CALLWITNESS: POST-RESOLUTION AUTHORIZATION FOR SECRET-BEARING AGENT TOOL CALLS

Anonymous authors
Paper under double-blind review

ABSTRACT

A ticket read produces a secret-labeled handle, a trusted transformation produces a derived handle, and an upload wrapper resolves an alias to its final endpoint. The security-critical edge connects that derived value to that resolved sink; neither endpoint authorization nor value lineage alone expresses the flow. CallWitness delays credential release until a broker-derived value graph and wrapper-attested destination form one authorization record. Unforgeable handles, lineage-preserving transformations, and broker-owned credentials realize this record across nine tool schemas.

The evaluation separates necessity, end-to-end effect, and causal attribution. On a paired factorial slice, literal-token seeded-run attack success is 1.4% with the joined record, compared with 29.8% for destination-scoped authority alone and 21.4% for lineage alone. Across 847 attacks, CallWitness records 11/847 successes (1.3%, 95% CI 0.7–2.2%) versus 43/847 (5.1%) for matched intent approval and 21.3–28.4% per channel for a DLP proxy. A browser attestation fork then reuses attack text, sampled calls, lineage, and policy while changing only sink visibility: any-of-five success rises from 5/320 (1.6%) to 75/320 (23.4%), or 15.0×. The effect recurs with GPT-5.4 and Claude Sonnet 5 integrations; for broker-mediated, destination-declaring calls, CallWitness delivers 94.3% benign completion at 4.2 ms median added latency.

1 INTRODUCTION

One credentialed call evolves through four runtime states. A ticket read returns handle h_0 with `secret` lineage; normalization creates h_1 with the same ancestry; an upload selects alias a; and the wrapper resolves a to concrete sink S. Credential release before the final transition sees a value-to-class edge $h_1 \to a$; release after it can evaluate the actual edge $h_1 \to S$. When S is attacker-controlled, that one-edge difference determines whether the runtime can stop the side effect. Indirect prompt injection can choose the hostile transition sequence (Greshake et al., 2023; Liu et al., 2024; Yi et al., 2025; Zhan et al., 2024; Shmatikov et al., 2023); the runtime's credentials realize it. Agents routinely create such source-to-sink paths across pages, files, messages, and remote services (Schick et al., 2023; Yao et al., 2023; Nakano et al., 2021; Significant Gravitas, 2023; OpenAI, 2023), making the final graph edge a consequential security boundary.

The critical boundary is the last decision before credential injection. CallWitness represents that decision by

$$E_{\text{auth}}(c) = \big(\text{consumed handles},\ \text{handle-derived lineage},\ \text{wrapper-resolved sink}\big).$$

A credential-holding broker derives the consumed set from argument expressions, propagates lineage through trusted transformations, and joins it with the recipient, URL, project, browser origin, or upload endpoint produced by the wrapper. The model cannot erase a dependency by omitting a handle identifier or restating protected bytes as a payload literal. Policy evaluates the joined record immediately before release. Under complete mediation, lineage closure, correct sink resolution, and denial of unendorsed secret-to-external flow, Proposition 1 establishes mediated-call containment.

Existing controls attach to different transitions in this trace. Prompt/data separation and guardrails act before the tool proposal (Chen et al., 2025; Rebedea et al., 2023); permission systems govern the tool and pre-resolution target (Wu et al., 2026; Li et al., 2025); CaMeL, PACT, and runtime provenance

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.

monitors follow value influence or argument provenance (Hayes et al., 2025; Wang et al., 2026; Lu et al., 2025). CallWitness contributes the terminal edge $h_1 \to S$: the broker's consumed-value graph and the wrapper's resolved sink become one enforceable pre-credential record. An attestation fork also makes the edge measurable, reusing attack, sampled call, lineage, and policy byte-for-byte while substituting the pre-resolution class for S in one arm.

The field removal result is decisive: literal-token any-of-five success changes from 5/320 (1.6%) with the resolved browser sink to 75/320 (23.4%) without it, a 21.9-point gap. On the broader 847-attack benchmark, the full record reaches 1.3% success versus 5.1% for matched intent approval. Section 5 then tests the necessity of the join with a paired factorial intervention, carries the visibility test across current providers and dispatchers, and measures benign operation. Together, the authorization object and field-removal protocol make post-resolution execution evidence a concrete control point for secret-bearing agent tool calls.

2 From Declared Actions to Materialized Calls

2.1 The Last Pre-Credential Decision

We target Reason–Act agents that read context, select tools, and issue structured calls (Yao et al., 2023). Browser, file, HTTP, ticket, email, and upload tools often carry authority broader than one task, so the runtime must restore least privilege at each use (Lampson, 1974; Levy, 1984; Saltzer & Schroeder, 1975). An intent decision operates on the proposed action and declared scope. The last pre-credential decision occurs after arguments and destinations have materialized; it can additionally bind consumed values and resolved targets. Aliases, redirects, transformations, and recipient expansion make these records observably different even when the proposed tool call is unchanged.

2.2 Threat Model

Some content ingested by the agent is adversary-controlled Greshake et al. (2023); Liu et al. (2024); Zhan et al. (2024), including an HTML page, markdown document, email body, or ticket comment. After the LLM reads this content, the adversary can induce it to follow malicious instructions, but cannot directly execute host code, compromise the broker, or modify the policy store. The evaluated goal, literal-token tool exfiltration, is to deliver the generated 32-byte value represented by a broker-labeled secret to an HTTP endpoint, outbound email recipient, ticket destination, or upload target controlled by the attacker. The model may choose tools and arguments adversarially after ingesting the attack content.

2.3 Security Scope

The protected value is broker-labeled data whose lineage contains `secret`. Protected bytes remain inside the trusted value store; the model receives an opaque handle and a redacted view of surrounding content. The broker, value store, policy store, labels, and credential boundary are trusted, and every evaluated egress passes through that boundary. The end-to-end experiment exercises wrapper and adapter conformance, including complete value attestation and correct sink resolution. Immediately before execution, policy evaluates E_{auth} and permits secret egress only through an explicit endorsement rule.

The evaluated regime is broker-mediated, destination-declaring side effects: payload arguments are materialized from handles, and wrappers expose final targets before credential release. It directly covers browser, HTTP, email, ticket, file, and object-upload workflows across the nine schemas in Appendix Table 2.

3 CallWitness Design

CallWitness authorizes a call in execution order. A manifest constrains the proposed operation, a broker-derived value graph identifies the arguments actually consumed, and the wrapper attests the target reached after resolution. Credential release occurs only after these records are joined.

