Composition Collapse: How Multi-Agent Orchestration Frameworks Lose the Initiating Principal's Authority

Anonymous Submission

Abstract

Multi-agent orchestration frameworks — CrewAI, LangGraph, AutoGen — let one agent hand off work to another. We show this composition silently breaks a security property each agent satisfies in isolation: an action should execute with no more authority than the principal who *initiated* it. When a request crosses a composition boundary, the initiator's authority is not carried in any form a downstream protected resource can use, so the resource's native access control (RBAC, a filesystem ACL, an OAuth scope) is enforced correctly — but against the *deputy*, not the initiator. We call this **principal substitution**; the underlying low-privilege-request → handoff → high-privilege-operation structure now ships by default in enterprise platforms such as ServiceNow Now Assist.

Frameworks have already tried to fix this, shipping identity primitives — AutoGen's `source`, CrewAI's `fingerprint`, LangGraph's `config.configurable`, A2A's transport identity — and not one works: carrying an initiator across composition requires a channel that is jointly *originating*, *unforgeable*, *decision-reachable*, and *attenuating*, and each secures only a *different* proper subset, so an identity field does not predict safety. We call this **false assurance** — a failure mode the 1988 confused deputy lacks, because there no fix had been attempted. We decide each property as a runtime verdict with a deterministic differential that removes the LLM confounder, against live authorization backends plus executed positive controls, and confirm a *pre-registered* verdict on a previously-unseen fourth framework. We package the checks as a framework-agnostic conformance test, and propose no new delegation protocol.

1. Introduction

1.1 A motivating incident

In November 2025, AppOmni disclosed a second-order prompt-injection chain in ServiceNow's Now Assist agents~[3, 58]. A low-privilege Workflow Triage agent ingests an attacker-influenced request and, through Now Assist's default agent-to-agent discovery, hands a derived task to a higher-privilege Data Retrieval agent, which executes it — exfiltrating sensitive records — even with ServiceNow's built-in prompt-injection protection enabled. Two details make this the right starting point. First, the data-retrieval agent's actions were *faithfully authorized*: it ran with its own legitimate privileges, guardrails not bypassed, yet the operation originated from a principal that should never have been able to request it. Second, this was not treated as a bug to patch: ServiceNow confirmed the behavior was *intended* and updated its documentation rather than shipping a fix, the disclosing researcher calling the chain "expected behavior as defined by certain default configuration options" rather than "a bug in the AI"~[3, 58] — agents are discoverable by default and auto-teamed when published. The vulnerability lives in how the *framework* composes independently reasonable agents, not in any single agent's logic.

This is not confined to one product: the same low-privilege-request → cross-agent-handoff → high-privilege-operation structure now ships *by default* in mainstream enterprise platforms (Microsoft Copilot Studio agent nodes, Salesforce Agentforce topic delegation), so the gap is activated by default as adoption grows (we return to deployment breadth and the vendor-stated invariant in §2.3 and §7.3). The question is not whether the structure is deployed, but whether the channel that would make it safe is — and across the frameworks we examined, it is not.

1.2 Adding an identity field does not close the gap

This incident is a symptom of a structural property of multi-agent orchestration~[16, 51, 62, 64, 67]. When agent A delegates to agent B, the request crosses a framework boundary (a handoff, delegation call, shared-state update, or serialized message), and the *authority* under which the work was requested must travel with it for a downstream protected resource to enforce access control against the right principal (Figure~1). Mainstream frameworks carry no such channel: the initiating

