*

Proof. Fix L, $\bar{c}$, and s_0, s_1. The verification condition gives $\hat{O}_\omega = F_0(v)$ almost surely, and F_0 does not read s. The mediator's action at ω is a fixed measurable map med of the call, applied at each occurrence, so $O_\omega = G(v)$ for the fixed measurable $G = \mathrm{med} \circ F_0$. The last hypothesis makes the law of v the same under s_0 and s_1. Equal laws pushed through one measurable map are equal, so $(O_\omega)_{\#}\mu^{\Gamma}_{L,\bar{c}[s\leftarrow s_0]} = G_{\#}v_{\#}\mu^{\Gamma}_{L,\bar{c}[s\leftarrow s_0]} = G_{\#}v_{\#}\mu^{\Gamma}_{L,\bar{c}[s\leftarrow s_1]} = (O_\omega)_{\#}\mu^{\Gamma}_{L,\bar{c}[s\leftarrow s_1]}$ and $\Delta_\Gamma = 0$. The conclusion concerns this site-level criterion only, and Lemma D.1 does not upgrade it to Φ_{NI}. □

The last hypothesis is the load-bearing one and Route I evidence does not supply it. It says the model's choice of the slice inputs is itself secret-independent, which is noninterference of M. It holds for a deterministic library call whose arguments are fixed by the code, and it fails for the mail skill of Section 3, where M builds *body* from the context. Theorem 1 and Proposition D.1 therefore bracket the space from opposite sides without meeting in the middle.

D.2 Certified Separation

Proof of Proposition 1. Clause (a) uses no semantic premise about executions. Acceptance is atomic with respect to the record append, so the freshness tests, the condition checks, and the store advance share one linearization point.

Conditions 1 to 3 of Section C.4 fix the objects. Live mode with $\mathrm{res} = \perp$ lets the same operation reserve q_t and append `DECIDE(accept)`, which is the theorem's protocol clause. The certificate names the current execution-state version σ_t and the immutable `PROPOSE` record of q_t with no decision record, so $\widehat{G}_t^-$ is exactly the active graph named, and the current epoch fixes $\widehat{\mathsf{Sev}}$, $\widehat{\mathsf{Impl}}$, $\widehat{T_{\text{bad}}}$ and the schema. Binding to the proposal, and not to a decision field, breaks the circularity.

Conditions 4 to 7 rebuild $\widehat{B}_t$, $\widehat{P}_t$, and $\widehat{U}_t$ from that fixed material, recompute every binding and capacity, and check that removing $\mathcal{C}_t$ separates $\widehat{U}_t$ from $\widehat{B}_t \cup \widehat{P}_t$ at the declared cost. Conditions 8 to 10 tie the cut to enforcement, giving $\mathrm{dom}(\mathcal{M}_t) = \mathcal{C}_t$ with one entry per edge, no $\perp$ binding, canonical hook metadata, and declared joint admissibility. Minimality is the one property the verifier takes on declaration rather than recomputation, which is why Proposition 1 states the declared cost and not an optimality claim. For clause (b), acceptance reruns all nine obligations on each effect atom and its capability witness. Their coverage, validator, declassifier, and hook premises then establish authority for the concrete effects of the checked call. □

D.3 The Cut: Closed Form

The network is the flow subgraph of G_t^- with a super-source joined to $B_t \cup P_t$ and a super-sink joined from U_t, both at infinite capacity, so max-flow/min-cut applies directly (L. R. Ford & Fulkerson,

1956) and the connectors never appear in a minimum cut. Each original edge appears once and is charged once. The emission rules make the resulting optimum explicit.

Proposition D.2 (Closed form of the certified cut)**.** *Fix a step with $U_t = \{T(q_t)\}$ and write $\kappa(C) = \sum_{e \in C} \kappa(e)$. Let A be the argument edges of a_t and e_{T} its terminal edge $\mathrm{op}(q_t) \to T(q_t)$. Let R be the set of nodes reachable from $B_t \cup P_t$ in the committed flow subgraph,* including $B_t \cup P_t$ itself*, and let $A^\star = \{e \in A : \mathsf{src}(e) \in R\}$. If $A^\star = \emptyset$ then $\mathcal{C}_t = \emptyset$. Otherwise, provided at least one of e_{T} and $A^\star$ is bindable throughout,*

$$\kappa(\mathcal{C}_t) = \min\{\, \kappa(\{e_{\mathrm{T}}\}),\ \kappa(A^\star)\,\}, \tag{10}$$

and $\mathcal{C}_t$ is the lexicographically least minimum-cost separating set under the canonical edge order. Placing e_{T} before every argument edge in that order makes $\mathcal{C}_t$ equal to $\{e_{\mathrm{T}}\}$ on a tie. One reachability pass over the committed flow subgraph therefore determines $\mathcal{C}_t$ in time linear in the size of G_t^-, given B_t and the bindings.

Proof. By Equation (6) the only finite-capacity edges are the proposed flow instances of a_t with a nonempty binding, and by (E1) and (E4) of Section C.2 those lie in $A \cup \{e_{\mathrm{T}}\}$; rules (E2), (E3), and (E5) fire at finalization and emit committed edges only. The node $\mathrm{op}(q_t)$ is fresh, its only out-edge before dispatch is e_{T}, and $T(q_t)$ has no other in-edge. Every directed path from $B_t \cup P_t$ to $T(q_t)$ therefore ends $\mathsf{src}(e) \to \mathrm{op}(q_t) \to T(q_t)$ for some $e \in A$ with $\mathsf{src}(e) \in R$, that is for some $e \in A^\star$; the reflexive case, where $\mathsf{src}(e)$ is itself protected or untrusted, is the path of length two. Conversely each $e \in A^\star$ carries such a path. If $A^\star = \emptyset$ no path exists and $\emptyset$ separates. Otherwise let C be any separating set of finite-capacity edges. For each $e \in A^\star$, cutting the path through e requires $e \in C$ or $e_{\mathrm{T}} \in C$, so either $e_{\mathrm{T}} \in C$ or $A^\star \subseteq C$, and by strict positivity of κ we get $\kappa(C) \geq \min\{\kappa(\{e_{\mathrm{T}}\}), \kappa(A^\star)\}$ with equality only for $C = \{e_{\mathrm{T}}\}$ or $C = A^\star$. Both of these separate, so they are the only minimizers and Equation (10) follows. If some edge of $A^\star$ is unbindable then $\kappa(A^\star) = \infty$ and the minimum is attained at $\{e_{\mathrm{T}}\}$ whenever e_{T} is bindable; if e_{T} is unbindable too, the minimum is infinite, the sentinel appears in every separating set, and R5 refuses. □

Only the current call's edges are severable, so the optimization reduces to a two-way choice between blocking the call and paying for its live argument edges, and Lemma C.1 shows the first branch always separates when the adapter offers `block_call`. Optimality therefore does not imply usefulness: the cut earns its place only where the argument branch is both cheaper and available, which the occurrence warrant makes rare for model-chosen calls. The preceding proof also gives the final-action criterion in Section 4.4: for any $S \subseteq A \cup \{e_{\mathrm{T}}\}$, every unsafe path is hit iff $e_{\mathrm{T}} \in S$ or $A^\star \subseteq S$. A restoration executes with $S = \emptyset$; when $A^\star \neq \emptyset$, at least one path survives regardless of C's verdict. We keep the max-flow formulation because it survives emission rules that add intermediate proposed edges, for which the two-candidate argument no longer applies.