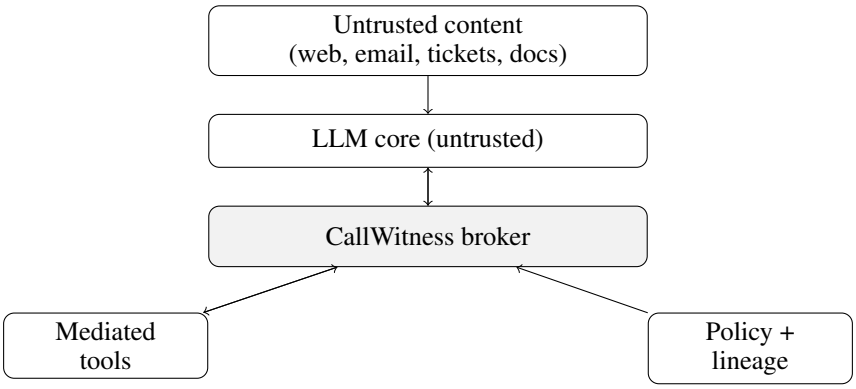

Figure 1: CallWitness data path. The broker holds tool credentials, keeps protected values behind opaque handles, propagates lineage through trusted transformations, and authorizes the wrapper-resolved call.

Figure 1 places credentialed authority at the brokered tool boundary. The model proposes operations but never receives tool credentials. The broker materializes value-bearing arguments, the wrapper attests the post-resolution destination, and policy evaluates the joined record before the side effect occurs.

3.1 Attested Value Flow

Every value returned by a mediated read receives an unforgeable broker handle and an initial lineage label. Secret-bearing spans remain in the broker's value store; the model observes an opaque typed handle together with a redacted view of the surrounding public content. A side-effect wrapper accepts an argument expression whose leaves are broker handles or schema-designated public literals. Operations over protected values—including slicing, concatenation, normalization, and encoding—are executed by an allowlisted transformation service inside the broker. The output receives a fresh handle and the union of its input lineage.

The broker derives the consumed set from this expression graph rather than trusting a handle list supplied by the model. A manifest may authorize a subset of handles, but it cannot remove a dependency from the derived set. Free-form literals are accepted only in fields designated public by the tool schema; value-bearing payload fields require a handle or a broker-executed transformation. An unresolved dependency, unsupported transformation, or handle outside the manifest is rejected before destination resolution. This construction covers direct copies and model-requested encodings without depending on the compromised model to report lineage.

3.2 The Final Authorization Record

For a proposed call c, the final authorization record is

$$E_{\text{auth}}(c) = \big(I_c, \{\mathsf{prov}(O) : O \in I_c\}, S_c\big),$$

where I_c is the set of handles reachable from the call's argument expressions and S_c is the fully resolved side-effect sink. The wrapper computes S_c after URL redirects, recipient aliases, project and ticket identifiers, browser subresources, and upload-target resolution. Immediately before credential injection, the policy engine consumes the broker-derived tuple and records its decision. Unlike the intent manifest, neither I_c nor S_c is asserted by the model.

3.3 Intent Manifests

An *intent manifest* accompanies each mediated call, declaring in structured form its permitted input handles, its possible output value category, and the destinations permitted for any side effects. By retaining only fields that require broker checks, the manifest is intentionally smaller than the agent's reasoning trace.

Table 1: Intent manifest fields checked before credentials are exercised.

Field	Description
tool	Identifies the tool; examples include `http`, `file`, or `ticket`.
action	Specifies the tool's operation, for example `get`, `post`, or `read`.
resource	Identifies a concrete resource target, for example a URL, file path, or ticket ID.
inputs	Lists IDs of broker handles permitted as call inputs.
output_cat	States the output category; examples include `public` or `customer-pii`.
destinations	Specifies permitted sinks for side effects: domains, recipients, local-only, or `none`.

Two common forms of ambient authority are eliminated by the manifest: inputs stops the agent from implicitly using its entire context window as an eligible input for any tool call, while destinations specifies the intended scope of egress. At credential release, the broker checks these allowlists against the graph-derived consumed set and wrapper-resolved sink in E_{auth}. The manifest constrains execution; it is not the source of the execution evidence.

3.4 Destination-Aware Capabilities and Value Lineage

Capabilities grant permission over both the operation and the sink:

$$\langle \texttt{tool}, \texttt{action}, \texttt{resource_pattern}, \texttt{destination_pattern} \rangle. \tag{1}$$

A workflow can thus permit workspace-scoped `file.read` while prohibiting all egress, permit `http.get` for an internal documentation domain, or restrict `ticket.post` to one specific project. Classic capability scoping is thereby extended: a side effect's destination becomes an object relevant to security Watson et al. (2012); Miller et al. (2008); Mettler et al. (2010).

Each value also carries a declared output_cat and lineage $\mathsf{prov}(O) \subseteq \{\texttt{untrusted}, \texttt{internal}, \texttt{secret}\}$. A wrapper output starts with a singleton lineage set; for a derived value, $\mathsf{prov}(O') = \bigcup_i \mathsf{prov}(O_i)$ governs inheritance unless a tag is removed through endorsement. Allowing a side effect from O' to sink S requires policy permission at S for both its declared category and its inherited lineage. Capabilities answer which operation may touch which sink; lineage answers which ancestry tags now flow there Denning (1976); Zeldovich et al. (2006); Krohn et al. (2007); Pasquier et al. (2017). The evaluation shows that either mechanism alone leaves large residual gaps.

3.5 Policies, Trusted Mechanism, and Security Scope

Capability tuples, flow rules, and conservative defaults are specified by policy. The evaluated configurations restrict file reads to a task workspace and deny outbound HTTP in the absence of an allowlist match; values with `untrusted` or `secret` in their lineage cannot flow to external sinks except under an explicit endorsement rule. Here a sink is external to the workflow when it matches no permitted destination pattern; a matched storage, recipient, project, or domain pattern is the policy-approved scope for that task. The trusted path parses manifests, materializes handle expressions, propagates lineage, resolves destinations, checks policy, and releases credentials.

The visibility intervention uses the same rules through two policy inputs. Before resolution, the original target is matched to a manifest destination pattern; the identifier of the matching pattern is its destination class, and an unmatched target receives `unknown` and is denied. In the opaque arm, this identifier replaces S_c: policy treats the class as terminal and never receives the post-redirect URL or origin. In the visible arm, the wrapper first expands aliases, redirects, and subresources and rematches the concrete sink against the same destination patterns. Thus an initially allowlisted storage alias can pass the opaque record even when its resolved endpoint fails the allowlist and is denied in the visible record. No rule, lineage label, sampled call, or initial target changes between arms.

Proposition 1 connects the final record to the enforced security property under complete mediation, correctly surfaced destinations, lineage closure, and the stated deny rule over secret-derived values.