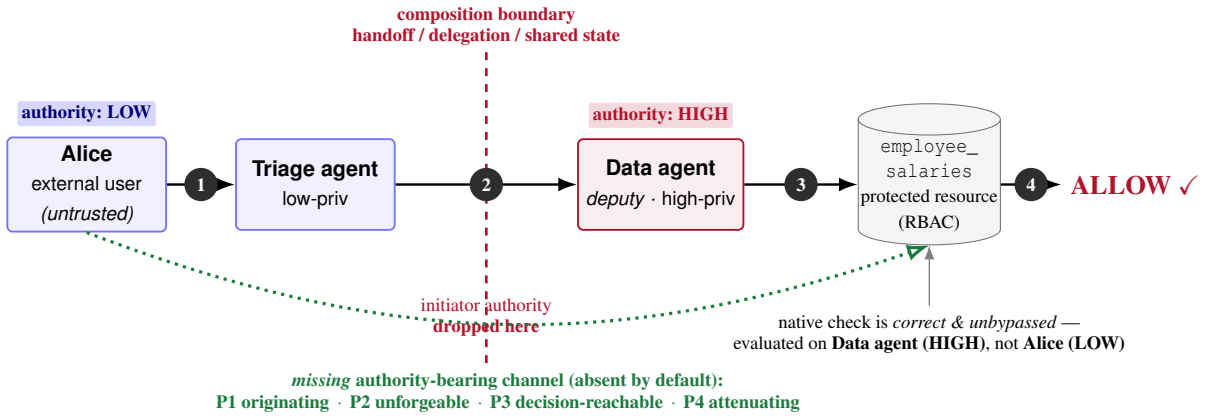


Figure 1: Principal substitution at the composition boundary (running example: **Alice**, a low-privilege user, ultimately drives a read of the `employee_salaries` table, the protected resource; in the wild, the ServiceNow Now Assist incident of §1.1). *What happens:* ❶ Alice's request enters at low authority; ❷ it crosses a handoff into the high-privilege *deputy*, and the initiator's authority is dropped at the boundary; ❸ the deputy calls the resource under its own credential; ❹ the resource's native access control (RBAC/ACL/OAuth) is correct and unbypassed, but evaluated against the *deputy* (**HIGH**), not Alice (**LOW**) — so the privileged read executes. *What is missing (dotted):* no native channel carries Alice's authority to the decision with the four properties needed — originating (P1), unforgeable (P2), decision-reachable (P3), attenuating (P4).

principal is degraded into natural-language text or an advisory identifier that downstream authorization does not consume, and no *native* channel supplies it in the runtime-set, integrity-protected, originating form safe consumption needs. The consequence is principal substitution: the protected resource's *native* access control (a data warehouse's row-level RBAC, a filesystem ACL, an OAuth scope) is enforced, correctly and unbypassed, but against the **deputy's** principal rather than the **initiator's**. An agent that is provably safe in isolation becomes unsafe purely by being composed.

A natural objection is that this merely restates the classical confused deputy~ [18] and is therefore obvious and long-solved. It is neither — and the sharpest evidence is that **frameworks have already tried to fix it, and the fixes do not hold.** Mainstream orchestrators now ship identity primitives aimed squarely at this concern: AutoGen stamps every message with a `source` field naming its sender, CrewAI assigns each component an immutable `fingerprint`, and Google's A2A protocol establishes a caller identity at the transport layer. Yet, as we show (§4), *each of these closes a different part of the problem and leaves principal substitution intact.* The reason is structural: carrying an initiator's authority across composition requires a channel that is **simultaneously** originating, unforgeable, decision-reachable, and attenuating (the four-property yardstick of §2.1), and each shipped primitive supplies only a *different* proper subset, none reaching the conjunction (the per-primitive verdicts are the failed-fix matrix of Table~4, our central result). **The field exists; the assurance it suggests does not.** A partial channel may be worse than none, because it can invite *false assurance*: a developer who sees an identity field may reasonably assume the principal is carried, while the missing property silently defeats it. The 1988 confused deputy could not exhibit false assurance — there was no identity field to trust; multi-agent frameworks are the first systems to ship that field *by default*, which is exactly why this is a new failure mode and not a restatement of an old one.

Official material reinforces this: CrewAI's tool-hook docs gate a sensitive tool on the *executing* agent's role, not the originator~ [8], and AutoGen's docs treat the speaker's `name` as "who actually sent the message"~ [32] — sound for the adjacent hop, unsound once that sender is itself a deputy. Practitioners notice the same: an identity that is "a string (role name)" with no "cryptographic proof"~ [7], and a runtime that does not "enforce that the source matches the actual sender"~ [33].

1.3 Why this is hard to see, and how we isolate it

Such failures are easy to mistake for prompt injection or model stochasticity, since deployed agent failures often involve injected instructions, untrusted documents, or probabilistic tool use~ [10,15,41,43,68]. We isolate the framework-level failure from these confounders with a deterministic differential (§3): a benign faithful relay makes a composed-only failure unattributable to model or payload, and a forgeability differential pins the cause to the missing channel (P2).

1.4 Contributions

The delta in one line: prior work *asserts* an authority-bearing channel *should* exist; we *measure*, at the implementation layer, that it does not, by default. We are accordingly explicit about what is *not* novel: the confused-deputy pattern is classical~ [18]; prior work has *observed* multi-agent confused-deputy

behavior (SEAgent~ [19]) and *named* the trust–authorization gap (SoK~ [52]). Our contributions are:

1. **A failed-fix natural experiment** (§5) — *our central result*: four identity primitives frameworks shipped to fix exactly this (AutoGen's `source`, CrewAI's `fingerprint`, LangGraph's `config.configurable`, A2A's transport identity) each secure a *different* proper subset of {P1–P4}, and not one reaches the conjunction — none attenuates (P4), and even the strongest is never consulted at the decision (P3) — so the existence of a shipped identity field does not predict safety (**false assurance**).
2. **The four-property yardstick that makes that verdict decidable** (§2.1): originating, unforgeable, decision-reachable, attenuating — the conditions a channel must satisfy *jointly*, each independently necessary and witnessed by a shipped primitive that secures the other three. We use it not as a taxonomy but as the instrument that turns "this fix fails" into "this fix fails because it misses exactly P_k," grounding sufficiency in executed positive controls rather than a proof.
3. **A deterministic differential method** (§3) that removes the LLM confounder: under a dutiful relay (a deterministic stand-in for each agent that simply forwards the task, with no model and no attack capability) and benign inputs, a forgeability differential decides P2 and a multi-hop differential decides P1 at runtime, turning each property into a verdict rather than an observed ALLOW/DENY. Because the account is about a *class* of primitives, we also pre-registered (in version control) the verdicts for a previously-unseen fourth framework — the OpenAI Agents SDK — and confirmed every cell, a guard against post-hoc fitting (§4).
4. **A framework-agnostic conformance test** (§7.1), the actionable artifact: a runnable diagnostic reporting which of P1–P4 each native channel satisfies, plus the minimal conformant out-of-band channel as root-cause validation. We do **not** propose a new delegation protocol.