D.4 Coverage and Simulation on the Transparent Fragment

Proof of Proposition 2. Write w_i for the call that realizes the link $x_i \to x_{i+1}$ for $i < m-1$, q_t for the call that realizes the final link $x_{m-1} \to \zeta_t$, and ν for the host's provenance map, and set $\alpha_{\mathrm{D}} = \nu$ on the values of γ and $\alpha_{\mathrm{T}}(\zeta_t) = T(q_t)$. The argument is an induction over the host trace using only (E1) to (E5) of Section C.2, which the mediation layer executes and the artifact does not.

Base and the direct case. x_0 has a provenance identifier by hypothesis. If $m = 1$ it enters q_t at a declared position, so E1 and E4 emit the final segment directly and the value-link induction is empty. If $m > 1$, transparency puts x_0 at a declared argument position j of w_0, coverage gives $\mathrm{schema}(\mathrm{op}(w_0), j) \neq \bot$, and E1 emits $\alpha_{\mathrm{D}}(x_0) \to \mathrm{op}(w_0)$ before w_0 leaves the host.

Value links. Fix $i \geq 0$ and suppose every earlier segment is emitted. Transparency puts x_{i+1} either at a declared return position of w_i or in a resource w_i declares that it writes. In the first case (E2) mints its identifier and emits $\mathrm{op}(w_i) \to \alpha_{\mathrm{D}}(x_{i+1})$ at the outcome hook, giving the two-edge segment $\pi_i = \alpha_{\mathrm{D}}(x_i) \to \mathrm{op}(w_i) \to \alpha_{\mathrm{D}}(x_{i+1})$; this is the shortest form available, since the construction has no `Data`-to-`Data` edge. In the second case the link is realized by the write w_i together with the read w_i' that follows it. Rule (E5) mints the resource version R_{v+1} and commits $\mathrm{op}(w_i) \to R_{v+1}$ with the union of that call's committed input labels, and w_i' contributes $R_{v+1} \to \mathrm{op}(w_i')$ by (E1) and $\mathrm{op}(w_i') \to \alpha_{\mathrm{D}}(x_{i+1})$ by (E2), so π_i has four edges and passes through R_{v+1}. The proposition's finalization hypothesis names both, w_i as the call that produced the resource version and w_i' as the

call that produced the value. Either way, identifier preservation and (E3) carry the identifier through any pass-through tool, and the transport hypothesis puts the value at the declared argument position of the next call, so the in-edge (E1) emits starts where the out-edge of w_i ended. A weaker notion of identity fails here: byte equality would merge unrelated values, and object identity would not survive serialization at the tool boundary.

The final effect link. Rules (E1) to (E3) cannot supply this step, since they emit only `Data–Operation–Data` segments and never reach a `Terminal` node. By transparency x_{m-1} enters q_t at a declared position, so (E1) emits $\alpha_{\mathrm{D}}(x_{m-1}) \to \mathrm{op}(q_t)$, and by (E4) the proposal extension creates $T(q_t)$ fresh and emits $\mathrm{op}(q_t) \to T(q_t)$, both at `before_tool_call`. Their composition $\pi_{m-1} = \alpha_{\mathrm{D}}(x_{m-1}) \to \mathrm{op}(q_t) \to T(q_t)$ reaches the effect and not only the last value. Without (E4) the proposition would establish value-to-value provenance and say nothing about confinement.

Concatenation, timeliness, and entailment. Consecutive segments share the endpoint $\alpha_{\mathrm{D}}(x_{i+1})$, so they concatenate into one flow path from $\alpha_{\mathrm{D}}(x_0)$ to $T(q_t)$. The calls $w_0, \ldots, w_{m-2}$ and every reading call w_i' appended `OUTCOME` before q_t was proposed, so their proposal, output, and resource-version edges are committed and appear in G_t^-, whether those calls were certified or permissively released. The two edges of π_{m-1} are `proposed` instances of q_t present before dispatch. Taking a direct value link requires its declared input and output, which fire (E1) and (E2). Taking a resource-mediated link requires the declared write and the declared read, which fire (E1) and (E5) for w_i and (E1) and (E2) for w_i', that is all four edges of π_i. The no-secondary-call condition excludes any other realization of a link, at value links and at the final link alike. Rule (E4) fires before dispatch and the schema assigns traversal of the terminal edge to crossing that mediated effect boundary. Taking a link therefore entails its segment, and γ is simulated. □

Remark D.1 (Reach of the transparent fragment). Hidden input and hidden effect escape the fragment; internal computation does not. The fragment is narrow because identifiers do not survive the model, so Section 5 reports its coverage over tools, calls, and whole chains. Coverage is stated apart from H1 to keep the pair from being circular: coverage is structural and audited offline, while emission and simulation are runtime claims inside H1. A covered link whose emitter never fires is therefore a violated premise, not a silently uncovered one. Source and terminal labeling, H3, is measured on its own.

D.5 Freshness, Authentication, and Anti-Replay

Proposition D.3 (One-shot authorization and one-shot consumption)**.** *Assume A1 to A8. Every accepted certificate advances the verifier store by exactly one. A certificate naming a stale committed state or an old sequence number is rejected, and one accepted once cannot be accepted again. A token issued once is admitted by the executor at most once. And after a crash, the absence of* `DISPATCH` *implies no external send.*

Proof. By (A5) acceptance is one linearization point: the reservation test, the freshness tests, the update $h^{\mathrm{auth}} \leftarrow \mathsf{h}(h^{\mathrm{auth}} \| \mathsf{cert}_t)$, `DECIDE(accept)`, and token issue are inseparable, so no record or crash leaves an accepted-but-undecided reservation, and two submissions of the same n_t cannot both succeed. Condition 2 of Section C.4 rejects stale authorization. A certificate with $n_t \neq n^* + 1$ is rejected, so the sequence is a strict successor relation, not an equality test against a current digest, which would reject every new certificate while admitting the most recent one. A certificate whose σ_t is not the current execution-state version is rejected as well; by (A6) the raw log tip is not tested, since the step's own records move it. Replay of an accepted certificate is caught by condition 2, which finds a decision already recorded for that attempt, and by the sequence test; a `BLOCKED` or `CANCEL` attempt does not advance σ_t, so the state test alone would not reject it. By (A4) the certificate cannot be re-presented as a fresh attempt without a different q_t and hence a different token preimage.

Verifier acceptance alone would not bound execution, since an issued token could be replayed while the first call is still running. The executor therefore admits tok_t by a compare-and-swap on its own store keyed by q_t, which by (A4) to (A6) admits it at most once and survives restart. This is also why q_t is never reused: a retry under the same identifier would collide with a consumed entry.

Binding ties the token to the manifest. By (A1) no party without k' produces a valid tok_t; by (A3) the preimage $q_t \| n_t \| \sigma_t \| \varepsilon_t \| \mathsf{h}(L_t^{\mathrm{p}}) \| \mathsf{h}(\mathcal{M}_t)$ parses uniquely, so a producer cannot shift bytes between fields; and by (A2) a producer cannot show one manifest to the verifier and a different manifest with

the same digest to the executor. Without (A2) or (A3) the executor could install a manifest the verifier never checked while the token still verifies.