Proposition 1 (Mediated-call containment)**.** *Assume that (i) the broker mediates every tool invocation that produces network egress or any other external side effect, (ii) every value-bearing argument is a broker handle or the result of a broker-executed transformation, so E_{auth} contains its complete handle-derived lineage, (iii) its sink is resolved correctly by the tool wrapper, and (iv) values with* `secret` *in their lineage are denied flows to external destinations by policy except under an explicit endorsement rule. Under these assumptions, no execution can deliver a* `secret`*-derived value through a mediated tool call to an external sink without endorsement, even if the LLM is fully compromised.*

Every mediated side effect passes through the broker; the handle graph makes lineage independent of model declarations; and condition (iv) denies each `secret`→external flow unless endorsement intervenes. Appendix B gives the complete proof. The sink oracle then tests the complete runtime path: it records success only when the generated token arrives at the designated external sink. Figure 2 targets sink resolution, while Figure 4 separates lineage from destination-scoped authority.

4 Implementation

We implement CallWitness as a standalone Rust broker that interposes on the last dispatch step of existing runtimes without changing their planning loops. Adapters preserve each runtime's native intent path while converting materialized calls into handle expressions and wrapper-resolved E_{auth} records.

The gRPC broker runs over a Unix domain socket and contains the manifest validator, opaque-handle value store, allowlisted transformation service, YAML policy engine, tool registry, and decision logger. It walks each argument expression, materializes values, unions all dependency labels, and checks the derived handles against the manifest before injecting broker-owned credentials. Wrappers expose protected spans as typed opaque references; schemas mark literal-safe fields, while payload-bearing fields reject untracked values.

Thin adapters replace LangChain's `ToolExecutor` and wrap AutoGPT's command dispatcher. Nine schemas cover browser navigation, HTTP, email, files, tickets, and object upload (Appendix Table 2); browsers additionally intercept navigation and subresource requests to expose their final origins. A denial returns the deciding rule and triggering handle or destination. JSON-lines records contain the tool/action, resource, resolved destination, handles, output category, lineage summary, and rule, without secret contents.

The 2,847-LoC Rust binary runs as a sidecar and adds 4.2 ms median latency per mediated call. Adapters change across the 2026-stack replications, while value materialization, authorization, lineage, credential release, and logging remain provider independent.

5 Evaluation

The evaluation follows the authorization record from identification to operation. A field-removal test measures the causal value of the resolved sink; end-to-end and factorial studies compare authorization objects; provider, dispatcher, policy, and adaptive tests probe portability; benign workloads measure operating cost.

5.1 Experimental Setup

The controlled stack uses Ubuntu 22.04, Docker 24.0.7, CallWitness compiled with `rustc` 1.75, and dispatch hooks in LangChain 0.1.16 and AutoGPT 0.5.1. GPT-4-Turbo (`gpt-4-1106-preview`) serves as the model, using native function calling with `tool_choice=auto`; its sampling settings are `temperature=0.2`, `top_p=0.95`, and `max_tokens=4096`. Contemporary replications use GPT-5.4 through the OpenAI Responses API and Agents SDK 0.21.0, and Claude Sonnet 5 through the Anthropic Messages API and LangGraph 1.2.11; both were evaluated on August 22, 2026.

The attack benchmark has 847 instances crossing four ingestion channels (HTML, markdown, email, and ticket comments), four sinks (HTTP, outbound email, ticket posts, and file uploads), and four payload families (role-play, encoding, multi-turn, and paraphrase) (Schulhoff et al., 2023; Liu et al., 2024; Zhan et al., 2024). Its nine tool schemas are browser navigation, HTTP request, email read/send, file read/write, ticket read/post, and object upload. Templates are instantiated with benign objectives from the corresponding tool workflows, then filtered only when the wrapper cannot express both the objective and candidate sink. Instance membership, attack text, generated 32-byte token, and success oracle are frozen before defense scoring. The 320-instance destination-visibility and 286-instance factorial analyses are fixed common-support slices of this benchmark: each retains only attacks that all conditions in the corresponding matched comparison can express.

The estimand is literal-token exfiltration: success requires the exact generated 32-byte value to appear at the designated external sink. Encoding and paraphrase name injection-payload families, not alternate secret-recovery oracles. Each instance runs under five agent-layer seeds; main and challenge ASR marks an instance successful if any run leaks. Conditions share the sampled trajectory until an authorization result changes the history, then continue independently under the same sink oracle. The factorial study reports seeded-run ASR over $5N$ paired runs. The utility set contains 312 tasks in five categories over the same schemas; completion credits automated execution, and escalation records a required endorsement boundary.

All conditions share benchmark instances, model settings, seeds, tool schemas, and the sink oracle. The field-removal, factorial, and matched-approval comparisons also share the handle materializer and wrappers; within each matched comparison, only authorization logic or the evidence supplied to it changes. Architectural baselines are prompt-only policy, allowlist egress, NeMo Guardrails 0.8.1 (Rebedea et al., 2023), a regex/rate-limited DLP proxy, and matched approval mediation. The approval condition asks the agent to submit a structured tool, action, object class, and destination-class declaration; an approval service applies the same deny goal before issuing a release token. The final dispatcher retains that token and declared class, but not broker handle IDs or the post-alias/redirect sink. An endorsement permits one declared flow and is logged identically in both systems. Empirical comparisons in Figure 3 refer to these implemented conditions; the execution-layer systems discussed in Section 6 differ in decision point or evidence object and provide mechanism-level context.

Rates use 10,000 task-clustered bootstrap resamples and paired resampling where conditions share instances. Confidence intervals are 95%; point estimates are empirical proportions at the unit stated for each study. Appendix Section A specifies slice construction, run-level aggregation, transfer denominators, and the 127,438-call latency trace.

Appendix Table 4 aligns each condition's credential-release evidence, stress diagnostic, analysis unit, and outcome.

5.2 RQ1: Does Resolved-Sink Visibility Control Exfiltration?

Figure 2 changes one field of E_{auth} on $N = 320$ matched attacks. The intervention forks at wrapper attestation: attack text, sampled calls, handle-derived lineage, and policy are byte-identical, while the opaque arm replaces the post-resolution browser target with the destination-pattern class matched by the original request. Unknown classes fail closed; an initially allowed class remains terminal only in the opaque arm, whereas the visible arm rematches the resolved sink as specified in Section 3. The visible arm records 5/320 successful instances (1.6%, CI 0.7–3.7%); the opaque arm records 75/320 (23.4%, CI 19.1–28.3%). The paired increase is 21.9 points (CI 17.4–26.6), with 15.0× relative risk (CI 8.1–29.7).

The HTTP (0.9% ASR, CI 0.3–2.7%) and email (1.2%, CI 0.5–3.1%) references remain in the same low-ASR regime as the visible browser. The sharp change confined to the opaque browser arm identifies post-resolution destination visibility as the changed execution fact.

5.3 RQ2: Does the Execution Record Matter End to End?

Figure 3 shows stable separation in every ingestion channel. Prompt-only control yields 78.9–86.7% ASR, allowlisting 53.8–62.1%, guardrails 34.6–41.8%, the DLP proxy 21.3–28.4%, and CallWitness 0.7–1.8%. Because all are any-of-five instance rates, the ordering cannot be attributed to a denominator change.