Two supporting checks round out the case: three production models take the framework-permitted unsafe path in 89 of 90 *benign* trials (§6.1), and a census of 13 frameworks finds no authority-bearing initiator field at any audited cross-handoff site (§6.2). Both are subordinate to the differential, which carries the root-cause attribution.

Roadmap. The argument runs: invariant (†) and the four-property yardstick (§2.1) → a differential that decides each property (§3) → the (†) violation reproduces, is attributed, and is confirmed out-of-sample on a held-out framework (§4) → *the result*: every shipped identity fix secures only a proper subset, so the fix fails (§5) → supporting breadth checks (§6) → a runnable conformance test (§7.1).

2. Model and Threat Model

2.1 When does composition preserve authority? A four-property account

We begin from the invariant a composed workflow must preserve and derive the conditions any solution must meet. Write $\mathrm{init}(r)$ for the principal that *initiated* request r and $\mathrm{exec}(r)$ for the principal that ultimately *executes* it at a protected resource. The least-privilege invariant is

$$\forall r: \quad \mathrm{effective\_authority}(\mathrm{exec}(r)) \leq \mathrm{authority}(\mathrm{init}(r)). \qquad (\dagger)$$

Routing a request through intermediaries must never *amplify* the authority under which it acts. **Principal substitution** is exactly a violation of (†): the protected resource's native check passes, but it is evaluated against the deputy $\mathrm{exec}(r)$ rather than the initiator $\mathrm{init}(r)$, so an action the initiator could not perform directly succeeds once composed. Two scoping conventions make (†) precise. First, $\mathrm{init}(r)$ is fixed at the workflow's trust boundary — the external party that introduced r in our threat model — and is *not* redefined by intermediate deputies, so it is well-defined across a multi-hop chain. Second, the order $\leq$ and the operator min are taken within a *single resource's* authorization domain (one warehouse's RBAC, one filesystem's ACL, one API's OAuth scopes), over which authority forms a partial order with a well-defined min for the (deputy, init) pairs we attenuate (not necessarily a full lattice — POSIX uids and RBAC roles need not be totally ordered); we do not assume an order across heterogeneous resources, and each result in §4 is stated against one protected resource. Unifying authority across them is out of scope.

To maintain (†) across a composition boundary, *some channel* must carry the initiating principal to the point of enforcement in a usable form. Such a channel must satisfy four properties **jointly**:

- **(P1) Originating.** It carries the *initiating* principal, not merely the adjacent peer of the last hop. A channel re-established at each hop authenticates the immediate sender, not the originator, and is laundered by multi-hop delegation.
- **(P2) Unforgeable.** Its value is set by the trusted runtime and is integrity-protected — not derived from attacker-influenceable content. A claim the adversary can state as text is an *ambient claim*, not a capability.
- **(P3) Decision-reachable.** It reaches, and is consultable by, the authorization decision at the protected resource — not confined to a logging, tracing, or audit path. This is necessary but not sufficient: a decision-reachable channel can still be forgeable (P2) — so a defender who *can* read a value has not thereby been handed a trustworthy one.

- **(P4) Attenuating.** Authorization is computed as $\min(\text{authority}(\text{deputy}), \text{authority}(\text{init}))$ per operation, so that carrying the initiator actually *constrains* the action.

P1 and P2 are properties of the *channel* (origination, integrity); P3 and P4 are the *consumer-side* rules that make the channel load-bearing — the decision must consult it, and consult it under attenuation. They are individually familiar from capability security and protection systems~ [12,23,26,34,35]; our contribution is empirical and turns on their **conjunction**. A channel satisfying any *proper subset* does not maintain (†), and a partial channel may be worse than none — the false assurance of §1.2.

Each property is necessary, and each failure mode is realized by a primitive that ships today. The four are not a wish-list we satisfy by construction: drop any one and a concrete bypass remains, witnessed by a shipped channel that secures the other three — A2A's transport identity and AutoGen's re-stamped `source` drop P1 (multi-hop laundering), an in-band origin claim drops P2 (forgeable text), CrewAI's `fingerprint` drops P3 (audit-only), and LangGraph's `config.configurable` drops P4 (never attenuated). These are the failed-fix matrix of §5, decided at runtime for P1/P2 (§4.4). Each property is thus independently load-bearing. *Sufficiency* we ground not in a proof but in the executed positive controls below, making P1–P4 an *empirically validated* target, not an impossibility theorem — a channel missing any one admits substitution in our tests.