The crash claim needs (A7) and (A8). A finalized policy block has a durable `BLOCKED` terminal and no send, and a failed preparation sends nothing and finalizes `CANCEL`. Otherwise, were `DISPATCH` appended only in memory, a request could leave the host while the crash erased the record, and recovery would then classify a completed irreversible effect as never dispatched and permit an automatic retry. Write-ahead durability with no reordering rules that out, so absence of `DISPATCH` implies no send and permits `CANCEL`. □

Proposition D.4 (The executed manifest is certified against the executed state)**.** *Under A1 to A6, A8, and A9, no other mediated call is reserved, dispatched, or committed between the moment* AcceptIfFresh *reserves q_t and the moment* Finalize *releases it. At dispatch the immutable proposal, resource snapshots or leases, execution-state version, policy epoch, and installed manifest are exactly those the certificate names, and the dispatched call is exactly* $\mathrm{Apply}_{\varepsilon_t}(\mathcal{M}_t, L_t^{\mathrm{p}})$.

Proof. By Proposition 1 the token commits to σ_t, ε_t, $\mathsf{h}(L_t^{\mathrm{p}})$, and $\mathsf{h}(\mathcal{M}_t)$. A5 and A6 make the global reservation linearizable and rollback-protected, so no other mediated call can advance σ_t before finalization, and by A9 no unmediated event can change the committed evidence without advancing σ_t either. By A8, Prepare spends the token only after rechecking q_t, σ_t, ε_t, and the two digests against the immutable `PROPOSE` record, and its `READY` branch dispatches nothing but $\mathrm{Apply}_{\varepsilon_t}(\mathcal{M}_t, L_t^{\mathrm{p}})$ while holding every resource lease through the certified access. Its LeaseIfCurrent(ε_t) atomically checks the epoch and acquires a read lease; a changed epoch finalizes `CANCEL` without dispatching, and after success a writer must wait. By collision resistance of h the installed manifest is the one whose digest the token carries, and preparation returns `READY` only once all entries are installed. Clauses three and four of H5 are therefore consequences of the protocol rather than assumptions, which is why Theorem 2 may cite this proposition. Dropping A9 leaves the statement true of the store and false of the world. □

Remark D.2 (Why the reservation is required). The epoch freezes configuration, not execution state, so without a global reservation this interleaving is admitted. Call q_1 reads a shared file and sends it to the network, and at certification the file is benign, so the cut is empty or cheap. Before q_1 dispatches, q_2 writes a protected value into that file and commits. No configuration changed, so ε did not move and the old token still verifies against the epoch, the certified cut never saw the new path, and q_1 sends the secret. Two properties close it: the token carries σ_t, which q_2's `OUTCOME` advances, so the executor's recheck turns the interleaving into a refusal, and the reservation prevents it from arising at all, covering permissive releases too. Separately, Proposition D.3 bounds how often a *token* is spent, not how often an external effect happens, which is why `DISPATCH` is committed before the call is invoked and why recovery retries only in the provably undispatched case.

D.6 Grounded Confinement and the Episode Invariant

Proof of Theorem 2. Let $\gamma = x_0 \to \cdots \to x_{m-1} \to \zeta_t$ be a schema-covered real causal chain from a registered protected source to the pending unsafe effect, and suppose the verifier accepted its certificate but the chain completes. For $i < m-1$ let w_i realize $x_i \to x_{i+1}$ with simulated segment π_i, let π_{m-1} be the final segment for q_t, and let $\alpha_{\mathrm{D}}, \alpha_{\mathrm{T}}$ be the simulation maps with $\alpha_{\mathrm{T}}(\zeta_t) = T(q_t)$. By H3a, registration gives $\alpha_{\mathrm{D}}(x_0) \in P_t$. By H3c, the unsafe-effect rule gives $T(q_t) \in U_t$. By H1 every link has a nonempty flow segment, the segments concatenate, and taking a real link entails traversing every edge of its segment. Write $\Pi = \pi_0 \cdots \pi_{m-1}$, a flow path from P_t to $T(q_t)$ in G_t^-. Concatenation is what makes Π a witness of reachability, since per-link emission alone would leave segments that never meet at shared endpoints. Because G_t^- is folded in phase 1 from the committed state and the proposal only, this witness exists before the call runs and uses no outcome of step t.

By Proposition 1, $T(q_t)$ is unreachable from P_t after removing $\mathcal{C}_t$, so some edge e of Π lies in $\mathcal{C}_t$. That edge lies on at least one segment; choose a corresponding real link, which by entailment cannot be taken unless e is traversed. The same theorem gives $\mathrm{dom}(\mathcal{M}_t) = \mathcal{C}_t$ with one entry per cut edge, so e has an entry, and that entry is the canonical minimum-cost member of $\mathsf{Sev}(e) \cap \mathsf{Impl}(e)$ on a `proposed` instance with the registry's argument and hook. By H4 the executor installs the entry and its severance contract holds under the complete manifest, and by the definition of Sev in Section 4.2 that contract speaks about real execution, not about the graph. By the theorem's freshness condition

the manifest, state, and epoch at dispatch are the certified ones, which Proposition D.4 implements under its protocol assumptions. The installed entry is therefore the entry checked, the link cannot be taken, and γ does not complete, contradicting the supposition. The scope is exactly coverage: a chain with an uncovered link has no declared segment, witnesses no reachability, and is untouched. □

Corollary D.1 (Grounded integrity)**.** *Replace H3a by H3b and let γ start at a live value the deployment marks untrusted. Under the same conditions γ cannot complete at this call.*

Proof. The proof above uses $\alpha_{\mathrm{D}}(x_0) \in P_t$ only to place the path's source in the cut's source set. H3b places it in B_t instead, and Proposition 1 separates U_t from $B_t \cup P_t$, so the rest of the argument is unchanged. The integrity statement is weaker in practice because the occurrence warrant is unavailable for model-chosen calls, so the cut that discharges it is almost always $\{e_{\mathrm{T}}\}$. □

Corollary D.2 (Monitor invariant over an episode)**.** *Let the certified steps of an episode be exactly $t_1 < \cdots < t_N$, with H1, H3a, H3c, H4, and H5 at each of them and A1 to A9 throughout. Then Theorem 2 holds at every t_i. Suppose in addition that the episode runs under* **PACE**-C *and that H2 holds for the class of effects at which a covered chain could complete, so the deployment registers every one of them as a mediated unsafe boundary. Then no covered chain from a registered protected source completes at any certified step.*

Proof. The first part applies Theorem 2 at each t_i. For the second, H2 routes every effect of the registered class through the mediator, so such an effect executes only under an accepted decision. A decision is accepted only inside an accepting AcceptIfFresh, and by Proposition D.3 a certificate is accepted at most once and its authorization admitted at most once. Under **PACE**-C there is no uncertified executing branch: R1, R4, R5, and R6 refuse, and the remaining branch certifies. A certified step with an empty cut installs nothing. But H3c makes $U_t \neq \emptyset$ wherever a covered chain could complete, so Proposition D.2 applies and an empty cut means $A^\star = \emptyset$. A covered chain would put $\alpha_{\mathrm{D}}(x_0) \in P_t \subseteq R$, carry its prefix over committed edges so that $\alpha_{\mathrm{D}}(x_{m-1}) \in R$, and place its last value at a declared argument position of q_t, which forces that argument edge into $A^\star$. So no covered chain completes at an empty-cut step either. Persistence of previously installed entries carries nothing, since after the committed state changes an old cut may be bypassed by a new edge. Execution is prevented by token discipline instead, which makes this a reference-monitor invariant and not a further confinement argument. H2 is a completeness assumption on the deployment's effect registry, and without it the statement says only that nothing leaks through boundaries the deployment labelled correctly.