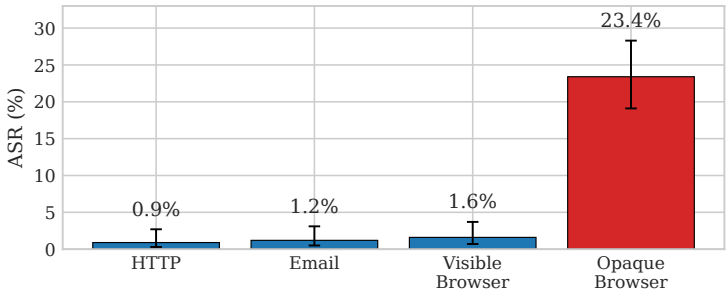

Figure 2: Same-policy intervention on 320 matched attacks. Error bars are paired-bootstrap 95% intervals. Only the final resolved browser sink is hidden in the opaque arm; HTTP and email provide visible-sink references.

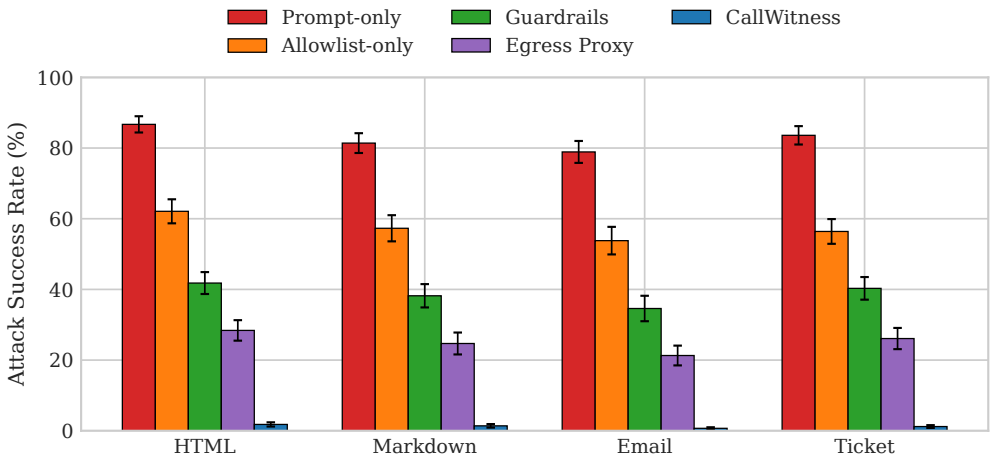

Figure 3: Any-of-five per-channel ASR on 847 attack instances; bars show task-clustered 95% bootstrap intervals.

The pooled matched comparison measures the value of retaining execution facts beyond the intent declaration: CallWitness has 11/847 successes (1.3%, CI 0.7–2.2%) versus 43/847 (5.1%, CI 3.7–6.6%) for approval mediation. The paired difference is 3.8 percentage points (CI 2.4–5.2); relative risk is 0.26 (CI 0.13–0.49). Structured intent helps substantially; retaining handle-derived lineage and the resolved sink reduces the remaining risk by 74% relative to the matched approval mediator on the same attacks, seeds, deny goal, and tool surface. Appendix A.5 confirms the same ranking before any-of-five aggregation. The 11 CallWitness successes bound empirical violation of the scoped deny goal at 1.3% under the full broker–adapter–wrapper stack. By Proposition 1, each residual belongs to at least one diagnostic class: incomplete mediation or lineage, incorrect sink attestation, or a policy/endorsement release; it is not a counterexample to containment when all four premises hold. The logged handles, resolved sink, and deciding rule make these classes observable. Appendix Table 3 records 98.2–98.9% blocking across HTTP, email, ticket, and upload attempts, so no single sink carries the pooled result.

5.4 RQ3: How Does Value Lineage Contribute?

Figure 4 uses all five seeded runs for each of $N = 286$ common-support attacks, with intervals clustered by instance. ASR is 1,183/1,430 (82.7%) without enforcement (CI 77.9–86.6%), 426/1,430 (29.8%) with destination-aware capabilities alone (CI 24.8–35.3%), 306/1,430 (21.4%) with lineage alone (CI 17.0–26.5%), and 20/1,430 (1.4%) with both (CI 0.5–3.5%). Capability-only exceeds full enforcement by 28.4 points (CI 23.1–33.8); lineage-only exceeds it by 20.0 points (CI 15.1–25.3). Thus neither mechanism substitutes for the joined record. The additive risk-scale interaction

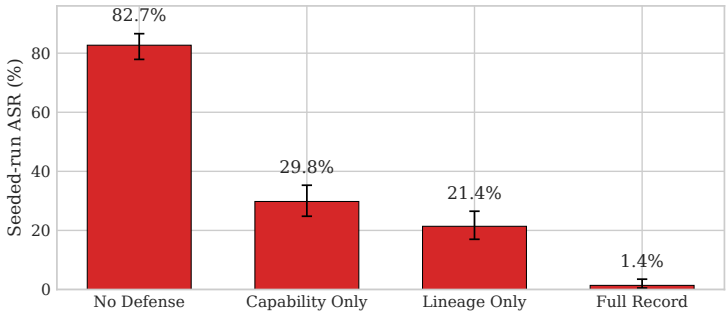

Figure 4: Seeded-run ASR on the paired 286-instance factorial slice; error bars are instance-clustered 95% bootstrap intervals. The full record's 1.4% is a controlled-slice estimate, distinct from the 1.3% any-of-five ASR over all 847 attacks.

$\mathrm{ASR}_{\mathrm{none}} - \mathrm{ASR}_{\mathrm{cap}} - \mathrm{ASR}_{\mathrm{prov}} + \mathrm{ASR}_{\mathrm{full}}$ is 32.9 points (CI 25.4–40.7), indicating overlapping rather than independent reductions on this scale.

5.5 RQ4: Portability, Policy Sensitivity, and Adaptive Attacks

Changing both provider and dispatch hook preserves the visibility effect. GPT-5.4 through the OpenAI Agents SDK records 4/304 visible successes (1.3%) versus 59/304 (19.4%), with 92.4% completion on 144 benign tasks. Claude Sonnet 5 through LangGraph records 5/296 (1.7%) versus 64/296 (21.6%), with 91.9% completion on 140 tasks. Thus the primary LangChain executor, OpenAI callback, and LangGraph `ToolNode` all place visible-sink ASR below 2% and opaque-sink ASR near 20%.

An independently authored policy, frozen from the nine schemas and deny goal before outcomes were revealed, records 11/178 successes (6.2%) and 90.6% completion on 96 tasks. The 14 disagreements with the primary bundle concentrate in alias normalization, cross-project scope, and endorsement requirements. This result quantifies policy-authoring sensitivity and identifies the rule categories that determine whether the execution record yields the intended decision. This separately evaluated 178-instance policy-transfer slice is not the 847-instance primary result, despite the coincident numerator of 11.