Positive control: P1–P4 are a projection of deployed authority-preserving designs, not properties we invented. They are the standard requirements mature, *non-agentic* delegation mechanisms already enforce: OAuth 2.0 token exchange / on-behalf-of preserves the original subject across hops (P1), runtime-issued (P2), the resource server enforcing the narrowed scope at the decision (P3, P4)~ [17, 20]; macaroons and SPKI make attenuation first-class rather than advisory strings~ [5, 13]; and a database `SECURITY INVOKER` routine evaluates authorization against the *calling* principal ($\text{exec}(r) = \text{init}(r)$). Each satisfies the full conjunction in its setting. Two we do not merely cite but *run* through the same quadruple harness: an RFC 8693 token exchange (real RS256 signatures) and a live PostgreSQL 16 `SECURITY INVOKER` routine each `DENY` the composed low-privilege cell while preserving the high-privilege one, with `SECURITY DEFINER` re-exhibiting the collapse on the real engine as a negative contrast (`poc/poc_positive_control_rfc8693.py`, `poc/poc_positive_control_pg_invoker.py`). Our out-of-band channel (§7.2) is the minimal projection of these onto the agent boundary — evidence the conjunction is not merely *realizable* but *executed*, not an artifact of our framing.

This account yields a falsifiable, counter-intuitive prediction — *the existence of an identity field does not predict safety* — which §5 confirms across every shipped primitive.

2.2 Threat model

Setting. A multi-agent workflow built on a mainstream orchestration framework. Agents delegate, hand off, or share state. We use *protected resource* for the downstream interface that performs the sensitive operation and enforces the deployment's native authorization policy — a database relation under RBAC/RLS, a file under an ACL, or an API operation under OAuth scopes; at least one such resource enforces this check on the principal that *executes* the call. This is platform-native authorization that production deployments rely on, grounded in standard access-control models such as RBAC and access-control logic~ [2, 50]; we add no guard of our own. (In the formal frame of §C we write *sink*/sink_allows as shorthand for this same enforcement point; in §8, *sink* keeps its standard taint-analysis meaning.)

Adversary. The attacker controls only the **untrusted external input** that flows into the workflow: the end-user prompt and any external data a tool returns, as in retrieval-augmented and tool-integrated LLM applications~ [27]. The attacker can therefore state arbitrary claims *as text*, including false claims about who is requesting an action.

Trust assumptions (the strong version). Every agent is **benign** — none poisoned, malicious, or running an injected prompt. We deliberately exclude the weak model with a malicious agent, which reduces to ordinary prompt injection. This makes the result **stronger**: the gap survives the framework's most favorable conditions — no injection to filter, no misaligned agent to detect — so a deployment that fully solved prompt injection and alignment would *still* exhibit it, and a malicious agent or payload only makes it worse: injection is the *trigger*, the framework's composition the *amplifier* we isolate (§6.3). The failure is a floor on the framework's behavior, not a worst case we engineered.

Security goal. The invariant (†) of §2.1: a request must execute with no more authority than its initiator possessed. The violation we study — principal substitution — is the case where the protected resource's native check passes correctly but is enforced against the deputy rather than the initiator.

Out of scope. Model jailbreaks, vulnerabilities in the protected resource's own access-control implementation, transport/network security, and supply-chain compromise of the framework. We assume the resource's access control is sound; the failure is upstream of it, at the framework's composition boundary.

2.3 Why (†) is a security property, not a configuration choice

A vendor may object — as ServiceNow did, confirming the Now Assist behavior was *intended* rather than a flaw to patch~ [58] — that if agents are configured to discover and delegate to one another, a high-privilege agent acting on a low-privilege request is simply working as designed. This

conflates a *functional* default with a *security* argument. The principle of least privilege~ [49] requires every operation to execute with no more authority than the principal on whose behalf it acts; an operation that *acquires* authority by being routed through a deputy is privilege amplification — precisely the failure least privilege exists to prevent. "By design" describes how the mechanism behaves; it does not establish that the behavior is safe. That this is the right security target, and not our idiosyncratic framing, is something other vendors state outright: Microsoft's Copilot Studio guidance requires that "an agent cannot indirectly obtain information or trigger actions through another agent that it could not access directly" — which is (†) in plain words. The disagreement is therefore not over whether (†) ought to hold but over whether it does; our results (§4) show that, across the frameworks these platforms are built on, no shipped channel enforces it.

Abandoning (†) is not local: if the deputy's authority governs, any initiator who can reach a deputy inherits it, and the effective access-control lattice collapses to the union of all reachable agents' privileges. (†) is the minimal condition preventing this collapse — exactly what each agent's native RBAC/ACL/OAuth enforces *within* an agent and composition silently drops across the boundary. That ServiceNow's prompt-injection protection was enabled yet did not prevent the exfiltration confirms the failure is orthogonal to injection: by-design $\neq$ by-security-design.

3. Method: Deterministic Differential Testing

Our method is an instrument for deciding the four properties of §2.1, turning each into a runtime verdict rather than an observed ALLOW/DENY. We give the intuition first — the dutiful relay (§3.1) removes the model as a confounder; the four-tuple (§3.2) exhibits the (†) violation; the forgeability differential (§3.3) decides unforgeability (P2), with a multi-hop variant deciding origination (P1) (§4.4); and the ablations (§3.4) test that restoring decision-reachability (P3) or attenuation (P4) each suffices, given a channel satisfying P1–P2. A controlled-experiment truth table and four propositions formalize the frame in Appendix C (§C); we decide each property at the strongest evidence it admits — *executed* or *runtime* where a differential can run, *structural* where the failure is a non-read — per cell in Table~9.