The corollary is stated for **PACE**-C because **PACE**-P has executing branches without a grounded path certificate. Under the semantically sound labels of Section C.6, an outward or irreversible effect cannot satisfy R2's three conjuncts. Outside R2, an unrepaired dispatch, including R3 and a restored P block, requires all nine effect obligations to pass; otherwise the only executing route is a registry-declared repair of a repairable verdict. This proves the payload-independent claim in Section 4.2: an out-of-authority outward or irreversible effect can execute only through such a repair, given sound request compilation, schema coverage and semantics, validators and declassifiers, complete mediation of the effects, and application of every required hook. Because repairs are not rechecked by R6, the argument does not certify their final effects. R2 and restoration likewise do not establish the episode confinement invariant. A leaking R2 call would require an unsound label or a missing sink/persistence declaration; reversibility alone is not its admission test. The per-call theorem applies to **PACE**-P only when its actual installed action satisfies all the theorem's premises. □

E Experimental Details

E.1 Frozen Hyperparameters

Every value below is fixed before the primary run and is not tuned per benchmark, model, or seed.

E.2 Benchmarks, Adapters, and Denominators

Each benchmark keeps its own success and utility functions. AgentDojo and AgentDyn use executable security and utility; InjecAgent reports direct-harm and data-stealing rates; ASB its five attack groups

Table 4: Frozen evaluation and policy configuration.

Field	Value
Target models	`gpt-5.6-luna`; `DeepSeek-V4-Flash`; local `Qwen-3.8-27B` served by vLLM
Decoding	Temperature 0; maximum completion length 1024 tokens
Ablation seed	`20260828`; single deterministic paired run
Concurrency	8 workers; throughput only, no effect on the statistical population
Defense lanes	`none`, `pace`, and the external baselines of Table 7
Layer switches	Path confinement, capability and effect verification, and execution-boundary adaptation, toggled independently for the ablation arms of Table 6
Cut solver	src–snk max-flow/min-cut over the flow subgraph of G_t^-
Effect obligations	The nine checks of Section 4.2, evaluated in a fixed order
Judge	Frozen prompt and aggregation; evidence required; no manual correction
Technical-failure ceiling	5% per reported cell

Table 5: Benchmarks, native evaluators, and the paired-subset size used in the ablation. Commit and data revisions are recorded in the release manifest of Section E.6.

Benchmark	Scaffold and native evaluator	Paired cases
AgentDojo	Official task, security, and utility functions	198
AgentDyn	Official three-domain runner and utility scorer	60
WASP	Official end-to-end evaluator; plain-text and URL groups	24
ASB	Official agents and metric scripts; seven run variants	280
PASB	Official personalized-agent workflow; IPI and four memory subsets	58
InjecAgent	Official tools and cases; Base and Enhanced splits	212
MCPTox	Official templates with the native refusal metric	135
MSB	Official stage scorer with PUA and NRP	200
Total		1167

with utility; WASP the plain-text and URL-injection groups with end-to-end user utility; PASB the IPI group and four memory subsets; MCPTox per-template attack success with a refusal rate; and MSB stage scores with PUA and NRP. Capabilities are compiled deterministically from the authenticated request, intersecting request-derived effect families, trusted argument constraints, organizational policy, and tool schemas. The compiler reads no injected artifact, attack label, or judge output. Policies and schemas were developed from public interfaces, task formats, and development tests, then frozen before the final runs and completion of the factorial arms.

For every adapter the harness stores an episode record containing the user task, the injected artifact, the target-model revision, the tool schema, every call to the defense hooks, tool results, selected actions, certificate-verifier decisions, native evaluator output, judge evidence, latency, token count, and terminal status. For a defended run, an episode lacking a required defense-hook call is invalid rather than a sample. The additional adaptive search uses the same AgentDojo version and configuration. An authority audit of all 629 (user, injection) pairs checks ground-truth target calls without dispatch: 375 (59.6%) fail AUTH, comprising 60/144 Banking, 39/105 Slack, 93/140 Travel, and 183/240 Workspace pairs. With seed 20260922, ten out-of-authority pairs are selected from each of Slack, Travel, and Workspace; Banking is excluded. The undefended agent and **PACE**-P are attacked separately on `Qwen/Qwen-3.8-27B-FP8` at temperature 0. Starting from the static injection, a mutator using that checkpoint at temperature 1.0 sees six candidates sampled from the eight best and four most recent, the agent's tool calls, guard decisions and first failed obligations where applicable, and the security verdict; it proposes four new payloads per round. Each pair stops at first success or 50 episodes. The protocol, sample, and code were fixed before an isolated five-pair pilot. Twenty of the 21 undefended successes, selected with a fixed seed, were reviewed manually.

Each reported rate carries four quantities: the numerator, the valid denominator, the invalid count, and the technical-failure count. Three conventions matter for reading the tables. InjecAgent DS-S2 is a conditional metric, so a cell with no eligible sample is reported as `N/A (0 eligible)` and never as a measured zero. MCPTox refusal and ASB refusal use the valid denominator of their own arm, which differs across arms whenever the invalid count differs. For MSB, group NRP is

$N_g = U_g(1 - A_g)$ on rates in $[0, 1]$. Full-table companion metrics pool their eligible samples, using $M_{\text{pool}} = \sum_g n_{g,M} M_g / \sum_g n_{g,M}$; undefined PUA/NRP groups contribute no samples. Thus a weighted mean of group NRPs need not equal the product of separately displayed PUA and ASR means. NRP is a composite score, not a counted joint safe-and-successful outcome. The official type-macro output instead averages ASR across valid attack types and PUA across types where it is defined, then computes $\text{PUA}_{\text{macro}}(1 - \text{ASR}_{\text{macro}})$. Full type coverage has 12 ASR and 8 PUA types; `false_error`, `simulated_user`, and their injection combinations lack native PUA. The ablation's four-stage macro is a separate summary and is not interchangeable with the full-table pooling.

E.3 Ablation Design

Table 6: Ablation arms. P is path confinement, C is capability and effect verification, B is execution-boundary adaptation. All eight arms run on the same paired subset, so component contrasts are paired; zero-ASR floors can prevent identifying a security benefit.

Arm	P	C	B	Role
A0				Wrapper-matched no-defense corner
A1	✓			Path confinement alone
A2		✓		Capability and effect verification alone
A3			✓	Execution-boundary adaptation alone
A4	✓	✓		Path confinement with verification
A5	✓		✓	Path confinement with boundary adaptation
A6		✓	✓	Verification with boundary adaptation
A7	✓	✓	✓	Complete **PACE**, paired anchor

Every arm keeps the runner, the transport, the case selector, and the evaluation chain of the complete system and changes only the layer switches, so a reduced arm is wrapper-matched to complete **PACE** rather than compared against a historical no-defense run. The complete-method column is re-aggregated offline from the run records of the full system on the frozen case identifiers, which issues no new model call and leaves the main tables untouched.

Two data-handling notes belong here rather than in the main text. Ten MSB cases had failed for a dependency reason in an earlier run and were re-executed under the same configuration and merged back by case identifier, which brings both affected arms to 200 scored cases with no technical failure. On ASB the paired selection is made by matching a stable trajectory identity, namely the agent name, the attack tool, and the initial user message; an earlier aggregation that took the first forty rows of each run file has been withdrawn, and the audited selection resolves all seven run variants one to one with no missing or ambiguous match. The completed factorial contains all 64 benchmark–arm cells. C toggles capability installation, effect expansion, verification, and the combiner together, so its contrasts concern that bundle. Metric-specific eligibility is applied after matching logical case IDs. ASB and InjecAgent-Enhanced have zero ASR already at A0 and identify no security improvement.