The adaptive package in Appendix A.4 freezes secrets, wrappers, policy goals, and five-seed semantics while attack designers select disjoint templates after observing the authorization record and baseline behavior. Visible CallWitness remains at 9/256 successes (3.5%, CI 1.7–6.2%), compared with 24/256 for approval (9.4%), 64/256 for the proxy (25.0%), and 58/256 for the opaque arm (22.7%). On this held-out adaptive slice, the visible rate rises to 3.5% from the primary study's 1.3%, while retaining the ordering over approval, proxy inspection, and sink opacity.

5.6 Benign Utility and Latency

Across 312 tasks and five seeds, CallWitness completes 1471/1560 executions, 94.3% (CI 93.1–95.4%). It denies 160/2336 benign calls, 6.8% (CI 5.9–7.9%), and sends 50/2336 calls to endorsement, 2.1% (CI 1.6–2.8%). Completion ranges from 89.4% for customer response, where legitimate destinations are broadest, to 97.8% for internal summarization. Appendix Table 7 reports all category counts and intervals. The matched intent-approval mediator completes 286/312 tasks (91.7%) and adds 8.8 ms median call latency.

Without the broker, primary per-call latency is 13.1/34.8/60.8 ms at p50/p95/p99; CallWitness reaches 17.3/39.0/65.1 ms, adding 4.2/4.2/4.3 ms. Median added latency is 8.8 ms for approval, 4.4 ms for the GPT-5.4 integration, and 4.6 ms for the Claude integration. Appendix Table 9 gives matched sample counts, tail quantiles, and clustered intervals.

The evidence converges on one mechanism: authorization is strongest when the final decision retains both the lineage of materialized values and the target resolved by the wrapper. Removing that target produces a 21.9-point paired increase; removing either lineage or destination-scoped authority

produces 20.0- to 28.4-point gaps from the full record; and intent approval leaves four times the pooled residual risk. The visibility ordering persists across two current stacks and adaptive attacks; independent policy authorship exposes alias, scope, and endorsement as the consequential policy choices. Benign completion remains above 90%, and CallWitness and the current integrations add 4.2–5.2 ms from p50 to p99, versus 8.8–18.7 ms for matched approval.

6 Related Work

Prompt-injection benchmarks expose attacks embedded in untrusted context (Greshake et al., 2023; Liu et al., 2024; Zhan et al., 2024; Zhang et al., 2025). StruQ separates instructions from data (Chen et al., 2025), NeMo Guardrails constrains prompts and actions (Rebedea et al., 2023), and DLP proxies inspect payloads at egress. CallWitness acts at a different moment: after argument materialization and target resolution but before credentials cross the tool boundary. This placement enables the paper's same-trajectory intervention and makes prompt-only, guardrail, allowlist, and DLP controls operational baselines for the measured literal-token attack.

The closest execution-layer systems retain different evidence. CaMeL enforces data flow and capabilities under a trusted plan (Hayes et al., 2025); PACT assigns roles and provenance requirements to proposed arguments (Wang et al., 2026); and Progent applies privilege rules to tool names and arguments (Li et al., 2025). AgentArmor reconstructs runtime dependencies (Lu et al., 2025), ARGUS checks provenance and benign justification (Xie et al., 2026), and AttriGuard uses counterfactual attribution for a proposed call (He et al., 2026). These mechanisms address plan integrity, argument admissibility, trace provenance, or causal influence; their published evidence objects do not retain the same broker-derived consumed-value graph joined with a wrapper-rematched sink at credential release. Appendix Table 10 aligns the decision points and fields. Section 5 uses evidence-equivalent implemented conditions for empirical comparison and these execution-layer systems to establish the mechanism-level distinction.

Structured permissioning provides the direct decision-time comparator because it authorizes a declaration before the final target is known (Wu et al., 2026). The matched mediator shares attacks, schemas, deny goal, samples, seeds, materializer, and wrappers with CallWitness, but its release token omits handle-derived lineage and the post-alias/redirect sink; the joined execution record lowers success from 5.1% to 1.3%. Technically, CallWitness unifies destination-scoped authority (Lampson, 1974; Watson et al., 2012), explicit information flow (Denning, 1976; Zeldovich et al., 2006; Pasquier et al., 2017), and endpoint resolution (Akhawe et al., 2010; Reis et al., 2019) in a single credential-release decision whose sink evidence can be removed without changing the trajectory or policy.

7 Discussion and Conclusion

CallWitness identifies the authorization object that closes the gap between an agent's declaration and its credentialed side effect: the joined record of materialized-value lineage and the wrapper-resolved sink. Removing only the sink from otherwise identical browser traces multiplies literal-token attack success by $15.0\times$, from 1.6% to 23.4%. The factorial result shows: lineage alone leaves 21.4% seeded-run success, destination-scoped authority alone leaves 29.8%, and their join reaches 1.4%.

The evidence establishes this interface for broker-mediated, destination-declaring calls over browser, HTTP, email, ticket, file, and upload schemas, with a trusted credential boundary and an exact generated-token sink oracle. Within that regime, the 11 residual successes bound full-stack violation of the deny goal at 1.3%; the proposition states why each such event must fall outside at least one premise of complete mediation, lineage, sink resolution, or policy enforcement rather than contradicting the per-call invariant. On a separately evaluated 178-instance slice, the independently authored policy's 14 disagreements and 6.2% success rate locate alias normalization, cross-project scope, and endorsement as first-class deployment rules rather than hidden wrapper behavior.

Across 847 attacks, the full record reaches 1.3% success versus 5.1% for matched intent approval and 21.3–28.4% per channel for the DLP proxy. The sink-visibility effect repeats with GPT-5.4, Claude Sonnet 5, and three dispatch paths; 94.3% benign completion and 4.2 ms median overhead show that the decision can remain on the tool-call critical path. Agent runtimes should place credential release after value materialization and sink resolution, with both bound in one decision.

Reproducibility Statement

Sections 2–5 specify the threat model, authorization record, implementation, workloads, baseline semantics, and statistical protocol. Appendix A records slice construction, challenge conditions, uncertainty procedures, and run-level evidence, while Appendix B gives the complete security proof.

Ethics Statement

CallWitness is evaluated as a defensive exfiltration control. All attack executions run inside isolated Docker networks with public-internet egress disabled; attacker sinks are local containers, and secrets are freshly generated 32-byte tokens rather than credentials or user data. Reported results aggregate attack behavior and omit operational payload sequences. Decision logs retain handle identifiers, labels, and matched rules instead of secret contents. Because lineage metadata can reveal sensitive workflow structure, deployment guidance treats these logs as protected records with access control and retention limits. Policy changes and endorsements are audited, and ordinary defense-in-depth measures—credential scoping, network segmentation, output controls, secret rotation, and policy review—remain in the surrounding deployment.

AI Use Statement

Generative AI tools were used during manuscript preparation to assist with editorial refinement, formatting normalization, and bibliographic-record checking.