3.1 The dutiful relay (removing the model confounder)

We replace each agent's LLM with a **deterministic, dutiful relay**: a rule-based stand-in that forwards the task and emits a fixed downstream action *regardless of who originated the request*~ [39, 40, 60]. Critically, it *faithfully transcribes whatever origin the request states, true or forged* — a benign relay cannot distinguish an injected origin claim from a real one; both arrive as text. With zero attack capability, any composed-only failure originates in the framework, not the model. The relay is a controlled witness, not a behavioral bound: it shows the framework *permits* the substitution along a benign, no-payload path; the in-vivo check (§6.1) then shows real models take it.

3.2 The four-tuple (reproducing the phenomenon)

For each framework we run four configurations against the *same* sensitive operation (Figure~6):

(i) `isolated-low` — low-priv principal → protected resource directly; expected `DENY` (local policy correct). *(ii)* `isolated-high-benign` — high-priv user → high-priv agent → protected resource; `ALLOW` (the deputy is legitimate *in isolation*). *(iii)* **`composed`** — low-priv request → delegation/handoff/broadcast → high-priv agent → protected resource; **`ALLOW`** (principal substitution). *(iv)* `patched-composed` — composed + initiator authority retained & attenuated; `DENY` (root-cause repair).

The outcome is read at the **protected resource** (did the privileged operation execute?), not from whether an agent raised — a relay can swallow an error and still answer. The `isolated-low` vs `composed` contrast is the classical confused deputy (an entailment, Prop 1 of §C); the novel weight is in the forgeability differential (§3.3), which attributes the collapse to the channel the framework lacks.

We instantiate the four-tuple in three frameworks with structurally distinct collapse mechanisms and **three real authorization backends**: CrewAI (delegation-constructor boundary; a live PostgreSQL 16 RLS engine on two roles~ [44]), LangGraph (`Command(goto)`; a kernel-enforced POSIX file ACL, mode-`0600` under `sudo -u nobody`), and AutoGen/AG2 (per-recipient role rebinding; the live GitHub REST API's fine-grained PAT scopes, `200`/`404`~ [14]). The security outcome turns on one property — *enforcement on the executing principal, with no channel to learn who asked* — definitional to RBAC, ACLs, and OAuth scope alike, not a guard we authored: the real backend makes every decision (modeled Snowflake/OAuth semantics~ [17, 55] are hermetic fallbacks). Because the collapse is strictly *upstream* of the protected resource, the result is independent of whether a resource is live or modeled: the four-scenario invariant holds verdict-for-verdict on all three real backends (§4.1). The relay's emitted operation is identical between the isolated-high and composed runs, confirming it is a relay, not an attacker.

3.3 The forgeability differential (attributing the cause to the missing channel)

The four-tuple shows *that* composition fails; it does not, by itself, show that the framework's missing channel is *why*. To attribute the cause we make the deputy **defensive and**

identical across cells — a least-privilege-aware component at the protected resource that attempts to attenuate its effective authority to `min(self, initiator)` — and vary only the **channel** from which it recovers the initiator:

Crucially, this defensive deputy is **author-written and on the *defender's* side**: the frameworks wire the protected resource no provenance, so we grant the defender the *strongest charitable defense it could mount* — a component that actively wants to attenuate to the initiator — and show even this best case is bypassed through the only channel the framework carries. We are explicit about the burden split: this differential establishes *only* P2; that *no* native authority-bearing channel exists is carried by the structural reading and the census (§6.2), not by this contrast.

- **In-band (the only channel the framework carries):** the deputy reads the initiator off the message/task text. This text *survives* the boundary (LangGraph's `add_messages` keeps the message objects; AutoGen's role-rebinding changes `role`, not content) — so this is *not* a claim that information is destroyed. With an **honest** request the in-band claim names the true low-priv initiator and the deputy attenuates correctly → `DENY` (the defense *appears* to work). With a **forged** request — the attacker, per the threat model, states a higher origin — the benign relay transcribes it and the deputy attenuates to the forged value → `ALLOW`. **The in-band defense is bypassed by forgery.**
- **Out-of-band (the channel the framework lacks):** the deputy reads the initiator from an integrity-protected channel set by the trusted runtime from the real session, never from request text. The same forged request **cannot move it** → `DENY`.

The deputy code, the request, *and the (non-native) plumbing* are byte-for-byte identical between the in-band-forged and out-of-band cells; the *only* independent variable is the channel's forgeability. So the plumbing we authored cannot be what flips one cell to ALLOW and the other to DENY — only forgeability can. This demonstrates that in-band provenance is an **ambient claim, not a capability** (failing P2), and that closing the gap requires a channel the framework does not provide — the causal attribution (Proposition 2 of §C), carried by this contrast rather than the identical resource operation of §3.2.