E.4 Statistical Analysis

The main and per-group tables report point estimates on native denominators; they do not display confidence intervals. For the adaptive case-success result we report x/n and a two-sided 95% Wilson interval. For correlated actions we resample complete episodes, and where several attacks share a task we resample the task cluster. Paired comparisons use identical case identifiers, target revisions, and decoding settings, with an exact McNemar test and Holm correction specified for inferential comparisons inside each family; no such significance claim is made from the displayed point estimates. The ablation is one deterministic paired run under a single seed, so we report differences on that population and make no multi-seed stability claim. The displayed arm rates remain descriptive comparisons; we make no claim of a statistically significant component effect. We do not pool attack success across benchmarks, since the native success functions are not commensurable, and we do not describe any difference as statistically significant without the corresponding test.

Technical failure includes endpoint timeout after the frozen retry policy, malformed official evaluator state, missing hook invocation, unrecoverable tool sandbox failure, or an absent native denominator.

It is reported as its own rate. A cell above the 5% ceiling is removed from the primary table and retained, with failure causes, in Section F.

The frozen judge returns an unsafe or safe label, the decisive trajectory span, the affected resource, and the native-metric mapping. The manifest records official evaluator outputs and judge evidence; the current tables do not provide a numerical judge-agreement or parse-failure analysis. Judge labels are never edited by hand.

E.5 Baseline Port Fidelity

Table 7: External baselines by family, with the benchmarks each is reported on and the fidelity of the port.

Family	Methods	Benchmarks and fidelity
No defense	Baseline, ReAct	All eight; native
Trusted flow and IFC	CaMeL (Debenedetti et al., 2025), FIDES (Costa et al., 2025), DRIFT (Li et al., 2025a)	AgentDojo, AgentDyn, WASP, PASB, MSB; native on AgentDojo, validated port elsewhere
Tool-use safeguards	Progent (Shi et al., 2025), Tool Filter (Debenedetti et al., 2024), Tool Allowlist (Wang et al., 2026c), MCIP Guardian (Jing et al., 2025), ToolShield (Li et al., 2026b)	AgentDojo, ASB, AgentDyn, MCPTox, MSB, PASB; native or validated port
Re-execution	MELON (Zhu et al., 2025)	AgentDojo; native
Detection	DataSentinel (Liu et al., 2025), PI-Detector (Debenedetti et al., 2024), PIGuard (Li et al., 2025b), PromptGuard-2 (Llama Team, 2025), Metadata Sanitization (Wang et al., 2026c)	AgentDojo, InjecAgent, ASB, AgentDyn, MCPTox, PASB, MSB; pre-filter plus end-to-end
Prompt-level	Delimiters, Sandwich, Spotlighting, Instructional Prevention, Direct and PoT Paraphrase, PoT Shuffle, Repeat User Prompt	WASP, ASB, InjecAgent, AgentDyn, PASB; native
Diagnostic reference	DTA (Enck et al., 2010; Kemerlis et al., 2012)	AgentDojo; faithful instrumentation

DTA propagates labels through instrumented hashes and persistent sessions and serves as an instrumented diagnostic reference, excluded from main-table selection. DataSentinel is evaluated end to end as a pre-filter. All reported comparisons are local executions with pinned method and benchmark revisions; manifests retain model identities, configurations, intervention positions, evaluators, and output provenance. Ports adapt APIs, tool schemas, and endpoints while preserving the method's intervention. Within a benchmark and target model, methods share the case universe, attack collection, and native evaluation chain. On AgentDojo, the undefended agent, **PACE**, CaMeL, DTA, Progent, MELON, DataSentinel, Spotlighting, and PI-Detector each completed 12/12 jobs across three attacks and four suites. Unsupported method–task combinations remain outside their subgroup averages; execution completion alone is not a valid denominator for every native metric.

E.6 Reproducibility Manifest and Replay

The artifact contains a machine-readable manifest with the following fields:

```
paper_protocol_revision
repository_commit
container_digest
benchmark_name, benchmark_commit, data_revision
agent_scaffold_revision, adapter_revision
target_provider, target_model, target_revision
attack_family, generator_model, generator_prompt_hash
judge_model, judge_prompt_hash, judge_temperature
```

```
seed, defense_lane, policy_config_hash
episode_id, graph_version
native_metric_numerators, native_metric_denominators
technical_failure_code, latency_ms, token_counts
```

The manifest is the authoritative record of every revision referenced in Section E.2. For each episode, replay reconstructs the initial graph, consumes the stored sequence of proposals and tool events, and checks every graph version, cut, binding, policy action, certificate-verifier decision, and executor receipt. A replay is successful only if all deterministic fields match. Model generations are stored as inputs to this replay rather than re-sampled. A separate live replay mode reissues model calls and is reported as a non-deterministic replication.

E.7 Cost, Release Checklist, and Artifact Safety

The harness records target-agent tokens, judge tokens, wall-clock latency, flow-solver time, peak graph size, certificate-check time, and tool sandbox CPU and memory. A numerical mediation-overhead summary is not reported in this manuscript. The solver is deterministic for a fixed graph, so model-call variance is reported separately. Hardware, vLLM revision, quantization status, tensor parallelism, container digest, and the API price snapshot are frozen in the reproducibility manifest.

1. Freeze all eight benchmark and scaffold revisions.
2. Publish the complete policy configuration, the effect schema, and the capability compiler.
3. Publish the agent prompt and the frozen judge prompt.
4. Release per-episode native evaluator inputs and outputs, subject to benchmark licenses and removal of operational secrets.
5. Release verifier tests for stale state, stale prefix, non-separating cut, misstated cut cost, unbindable edge, and each of the nine effect obligations.
6. Release the notebook that regenerates every table and interval from the result manifest, including the frozen case selectors of the paired subset.

The public artifact contains no live credentials, production endpoints, or working exfiltration targets. Harmful payloads are represented by inert local sinks and deterministic fixtures that preserve the path structure the evaluation needs.

F Extra Experiments

F.1 Per-Group Benchmark Results

Table 1 report, for each benchmark and model, the worst eligible attack group with its companion metric. Each block lists the undefended agent, **PACE**, and the three comparators with lowest three-model mean worst-group ASR; ties use the companion metric. DTA is excluded as a diagnostic reference. For InjecAgent the summary uses Direct Harm and DS-S1 for every method, excluding conditional DS-S2 and the derived data-stealing aggregate. Across the eight tables there are 84 elementary ASR columns; removing the three DS-S2 columns and two with zero undefended ASR leaves 79. **PACE** is strictly lowest in 62, tied in 14, and higher in three: Qwen WASP plain text (4.76% versus FIDES 2.38%, with 92.86% versus 0.0% utility), Qwen MSB-Call (2.50% versus Tool Filter 0.0%, with 60.14% versus 47.34% PUA), and DeepSeek AgentDyn GitHub (0.06% versus CaMeL 0.0%, with 63.57% versus 3.75% utility; PI-Detector and DRIFT also attain 0.0% ASR). The tables retain every group and ported baseline on the same runs as the main summaries; metric-specific eligibility follows Section E.2.