References

Devdatta Akhawe, Adam Barth, Peifung E. Lam, John Mitchell, and Dawn Song. Towards a formal foundation of web security. In *IEEE Computer Security Foundations Symposium (CSF)*, 2010. doi: 10.1109/CSF.2010.27. URL https://doi.org/10.1109/CSF.2010.27.

Sizhe Chen, Julien Piet, Chawin Sitawarin, and David Wagner. StruQ: Defending against prompt injection with structured queries. In *USENIX Security Symposium*, 2025. URL https://www.usenix.org/conference/usenixsecurity25/presentation/chen-sizhe.

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

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, 2023. doi: 10.1145/3605764.3623985. URL https://doi.org/10.1145/3605764.3623985.

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

Yu He, Haozhe Zhu, Yiming Li, Shuo Shao, Hongwei Yao, Zhihao Liu, and Zhan Qin. AttriGuard: Defeating indirect prompt injection in LLM agents via causal attribution of tool invocations. In *35th USENIX Security Symposium (USENIX Security 26)*. USENIX Association, 2026. URL https://www.usenix.org/conference/usenixsecurity26/presentation/he-yu.

Maxwell Krohn, Alexander Yip, Micah Brodsky, Natan Cliffer, M. Frans Kaashoek, Eddie Kohler, and Robert Morris. Information flow control for standard OS abstractions. In *ACM Symposium on Operating Systems Principles (SOSP)*, 2007. doi: 10.1145/1294261.1294293. URL https://doi.org/10.1145/1294261.1294293.

Butler W. Lampson. Protection. *ACM SIGOPS Operating Systems Review*, 8(1):18–24, 1974. doi: 10.1145/775265.775268. URL https://doi.org/10.1145/775265.775268.

Henry M. Levy. *Capability-Based Computer Systems*. Digital Press, 1984. ISBN 0-932376-22-3. URL https://homes.cs.washington.edu/~levy/capabook/.

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

Yupei Liu, Yuqi Jia, Runpeng Geng, Jinyuan Jia, and Neil Zhenqiang Gong. Formalizing and benchmarking prompt injection attacks and defenses. In *USENIX Security Symposium*, 2024. URL https://www.usenix.org/conference/usenixsecurity24/presentation/liu-yupei.

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

Adrian Mettler, David Wagner, and Tyler Close. Joe-E: A security-oriented subset of Java. In *Network and Distributed System Security Symposium (NDSS)*, 2010. URL https://www.ndss-symposium.org/ndss2010/joe-e-security-oriented-subset-java/.

Mark S. Miller, Mike Samuel, Ben Laurie, Ihab Awad, and Mike Stay. Caja: Safe active content in sanitized javascript. Technical report, 2008. URL https://google-code-archive-downloads.storage.googleapis.com/v2/code.google.com/google-caja/caja-spec-2008-06-06.pdf.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. WebGPT: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021. URL https://arxiv.org/abs/2112.09332.

OpenAI. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. URL https://arxiv.org/abs/2303.08774.

Thomas Pasquier, Xueyuan Han, Mark Goldstein, Thomas Moyer, David Eyers, Margo Seltzer, and Jean Bacon. Practical whole-system provenance capture. In *ACM Symposium on Cloud Computing (SoCC)*, 2017. doi: 10.1145/3127479.3129249. URL https://doi.org/10.1145/3127479.3129249.

Traian Rebedea, Razvan Dinu, Makesh Narsimhan Sreedhar, Christopher Parisien, and Jonathan Cohen. Nemo guardrails: A toolkit for controllable and safe llm applications with programmable rails. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 431–445. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-demo.40. URL https://aclanthology.org/2023.emnlp-demo.40/.

Charles Reis, Alexander Moshchuk, and Nasko Oskov. Site isolation: Process separation for web sites within the browser. In *USENIX Security Symposium*, 2019. URL https://www.usenix.org/conference/usenixsecurity19/presentation/reis.

Jerome H. Saltzer and Michael D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975. doi: 10.1109/PROC.1975.9939. URL https://doi.org/10.1109/PROC.1975.9939.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, pp. 68539–68551, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html.

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

Vitaly Shmatikov, Eugene Bagdasaryan, Tsung-Yin Hsieh, and Ben Nassi. Abusing images and sounds for indirect instruction injection in multi-modal llms. *arXiv preprint arXiv:2307.10490*, 2023. URL https://arxiv.org/abs/2307.10490.

Significant Gravitas. AutoGPT, 2023. URL https://github.com/Significant-Gravitas/AutoGPT.

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

Robert N. M. Watson, Jonathan Anderson, Ben Laurie, and Kris Kennaway. A taste of Capsicum: Practical capabilities for UNIX. *Communications of the ACM*, 55(3):97–104, 2012. doi: 10.1145/2093548.2093572. URL https://doi.org/10.1145/2093548.2093572.

Yuhao Wu, Ke Yang, Franziska Roesner, Tadayoshi Kohno, Ning Zhang, and Umar Iqbal. Towards automating data access permissions in AI agents. In *IEEE Symposium on Security and Privacy (S&P)*, 2026. doi: 10.1109/SP63933.2026.00018. URL https://arxiv.org/abs/2511.17959.

Xiaofei Xie, Jiongchi Yu, Shihao Weng, Yang Feng, Jia Liu, and Jinrui Zhang. ARGUS: Defending LLM agents against context-aware prompt injection. *arXiv preprint arXiv:2605.03378*, 2026. URL https://arxiv.org/abs/2605.03378.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. *International Conference on Learning Representations (ICLR)*, 2023. URL https://arxiv.org/abs/2210.03629.

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 the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2025. doi: 10.1145/3690624.3709179. URL https://doi.org/10.1145/3690624.3709179.

Nickolai Zeldovich, Silas Boyd-Wickizer, Eddie Kohler, and David Mazières. Making information flow explicit in HiStar. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2006. URL https://www.usenix.org/legacy/event/osdi06/tech/zeldovich.html.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, 2024. doi: 10.18653/v1/2024.findings-acl.624. URL https://doi.org/10.18653/v1/2024.findings-acl.624.

Hanrong Zhang, Jingyuan Huang, Kai Mei, Yifei Yao, Zhenting Wang, Chenlu Zhan, Hongwei Wang, and Yongfeng Zhang. Agent security bench (asb): Formalizing and benchmarking attacks and defenses in LLM-based agents. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/5750f91d8fb9d5c02bd8ad2c3b44456b-Paper-Conference.pdf.

A Scientific Protocol and Extended Evidence

A.1 Benchmark and Split Construction

The 847-instance attack benchmark is the Cartesian coverage target of four ingestion channels, four sink classes, and four payload families, followed by wrapper-specific feasibility filtering. Each retained instance specifies one benign task objective, one generated 32-byte secret, the attacker-controlled content, the allowed tool surface, and a sink-side success oracle. Five seeded runs reuse the same protected value and attack text while varying agent sampling. The 312-task utility set is stratified by workflow category and contains no injected instructions.