3.4 Single-variable ablations and the minimal repair

The composed failure rests on a *single missing premise* — the framework carries no originating, unforgeable channel (P1∧P2) — on top of which **two consumer-side repairs** each independently close the gap, one for each remaining property. These are not two independent root causes: once an out-of-band channel is assumed, they address the two distinct ways a recovered initiator can still be ignored at the protected resource (it never reaches the decision, or it reaches it but is not attenuated against). We test them as single-variable interventions, each presupposing only that provenance has been carried across the boundary (the out-of-band channel of §3.3):

- **E3 — origin-aware authorization (policy layer):** the protected resource consults the retained initiator while the deputy's credential is *unchanged*. The action is denied because the decision is now a function of the initiator.
- **E4 — authority attenuation (capability layer):** the deputy's effective credential is downscoped to `min(deputy, initiator)`; the *native* resource check is unchanged. The action is denied because the downscoped credential fails the existing native check.

Each independently restores `DENY` in all three frameworks; both are the *consumer-side* repairs of §2.1 and **presuppose a channel that already satisfies P1∧P2**, which the framework does not provide. Installing that channel is **non-default and invasive**: a *seam* is a location where correct attenuation forces added code because no native API exists, and in CrewAI, LangGraph, and AutoGen alike the count is the same three — (1) an out-of-band channel to carry the initiator (the payload has no authority field), (2) an attenuation point at the deputy (components bind a *static* credential, no per-call `min(self, initiator)` hook), (3) a runtime that sets the channel at the trust boundary (no native initiator propagation). This is a qualitative repair-distance indicator, not a metric; what is robust is that the count is non-zero and uniform across structurally distinct frameworks. We frame the repair strictly as root-cause validation, not a proposed protocol.

Limitations. Each piece of evidence is scoped in *one* place — the claim-strength summary of §6.4 (Table~5), with full statements in Appendix C (in-band modeling, sink fidelity, controlled-witness vs. real-model check, corpus, falsifiable boundary).

Formal frame. These components instantiate a two-factor controlled experiment (*composition* × *initiator channel*) with the relay held constant and one fixed deputy rule, so each cell isolates one variable; the single-sink truth table and Propositions 1–4 (Prop 1 entailment, so the four-tuple is a controlled witness; Prop 2/P2 and Prop 3/P1 carry the empirical weight; Prop 4 the necessity-with-realized-witnesses synthesis) are in Appendix C (§C).

4. Evaluation

We organize the evaluation around the four-property account (§2.1). After fixing the setup (§4.1), we first put the account to its sharpest test — a pre-registered prediction on

a previously-unseen framework (§4.2) — then turn to the three deeply-examined frameworks: the phenomenon — the (†) violation — reproduces at structurally distinct composition primitives (§4.3), and attribution isolates the missing properties via a forgeability differential and single-variable ablations (§4.4). The climax (§5) shows that every identity primitive frameworks have *shipped* satisfies only a partial subset of {P1–P4}, so none maintains the invariant; supporting checks (§6.1–§6.3) then confirm the path is reachable by real models, is the default across a 13-framework census, and is orthogonal to injection defense. Every deterministic artifact reproduces on the first run (exit 0). PoC sources are in `poc/`, including the framework-agnostic conformance test (`poc/conformance_aggregate.py`) whose output is the §5 per-property matrix.

4.1 Setup

We study three mainstream frameworks at structurally distinct composition primitives against **three real authorization backends** — CrewAI / live PostgreSQL 16 RLS, LangGraph / kernel-enforced POSIX ACL, AutoGen / live GitHub PAT scopes (Table~1; mechanisms and resource fidelity in §3.2)~ [24, 25, 62]. Each agent's LLM is the deterministic dutiful relay (§3.1) except in vivo (§6.1). We additionally **hold out** the OpenAI Agents SDK (`handoff()` boundary) for the pre-registered out-of-sample prediction of §4.2, deliberately *not* examined while building the account.

(We examine both AutoGen generations as one framework family: AG2's `name`-stamped messages for the role-rebinding four-tuple §4.3, autogen-core's per-message `source` field for the failed-fix §5.)

The failed-fix experiment (§5) additionally uses autogen-core/autogen-agentchat `0.7.5` for the per-message `source` field, and the multi-hop P1 differential (§4.4) uses autogen-agentchat `0.7.5`.

Real-backend confirmation. To rule out a *modeled*-backend objection, we replaced the in-process resources with live backends and re-ran the four-tuple, leaving the decision entirely to the real system. Against the live PostgreSQL 16 RLS engine (CrewAI) and live GitHub REST API (AutoGen), the verdicts are identical to the modeled backends — DENY/ALLOW/ALLOW/DENY — and both the composed `ALLOW` and patched `DENY` are produced by the backend itself (`source_of_verdict` ∈ {`postgres-engine`, `github-api`}), with no application guard: composed, the defensive deputy finds no trustworthy initiator channel and executes on its *own* principal (the engine correctly authorizes it — principal substitution); patched, the *same* code attenuates to `min(deputy, initiator)` and the backend refuses. With the kernel POSIX ACL, all three backends are real; modeled implementations are retained as reproducibility fallbacks. Artifacts: `experiments/real-sink-postgres/`, `experiments/real-sink-github/` (the latter ships a tokenless `--replay` fixture).