F.2 Ablation Notes

The ASB utility row. Table 2 reports 0/280 ASB utility for six of the eight arms, with only P+C and complete **PACE** at 84/280. The exact-pair audit confirms that these zeros come from the official `Original Task Successful` column and are not an aggregator artifact that turned missing

Table 8: Results on WASP under the official end-to-end evaluator. Attack success is reported for the plain-text and URL-injection formats; utility is the native user-task accuracy. Lower is better for ASR, higher for utility; best ASR per block in bold.

	Qwen-3.8-27B			DeepSeek-V4-Flash			gpt-5.6-luna		
Method	**Plain-text ASR ↓**	**URL Injection ASR ↓**	**Utility Accuracy ↑**	**Plain-text ASR ↓**	**URL Injection ASR ↓**	**Utility Accuracy ↑**	**Plain-text ASR ↓**	**URL Injection ASR ↓**	**Utility Accuracy ↑**
Baseline	19.05	14.29	78.57	9.52	7.14	79.76	4.76	4.76	86.90
Prompt Filter	9.52	14.29	82.14	7.14	4.76	75.00	11.90	9.52	83.33
Simple Static	20.37	15.32	72.64	16.67	14.29	64.29	19.04	14.29	84.52
FIDES	2.38	40.48	0.0	19.05	30.95	32.14	4.76	23.81	59.52
PACE(Ours)	4.76	**7.14**	**92.86**	**0.0**	**0.0**	**91.67**	**0.0**	**0.0**	**95.24**

Table 9: Results on AgentDojo under the official security and utility functions, by attack type. Lower is better for ASR, higher for utility; best ASR per block in bold.

	Qwen-3.8-27B				DeepSeek-V4-Flash				gpt-5.6-luna			
Method	**Direct ASR ↓**	**Import Instructions ASR ↓**	**Tool Knowledge ASR ↓**	**Utility Accuracy ↑**	**Direct ASR ↓**	**Import Instructions ASR ↓**	**Tool Knowledge ASR ↓**	**Utility Accuracy ↑**	**Direct ASR ↓**	**Import Instructions ASR ↓**	**Tool Knowledge ASR ↓**	**Utility Accuracy ↑**
Baseline	48.00	30.21	30.52	62.59	97.62	99.68	99.84	75.68	99.05	99.84	100	73.77
CaMeL	0.32	0.32	0.32	35.14	20.51	24.48	28.14	60.04	31.00	36.09	34.02	57.02
DTA	0.48	0.95	0.95	29.84	24.17	28.46	33.33	54.00	42.13	48.01	55.01	56.01
Spotlighting	98.40	86.30	80.00	62.47	100	92.05	90.14	75.42	100	100	100	70.01
PI Detector	99.50	96.00	90.30	44.40	100	100	92.05	67.99	100	100	99.68	71.01
MELON	0.0	12.70	1.59	49.76	16.06	50.08	20.03	57.98	61.05	43.08	54.05	52.52
Progent	0.64	2.54	3.02	57.76	35.14	42.13	61.53	64.02	48.01	53.10	69.00	62.94
DataSentinel	3.02	35.29	30.37	73.77	62.00	78.06	74.09	85.00	84.10	80.92	75.99	82.95
PACE(Ours)	**0.0**	**0.0**	**0.0**	59.67	**5.09**	**2.70**	**3.02**	75.25	**8.11**	**6.83**	**7.79**	72.05

Table 10: Results on the InjecAgent-Enhanced split. Direct Harm and the two data-stealing stages DS-S1 and DS-S2 are native attack-success rates, as is the aggregate data-stealing column. DS-S2 is conditional and excluded, together with the derived aggregate, from the main-table summary and 79-column comparison. Lower is better; best per block in bold.

	Qwen-3.8-27B				DeepSeek-V4-Flash				gpt-5.6-luna			
Method	**Direct Harm ASR(%) ↓**	**DS-S1 ASR(%) ↓**	**DS-S2 ASR(%) ↓**	**Data-stealing ASR(%) ↓**	**Direct Harm ASR(%) ↓**	**DS-S1 ASR(%) ↓**	**DS-S2 ASR(%) ↓**	**Data-stealing ASR(%) ↓**	**Direct Harm ASR(%) ↓**	**DS-S1 ASR(%) ↓**	**DS-S2 ASR(%) ↓**	**Data-stealing ASR(%) ↓**
Baseline	12.35	23.71	79.07	18.75	19.61	28.13	52.29	14.71	12.55	18.57	50.50	9.38
ReAct	24.31	27.94	92.11	25.74	27.84	37.68	79.02	29.77	16.86	23.35	66.14	15.44
PromptGuard	13.14	24.45	80.45	19.67	21.57	31.80	64.74	20.59	10.00	17.28	56.38	9.74
Prompt-Sandwich	0.59	0.18	0.15	0.12	4.71	14.89	50.62	7.54	2.16	17.10	73.12	12.50
Instructional Prevention	1.37	3.13	58.82	1.83	9.61	19.67	67.29	13.24	3.73	15.26	56.63	8.64
PI Detector	3.13	11.03	20.00	2.21	13.92	27.21	56.08	15.26	5.29	23.71	57.36	13.60
DataSentinel	13.14	24.45	80.45	19.67	22.16	25.18	74.45	18.75	17.06	12.13	65.15	7.90
PromptGuard-2	12.75	23.89	80.77	19.30	20.59	26.29	87.41	22.98	11.57	19.49	82.06	15.99
PACE(Ours)	**0.0**	**0.0**	**0.0**	**0.0**	**0.98**	**3.12**	**0.0**	**0.0**	**0.59**	**2.39**	**30.77**	**0.74**

Table 11: Results on ASB across its five attack settings, with the native utility score. Lower is better for ASR, higher for utility; best ASR per block in bold.

	Qwen-3.8-27B						DeepSeek-V4-Flash						gpt-5.6-luna					
Method	**DPI ASR(%) ↓**	**OPI ASR(%) ↓**	**MP ASR(%) ↓**	**Mixed ASR(%) ↓**	**PoT ASR(%) ↓**	**Utility Accuracy(%) ↑**	**DPI ASR(%) ↓**	**OPI ASR(%) ↓**	**MP ASR(%) ↓**	**Mixed ASR(%) ↓**	**PoT ASR(%) ↓**	**Utility Accuracy(%) ↑**	**DPI ASR(%) ↓**	**OPI ASR(%) ↓**	**MP ASR(%) ↓**	**Mixed ASR(%) ↓**	**PoT ASR(%) ↓**	**Utility Accuracy(%) ↑**
Baseline	84.92	66.00	13.25	88.50	56.25	30.29	74.08	58.25	36.50	70.75	26.00	64.00	81.08	31.50	27.00	72.75	73.50	80.25
Delimiters	88.08	73.75	13.50	86.75	23.75	24.61	88.42	85.50	51.75	93.00	38.00	54.39	81.63	68.25	48.25	88.00	76.25	66.57
Sandwich	85.33	61.75	11.75	88.50	68.75	32.50	81.83	76.25	46.00	59.75	25.75	69.07	71.21	61.50	52.00	63.25	57.25	87.86
Instructional Prevention	71.92	60.75	17.00	82.00	52.75	33.93	58.75	61.75	40.75	78.50	20.75	72.50	54.57	40.75	43.25	83.75	46.00	87.25
Direct Paraphrase	60.08	71.50	13.00	49.50	55.50	62.00	68.08	79.00	49.50	51.50	43.25	81.64	51.07	60.75	40.50	70.75	81.75	81.57
PoT Paraphrase	84.50	70.00	13.75	89.00	59.00	30.82	78.58	87.75	53.75	66.25	56.75	62.96	61.50	82.00	50.25	43.75	92.00	82.43
PoT Shuffle	84.92	71.75	13.00	88.50	99.00	28.57	65.42	43.25	20.75	45.75	23.50	57.43	57.43	51.00	34.50	59.25	43.00	67.04
PACE(Ours)	**0.42**	**0.00**	**0.00**	**0.00**	**0.00**	29.57	**7.08**	**4.75**	**7.25**	**10.5**	**3.25**	64.64	**5.07**	**3.75**	**5.50**	**8.75**	**6.75**	81.63

Table 12: Results on AgentDyn across its three task domains, with the native utility score. Lower is better for ASR, higher for utility; best ASR per block in bold.