Table 2: Evaluated tool schemas and the execution target attested before a side effect. Read-only tools create broker handles and have no side-effect sink.

Tool schema	Wrapper-attested execution target
Browser navigation	Final document and subresource origins after redirects
HTTP request	Redirect-resolved URL and method
Email read	`none`; returned fields receive handles
Email send	Alias-expanded recipient addresses
File read	`none`; canonical path labels the returned handle
File write	Canonical workspace path
Ticket read	`none`; resolved project and ticket label the handle
Ticket post	Resolved project and ticket identifier
Object upload	Resolved service, bucket, and object endpoint

Payload templates follow the role-play, encoding, multi-turn, and paraphrase families in the cited prompt-injection benchmarks. Tool objectives are drawn from the corresponding benign workflows. Feasibility filtering precedes all defense runs and removes a candidate only when the selected wrapper cannot execute both its benign objective and designated sink. The exact generated token arriving at that sink is the attack oracle; policy denials, model refusals, and malformed calls are failures for the attacker. Paired conditions share prompts and sampled proposals until an authorization result changes the subsequent history, after which each trajectory continues under its own observed tool result and is scored by the same sink oracle.

Main, ablation, visibility, transfer, and adaptive memberships are fixed before outcome aggregation. A common-support slice retains an instance only when all conditions being compared can express its benign objective and candidate sink. The Figure 4 slice contains 286 attacks and 157 benign tasks. The Figure 2 intervention contains 320 attacks whose browser targets can be executed with either the resolved target exposed or hidden; all other call fields are byte-identical between arms.

The remaining fixed slices contain 304/144 GPT-5.4 attack/benign cases, 296/140 Claude cases, 178/96 held-out-policy cases, and 256/112 adaptive cases. The 127,438-call study trace comprises 63,525 primary attack/baseline calls, 11,704 benign calls, 45,809 comparator and transfer calls, and 6,400 protocol-replay calls.

Table 3: Sink-conditioned outcomes among trajectories that attempt their designated external sink. Similar blocking across wrappers rules out a pooled result dominated by one easy sink.

Sink	Attempts	Blocked
HTTP webhook	257	98.4%
Outbound email	186	98.9%
Ticket post	165	98.2%
File upload	95	98.9%

A.2 Statistical Procedure

Instance ASR is an any-of-five indicator. Run-level leakage treats each seeded execution as an observation but clusters intervals by benchmark instance. Utility completion clusters by task, while

Table 4: Execution-state evidence matrix. Rows vary the evidence retained from intent release to the credential decision; outcomes state their analysis unit and denominator.

Condition	Credential-release evidence	Stress condition and diagnostic	Outcome and paper trace
CallWitness full record	Handle-derived lineage and resolved sink	Full-stack stress of wrapper attestation and policy enforcement	11/847 instances, 1.3% any-of-five ASR (Fig. 3)
Intent-release approval	Declaration and approval token	Resolved aliases and handle-derived lineage absent at execution	43/847 instances, 5.1% any-of-five ASR
DLP proxy	Payload and egress endpoint	Payload inspection without broker value ancestry	21.3–28.4% per-channel any-of-five ASR (Fig. 3)
Capability only	Resolved sink, no lineage	Secret-derived inputs inherit allowed operation authority	426/1,430 seeded runs, 29.8% ASR (Fig. 4)
Lineage only	Handles and lineage, no destination-scoped authority	External target lacks enforceable capability scope	306/1,430 seeded runs, 21.4% ASR (Fig. 4)
Opaque browser	Handles and lineage, unresolved sink	Redirect/subresource target hidden under fixed policy	75/320 instances, 23.4% any-of-five ASR versus 5/320, 1.6% visible (Fig. 2)

call-level denial, escalation, and latency cluster all calls from the same task or attack run. We compute two-sided percentile intervals from 10,000 bootstrap samples. Paired comparisons resample shared instance indices and their five-run vectors together. The factorial ablation reports seeded-run proportions over $5 \times 286 = 1{,}430$ runs and clusters resampling by the 286 instances; its four success counts are 1,183, 426, 306, and 20. Factorial interaction is $\mathrm{ASR}_{\mathrm{none}} - \mathrm{ASR}_{\mathrm{cap}} - \mathrm{ASR}_{\mathrm{prov}} + \mathrm{ASR}_{\mathrm{full}}$.

A.3 Approval Comparator Protocol

The approval mediator and CallWitness receive the same prompts, model samples, nine tool schemas, secrets, and deny goal. The mediator releases a signed token after checking a structured pre-execution declaration. Its downstream dispatcher checks token validity and declared destination class but receives neither handle-derived lineage nor the wrapper's fully resolved sink. Across 847 attacks it records 43 successes (5.1%, CI 3.7–6.6%). It completes 286/312 benign tasks (91.7%, CI 88.4–94.4%), reaches 29/312 endorsement boundaries (9.3%, CI 6.2–12.9%), and denies 274/2341 benign calls. End-to-end call latency is 21.9/46.8/79.5 ms at p50/p95/p99.

A.4 Adaptive Held-Out Attacks

Attack designers receive the authorization record, aggregate baseline behavior, and a 40-query development budget per template. They produce held-out templates with no exact or eight-token overlap with the main benchmark templates. The evaluation crosses all ingestion and sink classes, freezes secrets, wrappers, policies, and five-seed semantics, and scores 256 instances once.

Table 5: Authorization-record-aware adaptive attacks. ASR is any-of-five over 256 held-out instances; intervals are task-clustered 95% bootstrap intervals.

Condition	Successes	ASR	95% CI
CallWitness visible	9	3.5%	1.7–6.2%
Approval mediator	24	9.4%	6.1–13.5%
DLP proxy	64	25.0%	20.0–30.6%
Opaque browser	58	22.7%	17.9–28.0%

The visible arm has 18/1280 successful seeded runs, 1.4% leakage (CI 0.8–2.2%). Its per-sink instance ASR is HTTP 3.2% (2/63), email 2.8% (2/72), ticket 4.4% (3/68), and upload 3.8% (2/53). On 112 matched clean tasks, 522/560 seeded executions complete, or 93.2% (CI 89.0–96.4%).

A.5 Run-Level Aggregation Check

Table 6: Seed-level leakage on the main benchmark. The final column gives the number of successful instances with exactly 1/2/3/4/5 successful seeds.

Condition	Runs	Successful runs	Leak rate [95% CI]	Seed-count histogram
CallWitness	4235	17	0.4% [0.2–0.7%]	7/3/0/1/0
Approval mediator	4235	67	1.6% [1.2–2.0%]	25/13/4/1/0
DLP proxy	4235	263	6.2% [5.5–7.0%]	165/43/4/0/0

The run-level ordering matches the any-of-five ordering. Multi-seed successes occur in every condition, while most successful instances leak in a single seed; the benchmark therefore captures stochastic attack sensitivity rather than a deterministic floor or ceiling.