4.2 A pre-registered out-of-sample test

We first apply the instrument to its sharpest test — a primitive the account has never seen. Because the four-property account is a claim about a *class* of composition primitives, the strongest form of that claim is a prediction: register the verdicts in advance, then check a held-out framework — out-of-sample, not a post-hoc fit. The account predicts that *any* composition primitive whose boundary carries no native, runtime-set, integrity-protected, originating channel will admit principal substitution, regardless of the boundary's surface mechanism. We pre-registered this prediction against a fourth, previously-unexamined framework — the OpenAI Agents SDK `handoffs` primitive (`openai-agents` 0.17.4, HEAD `8eaa4b9`) — fixing the predicted verdicts in version control *before* writing or running the test (committed strictly earlier than the PoC). Its boundary is a mechanically distinct fourth member of the class (Figure~2): the transfer is a model-emitted `transfer_to_<agent>` tool call, and what crosses is a *reprojection of the conversation transcript*. Distinctively among the four, the SDK *retains* the full transcript across the handoff — the original request, including a forgeable origin claim, crosses verbatim — the strongest case for provenance survival. The prediction holds nonetheless: **retention is not authentication** — the retained originator is forgeable, unauthenticated text with no principal–authority binding.

The native handoff satisfies *none* of P1–P4 (§7.1 conformance test). The one place observation refined prediction — the *mechanism* of the P1 failure (the transcript is retained-but-forgeable, not re-stamped to the adjacent peer as for AutoGen's `source`) — we report rather than retrofit (`results.md`).

R4 (out-of-sample). Every verdict we pre-registered for the held-out OpenAI Agents SDK was confirmed; its native handoff satisfies *none* of P1–P4. The account predicts primitives it never saw, not only the ones it was built on.

4.3 The phenomenon: the (†) violation reproduces

Across all three frameworks the results are uniform (Figure~6): `isolated-low` is denied and `isolated-high-benign` allowed (the deputy is legitimate in isolation), while the `composed` cell is allowed — the privileged operation executes though initiated by the low-privilege party, the (†) violation read at the protected resource (ground truth is whether the operation executed, not whether an agent raised). `patched-composed` restores `DENY` in all three. This is the classical confused deputy — a controlled witness, not the finding (Prop 1 of §C); the causal weight rests on the attribution in §4.4.

Table 1: Experimental setup. Three deeply-examined frameworks at structurally distinct composition primitives against three real authorization backends, plus a held-out fourth used only for the pre-registered prediction test (§4.2). The property tested is identical throughout — enforcement on the executing principal, with no native channel to learn who initiated. Pinned commits are in Appendix A.1; each LLM is the deterministic dutiful relay (§3.1) except in vivo (§6.1).

framework	composition primitive (boundary)	authorization backend
CrewAI	task-object reconstruction — `base_agent_tools._execute` rebuilds `Task(...)` (constructor boundary)	**live PostgreSQL 16 RLS** on two real DB roles (modeled Snowflake fallback)
LangGraph	`Command(goto=...)` handoff + `add_messages` reducer (state-flatten boundary)	**kernel POSIX file ACL** — mode-`0600`, `sudo -u nobody` → `EACCES`
AutoGen / AG2	per-recipient role-rebinding — `conversable_agent.py` (serialization boundary)	**live GitHub REST API** — fine-grained PAT scope, `200`/`404` (modeled OAuth fallback)
OpenAI Agents SDK *(held-out)*	transcript-reprojection (the conversation so far is re-passed to the next agent) — `handoff()` model-emitted `transfer_to_<agent>` call (function-transfer boundary)	OAuth scope — HTTP `403` `insufficient_scope`