	Qwen-3.8-27B				DeepSeek-V4-Flash				gpt-5.6-luna			
Method	**Shopping ASR(%) ↓**	**Github ASR(%) ↓**	**DailyLife ASR(%) ↓**	**Utility Utility(%) ↑**	**Shopping ASR(%) ↓**	**Github ASR(%) ↓**	**DailyLife ASR(%) ↓**	**Utility Utility(%) ↑**	**Shopping ASR(%) ↓**	**Github ASR(%) ↓**	**DailyLife ASR(%) ↓**	**Utility Utility(%) ↑**
Baseline	11.11	17.22	32.50	63.39	12.78	17.22	19.50	65.00	7.78	10.56	19.50	74.82
Repeat User Prompt	6.67	15.56	34.50	61.25	22.78	10.00	31.50	61.43	12.28	10.00	31.50	70.54
Spotlighting	3.33	13.89	5.50	65.00	13.33	7.22	3.00	58.21	17.22	15.00	27.50	64.46
Tool Filter	0.0	0.0	0.05	7.86	1.67	2.78	2.00	8.04	2.78	3.89	6.50	9.11
PI-Detector	0.0	0.0	0.05	1.25	0.0	0.0	3.50	0.08	7.22	1.67	2.50	2.14
PI-Guard	0.0	0.0	5.50	6.79	2.78	6.67	13.50	6.60	3.89	8.33	1.00	3.93
PromptGuard-2	6.11	15.56	33.00	58.42	9.44	17.22	23.50	60.36	3.33	12.78	25.50	68.57
CaMeL	0.0	0.0	0.0	0.0	1.11	0.0	5.50	3.75	0.0	0.0	0.0	0.0
Progent	1.67	1.67	11.50	26.96	5.56	1.67	8.50	9.82	2.22	1.67	5.50	11.25
DRIFT	0.0	0.56	0.50	29.82	2.78	0.0	1.50	13.39	0.0	3.33	4.50	14.64
PACE(Ours)	**0.0**	**0.0**	**0.0**	60.54	**0.0**	0.06	**0.05**	63.57	**0.0**	**0.0**	**0.0**	73.57

Table 13: Results on PASB for indirect prompt injection and the four memory subsets, with the native IPI utility. Lower is better for ASR, higher for utility; best ASR per block in bold.

Method	Qwen-3.8-27B						DeepSeek-V4-Flash						gpt-5.6-luna					
	IPI ASR(%) ↓	Short Read ASR(%) ↓	Short Modify ASR(%) ↓	Long Read ASR(%) ↓	Long Modify ASR(%) ↓	Utility Accuracy(%) ↑	IPI ASR(%) ↓	Short Read ASR(%) ↓	Short Modify ASR(%) ↓	Long Read ASR(%) ↓	Long Modify ASR(%) ↓	Utility Accuracy(%) ↑	IPI ASR(%) ↓	Short Read ASR(%) ↓	Short Modify ASR(%) ↓	Long Read ASR(%) ↓	Long Modify ASR(%) ↓	Utility Accuracy(%) ↑
Baseline	0.0	85.00	62.50	62.50	75.00	98.47	0.0	67.50	72.50	60.00	92.50	99.24	2.29	55.00	42.50	50.00	97.5	100.0
Delimiters	0.0	25.00	0.0	45.00	20.00	99.23	0.0	30.00	55.00	42.50	32.50	98.47	1.53	22.50	35.00	30.00	35.00	99.24
Sandwich	0.0	85.00	70.00	65.00	72.50	99.24	0.0	72.50	85.00	57.50	87.50	98.47	3.82	40.00	25.00	42.50	82.50	100.0
Instructional Prevention	0.0	17.50	0.0	17.50	0.0	97.67	0.0	20.00	17.50	27.50	17.50	96.95	6.87	17.50	7.50	27.50	25.00	98.47
PI-Detector	0.0	85.00	50.00	70.00	67.50	99.24	0.0	77.50	65.00	60.00	82.50	100.0	2.29	67.50	35.00	40.00	70.00	98.47
PI-Guard	0.0	87.50	65.00	70.00	67.50	100.0	0.0	82.50	77.50	55.00	87.50	99.27	3.05	77.50	45.00	50.00	80.00	96.95
PromptGuard-2	0.0	85.00	45.00	62.50	57.50	100.0	0.0	72.50	55.00	50.00	72.50	97.71	83.97	70.00	40.00	32.50	62.50	99.24
Progent	0.0	80.00	5.00	65.00	12.50	100.0	0.0	77.50	12.50	67.50	22.50	100.0	12.98	62.50	20.00	62.50	35.00	98.47
FIDES	0.0	0.0	27.50	0.0	40.00	99.24	0.0	5.00	15.00	2.50	35.00	98.47	14.50	22.50	32.50	12.50	65.00	99.27
ToolShield	0.0	47.50	42.50	57.50	55.00	98.44	0.0	50.00	35.00	55.00	60.00	95.42	9.16	62.50	42.50	70.00	77.50	96.18
CaMeL	0.0	71.75	13.00	88.50	99.00	28.57	0.0	7.50	15.00	37.50	52.50	98.47	5.34	35.00	62.50	32.50	67.50	99.24
PACE(Ours)	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	97.69	**0.0**	**0.0**	**0.0**	**2.50**	**5.0**	99.24	**0.0**	**0.0**	**2.50**	**0.0**	**2.50**	100.0

Table 14: Results on MCPTox for the three poisoning templates, with the native refusal rate. Lower is better for both; refusal is reported so that a low attack-success rate cannot be read without its cost. Best ASR per block in bold.