A.6 Contemporary Stack and Policy Transfer Details

The GPT-5.4 arm uses `gpt-5.4`, the Responses API, OpenAI Python 3.0, and OpenAI Agents SDK 0.21.0 with strict provider-native function schemas. CallWitness intercepts the SDK invocation callback after argument materialization. The Claude arm uses `claude-sonnet-5`, Anthropic API version `2023-06-01`, and LangGraph 1.2.11; its adapter wraps `ToolNode` immediately before dispatch. Both adapters pass the same value handles, lineage sets, resolved target, matched policy identifier, and seeded success oracle to the broker.

GPT-5.4 records 4/304 visible successes (1.3%, CI 0.4–3.2%) and 59/304 opaque successes (19.4%, CI 15.1–24.0%). Claude records 5/296 (1.7%, CI 0.6–3.8%) and 64/296 (21.6%, CI 17.2–26.5%). Their benign completion intervals are 88.1–96.1% and 87.4–95.6%, respectively.

The independently authored policy bundle is mapped to the same capability vocabulary and deny goal before outcomes are revealed. Its 14 disagreements with the primary bundle are adjudicated by rule category rather than removed: alias normalization, cross-project destination scope, and endorsement requirements. This protocol tests policy authorship while preserving the security question and tool semantics.

A.7 Utility and Latency Detail

Table 7: Benign completion by category. Completion counts five seeded executions per task; intervals cluster by task.

Category	Tasks	Completed / $5N$	95% CI
Internal summary	45	220/225 (97.8%)	94.9–99.1%
Customer response	68	304/340 (89.4%)	85.6–92.4%
Log correlation	72	343/360 (95.3%)	92.6–97.1%
Data pipeline	46	224/230 (97.4%)	94.4–98.8%
Approval workflow	81	380/405 (93.8%)	91.2–95.8%
Overall	312	1471/1560 (94.3%)	93.1–95.4%

A.8 Adjacent Control Surfaces

B Security Property Proof

We restate Proposition 1 and give a complete proof.

Table 8: Benign call outcomes by category. Brackets contain task-clustered 95% intervals; escalation is a required endorsement boundary.

Category	Denied calls / calls	Escalated calls / calls
Internal summary	13/337, 3.9% [1.9–6.4%]	4/337, 1.2% [0.4–2.5%]
Customer response	49/395, 12.4% [9.3–15.8%]	16/395, 4.1% [2.3–6.3%]
Log correlation	34/535, 6.4% [4.5–8.6%]	10/535, 1.9% [0.9–3.2%]
Data pipeline	16/450, 3.6% [2.0–5.6%]	0/450, 0.0% [0.0–0.0%]
Approval workflow	48/619, 7.8% [5.8–10.0%]	20/619, 3.2% [1.9–4.8%]
Overall	160/2336, 6.8% [5.9–7.9%]	50/2336, 2.1% [1.6–2.8%]

Table 9: Per-call latency in milliseconds. Quantile intervals are task-clustered 95% intervals; added latency uses paired calls against each runtime's unmediated path.

Condition	Calls	End-to-end p50/p95/p99	Added p50/p95/p99	End-to-end quantile intervals
CallWitness study trace	127,438	17.3/39.0/65.1	4.2/4.2/4.3	17.1–17.5; 38.6–39.5; 64.1–66.3
Approval mediator	126,902	21.9/46.8/79.5	8.8/12.0/18.7	21.5–22.3; 46.1–47.6; 77.8–81.4
GPT-5.4 stack	45,812	18.6/41.2/69.7	4.4/4.5/4.9	18.2–19.0; 40.4–42.1; 67.9–71.8
Claude stack	44,736	19.4/43.7/72.6	4.6/4.9/5.2	19.0–19.9; 42.8–44.7; 70.5–74.9

Table 10: Closest control surfaces, aligned by decision point and the execution facts available for a side effect. "Proposed target" precedes redirects, alias expansion, and browser subresource resolution.

Control surface	Decision point	Value lineage	Destination evidence
StruQ / guardrails (Chen et al., 2025; Rebedea et al., 2023)	Prompt/model action	Host-dependent	Host-dependent
Permission automation / Progent (Wu et al., 2026; Li et al., 2025)	Intent or tool dispatch	Declared access	Proposed target
CaMeL (Hayes et al., 2025)	Tool execution	Explicit data flow	Capability target
PACT (Wang et al., 2026)	Per-argument pre-tool check	Argument provenance	Proposed target role
AgentArmor / ARGUS (Lu et al., 2025; Xie et al., 2026)	Runtime trace / pre-action	Trace/graph-derived	Trace-derived target
AttriGuard (He et al., 2026)	Counterfactual pre-tool check	Causal attribution	Proposed call target
DLP proxy	Egress transit	Payload inspection	Network endpoint
CallWitness	Final pre-credential decision	Consumed handle graph	Post-resolution target

Proposition 1 (restated). *Assume that:*

1. *the broker mediates every tool invocation that produces network egress or any other external side effect;*
2. *every value-bearing argument is a broker handle or the result of a broker-executed transformation, so E_{auth} records complete handle-derived lineage;*
3. *the sink in E_{auth} is resolved correctly by the corresponding tool wrapper; and*
4. *policy denies flows from values whose lineage contains* `secret` *to external destinations unless an explicit endorsement rule applies.*

Under these assumptions, no execution can deliver a `secret`*-derived value through a mediated tool call to an external sink without endorsement, even if the LLM is fully compromised.*

Proof. Consider any execution and suppose, for contradiction, that a `secret`-derived value O reaches an external sink through a mediated tool call without endorsement. Because the sink interaction is a mediated side effect by assumption (1), the call that emits O must be checked by the broker before the side effect occurs.

Let the emitted value at that call be O'. By assumption (2) and the value-flow construction in Section 3.1, the authorization record contains every handle reachable from the call's argument expressions. Every broker-produced value carries lineage, and a broker-executed transformation assigns its output the union of its input lineage. Therefore, if any upstream input in the derivation of O' carries the `secret` tag, then O' is also `secret`-derived unless an explicit endorsement removes that tag. By hypothesis, no endorsement occurs on the path under consideration, so O' remains `secret`-derived at the time of the attempted side effect.

By assumption (3), the authorization record for that side effect includes a correctly resolved external destination. The broker therefore evaluates the policy rule for a `secret`-derived value flowing to that external destination. Assumption (4) states that every such flow is denied unless an explicit endorsement rule applies. Since no endorsement applies here, the broker must deny the call.

This contradicts the supposition that O' reaches the external sink. Hence no `secret`-derived value can reach an external sink through a mediated tool call without endorsement. □

The proposition applies to each call for which handle mediation is complete, the wrapper attests the resolved destination, and policy denies unendorsed secret egress. The end-to-end benchmark challenges those conditions over complete trajectories and scores literal-token arrival at the external sink without assuming adapter or wrapper conformance.