Method	Qwen-3.8-27B				DeepSeek-V4-Flash				gpt-5.6-luna			
	Template-1 ASR(%) ↓	Template-2 ASR(%) ↓	Template-3 ASR(%) ↓	Refusal Refusal Rate(%) ↓	Template-1 ASR(%) ↓	Template-2 ASR(%) ↓	Template-3 ASR(%) ↓	Refusal Refusal Rate(%) ↓	Template-1 ASR(%) ↓	Template-2 ASR(%) ↓	Template-3 ASR(%) ↓	Refusal Refusal Rate(%) ↓
Baseline	22.39	8.84	32.52	0.16	56.54	52.61	56.06	0.42	23.04	26.10	86.93	27.96
Metadata Sanitization	5.94	1.38	17.41	0.15	13.09	9.91	28.60	0.37	5.76	5.43	91.48	26.67
PI-Detector	20.53	7.37	25.22	0.40	85.34	4.59	50.19	1.63	48.17	3.97	95.45	30.14
PI-Guard	11.59	3.09	11.07	0.91	50.79	27.56	30.68	3.15	34.03	9.19	63.64	46.32
PromptGuard-2	21.43	7.68	27.08	0.63	63.35	55.95	53.98	1.68	60.73	37.16	97.72	31.02
Tool Allowlist	0.72	0.0	0.18	2.46	15.18	13.57	2.84	7.66	6.81	5.85	16.29	55.78
MCIP Guardian	0.51	0.40	0.19	54.23	9.95	5.01	1.89	76.72	7.33	3.55	10.04	81.22
Progent	1.52	1.98	8.80	45.88	20.42	15.24	16.48	66.85	19.90	11.27	46.40	78.43
PACE(Ours)	**0.0**	**0.0**	**0.0**	1.97	**2.62**	**1.46**	**0.76**	0.58	**0.52**	**0.0**	**2.65**	28.55

values into zeros. The audit settles the question of sample selection and does not by itself explain why the official scorer returns zero for those six arms. We therefore read that row as an implementation-sensitive diagnostic and interpret it only together with trajectory completion, the refusal column, and the native scorer semantics. The same audit withdraws the earlier paired values of 65/280 utility and 106/280 refusal, which corresponded to the first forty rows of each run file rather than to the forty trajectories the frozen selector names.

Coverage and certificate health. The quantities in Table 16 are instrumentation fields, not numerical estimates of semantic coverage. The A7 records contain 32867 PACE decisions across eight benchmarks, 26272 with full witnesses across six; MCPTox and MSB retain compact records. All nine obligations are evaluated on 25972 schema-covered decisions, yielding 233748 predicate evaluations, including failures. There are 22685 nonempty candidate cuts, 5911 terminal blocks, 365 rewrite-bearing decision records, and 22191 named adapter activations. These count different objects and cannot be summed; a rewrite field is not proof that a rewrite executed. AgentDyn has 6647 guarded-call records, including failures and recovery attempts: every candidate path certificate passes, C certifies 6190, and 5842 have nonempty cuts containing 6080 edges. No nonterminal rewrite executes, so restored utility there comes from the combiner overriding P's terminal blocks. Existing tests include positive and negative cases for each obligation, forged witnesses and capabilities, and 20 mutations each for token forgery, schema drift, stale certificates, parameter smuggling, and omitted adapters. These check interface behavior, not semantic completeness or distributed durability.

Reporting conventions. The displayed benchmark cells are rounded point estimates; native numerators and denominators belong to the run records. Provider refusal, tool failure, policy intervention, and native task failure are separate terminal states, so a safe model refusal is never attributed to **PACE** and a broken tool episode never counts as a defense success. Cells excluded from a primary table under the technical-failure ceiling of Section E.4 are listed with endpoint timeout, evaluator error, missing hook, sandbox failure, and malformed certificate as mutually exclusive causes, with retried and non-retried denominators both recorded. In the reduced-scale adaptive experiment, undefended successes are 10/10 Slack, 7/10 Travel, and 4/10 Workspace, totalling 21/30 (70.0%; two-sided

Table 15: Results on MSB by attack stage, with the native PUA and NRP scores. Attack success is lower better; PUA and NRP are higher better. Cells marked – were not produced by the official scorer for that configuration.

Method	Qwen-3.8-27B						DeepSeek-V4-Flash						gpt-5.6-luna					
	Plan ASR(%) ↓	Call ASR(%) ↓	Response ASR(%) ↓	Multi-Stage ASR(%) ↓	PUA Accuracy(%) ↑	NRP Accuracy(%) ↑	Plan ASR(%) ↓	Call ASR(%) ↓	Response ASR(%) ↓	Multi-Stage ASR(%) ↓	PUA Accuracy(%) ↑	NRP Accuracy(%) ↑	Plan ASR(%) ↓	Call ASR(%) ↓	Response ASR(%) ↓	Multi-Stage ASR(%) ↓	PUA Accuracy(%) ↑	NRP Accuracy(%) ↑
Baseline	5.67	38.75	26.35	18.70	46.34	36.29	61.33	100.0	29.68	52.00	91.24	49.29	67.00	98.75	9.03	50.20	89.08	54.51
MCIP Guardian	13.33	50.00	14.03	8.40	59.15	55.70	8.00	77.50	26.13	37.30	96.55	51.32	29.33	76.25	8.55	36.90	92.34	56.87
PI-Detector	16.67	38.75	7.02	5.20	54.36	49.83	55.33	53.75	18.55	48.40	94.81	51.18	65.00	70.00	7.10	47.10	90.85	56.22
PI-Guard	14.00	37.50	13.87	9.00	56.42	53.26	32.00	57.50	36.94	36.90	91.98	49.95	40.67	66.25	10.00	38.50	89.97	54.25
PromptGuard-2	14.67	38.75	25.00	17.60	47.85	40.87	61.67	93.75	68.06	55.20	93.27	45.57	78.00	87.50	16.29	53.60	91.66	50.58
Tool Filter	15.00	0.0	30.81	14.20	47.34	43.48	9.67	42.50	59.35	51.70	93.65	47.46	22.33	52.50	13.71	50.60	91.04	49.75
Progent	11.00	38.75	5.65	4.30	50.98	47.37	47.67	97.50	26.45	30.60	97.08	46.67	58.33	96.25	7.90	34.40	90.51	47.64
FIDES	1.33	52.50	10.16	10.60	45.88	40.40	29.00	100.0	23.71	49.80	90.66	52.62	34.67	100.0	8.87	48.20	87.73	56.45
ToolShield	2.71	38.75	7.02	5.20	53.94	45.89	4.33	100.0	20.16	56.00	96.24	53.58	9.00	100.0	6.77	65.20	–	–
PACE(Ours)	**0.0**	**2.50**	**0.0**	**0.0**	**60.14**	**60.01**	**0.0**	**3.75**	**0.16**	**1.40**	**98.83**	**59.84**	**0.33**	**3.75**	**0.0**	**0.7**	**94.87**	**60.61**

95% Wilson interval [52.1, 83.3]%). Twelve first succeed with the static seed and nine with search-generated payloads at iterations 3–39, using 632 episodes in total. **PACE**-P yields 0/30 (interval [0, 11.4]%) after 1500 episodes. Its boundary trace records 157 exact out-of-authority target calls in 126 episodes across 14 pairs, all blocked before execution: 149 fail AUTH and BUDGET, and eight additionally fail BIND and CTRL. None uses R2. Two controlled checks are excluded from ASR: direct boundary dispatch blocks all 37 frozen target-call patterns, and submitting a target call in fresh contexts carrying payloads from 1003 selected target-free episodes yields the same refusal. These do not increase the 30-case attack sample. The sample covers only out-of-authority goals in three suites, uses the target checkpoint as mutator, and permits 50 episodes per case; stronger mutators and attacks on parsing or unrepresented transformations remain untested.

Table 16: Certificate-health quantities recorded per episode.

Quantity	Statistic	Interpretation
`forward_simulation_observed=true`	Rate + CI	Last event required no graph expansion; not proof of H1
`op.unknown` expansions	Histogram/quantiles	Observed abstraction incompleteness
Unbindable minimum cuts	Rate + CI	Graph separation lacking an executor action
Stale-state rejection	Rate + CI	Freshness check exercised before dispatch
Effect-obligation failures	Histogram by obligation	Which of the nine checks fails, and how often
Bound action executed	Rate + CI	Observable executor agreement relevant to H4
Verifier reconstruction mismatch	Rate + cause	Certificate construction or serialization defect