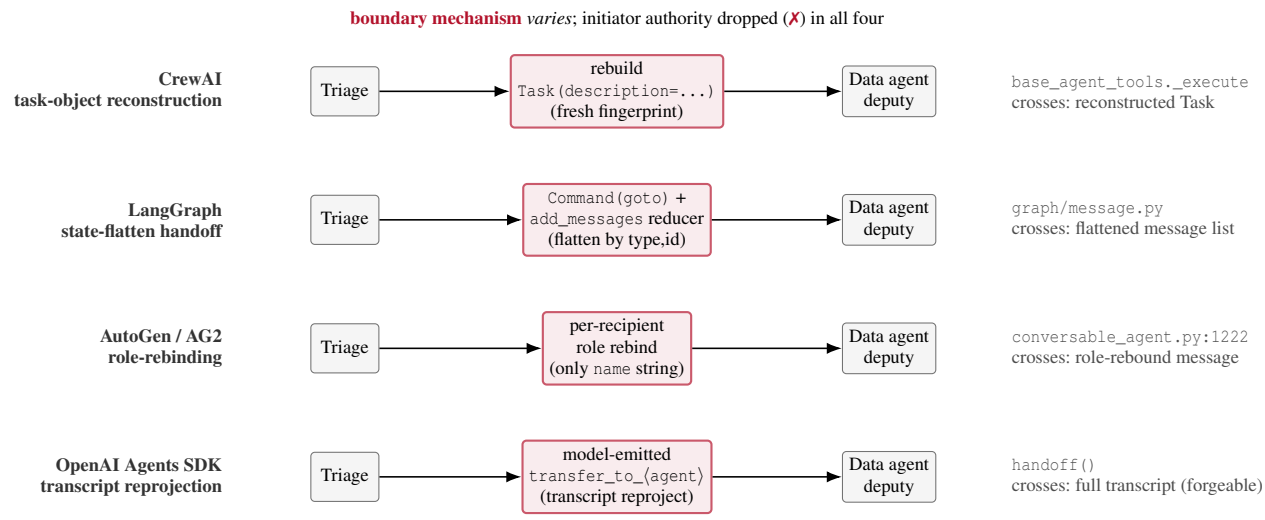


Figure 2: The four composition-primitive boundaries tested in this paper. Each framework loses the initiator's authority by a mechanically different route — a constructor that rebuilds a task object (CrewAI), a reducer that flattens state by id (LangGraph), a per-recipient role rebinding (AutoGen), and a model-emitted transcript reprojection (the held-out OpenAI Agents SDK) — yet the loss recurs identically, as the four-property account predicts (red = the varying boundary mechanism).

R1 († violated). Under the framework default, composition lets an operation a low-privilege initiator could not reach directly (`DENY` in isolation) execute at the protected resource (`ALLOW`): the resource's check passes, but against the deputy — the privileged operation runs though the low-privilege party initiated it.

4.4 Attribution: forgeability (P2) and single-variable ablations (P3, P4)

Forgeability differential (P2). With the defensive deputy of §3.3 (attenuating to $\min(\text{self}, \text{initiator})$) held byte-identical and only the channel varied: in-band, the defense denies an honest request but is **bypassed by a forged one**; out-of-band, the same forged request **cannot move it** (`DENY`). The sole free variable is forgeability — so the carried channel is an ambient claim, not a capability (P2).

R2 (forgeable channel, P2). With byte-identical defensive-deputy code, the initiator recovered in-band moves `low→high` under a forged request — the only free variable is the channel. In-band provenance is an ambient claim, not a capability.

Multi-hop origination differential (P1). Origination bites only across hops, so we test it separately and *with no forgery*.

Table 2: Held-out prediction (OpenAI Agents SDK): pre-registered verdicts vs. observed — *every cell confirmed*, two runs byte-identical, exit 0. Verdict color: DENY=safe, ALLOW=principal substitution / forgery bypass.

cell / contrast	predicted	observed
isolated-low	DENY	DENY
isolated-high-benign	ALLOW	ALLOW
composed	**ALLOW (collapse)**	**ALLOW**
patched-composed (out-of-band)	DENY	DENY
forgeability P2 — in-band honest	DENY	DENY
forgeability P2 — in-band **forged**	ALLOW (bypassed)	**ALLOW**
forgeability P2 — out-of-band forged	DENY	DENY

Table 3: Four-tuple results — *identical verdicts across all three frameworks* (CrewAI/PostgreSQL RLS, LangGraph/POSIX ACL, AutoGen/GitHub PAT scope). Every cell is deterministic and reproduces on the first run (exit 0); the verdict is read at the protected resource (did the privileged operation execute). Shading marks safety: the composed row is the principal-substitution violation; patched-composed restores denial.

cell	verdict	reads
`isolated-low`	DENY	local policy correct
`isolated-high-benign`	ALLOW	deputy legitimate in isolation
`composed`	**ALLOW**	**principal substitution —** (†) **violated**
`patched-composed`	DENY	root-cause repair (validation, not independent evidence)

In a real AutoGen `RoundRobinGroupChat` chain $A \to B \to C$ (A low-priv initiator, B a benign admin-tier relay, C the defensive deputy), the runtime re-stamps each message's `source` to the *producing* agent, so the deputy recovers B (the adjacent peer), not A, and attenuates to B's authority → `ALLOW` where $\min(\text{deputy}, A)$ would `DENY`; an out-of-band channel carrying A end-to-end, and the single-hop control $A \to C$, both `DENY`. The same failure reproduces on a structurally distinct LangGraph `StateGraph` chain (`poc/poc_*_multihop_p1.py`). This makes the A2A corollary (§5) concrete: a single-hop transport-level peer identity authenticates $B \to C$ but not A, collapsing at hop 2 — `source` answers "who spoke last," not "who originated" (Prop 3 of §C).

Single-variable ablations (P3, P4). Given the one missing premise — a P1∧P2 channel — the gap admits *two consumer-side repairs*. **E3** (origin-aware authorization) makes the protected resource consult the retained initiator with the deputy's credential unchanged → `DENY` (restores P3); **E4** (authority attenuation) downscopes the deputy's effective credential to $\min(\text{deputy}, \text{initiator})$ with the native resource check unchanged → `DENY` (restores P4). Each is sufficient alone, but both presuppose the P1∧P2 substrate §5 shows no shipped primitive provides. These are the literature's two defense families read onto our model — origin-aware authorization (SEAgent's provenance graph~ [19]) and capability attenuation (AIP's attenuated tokens~ [45]) — each defends *given* the initiator, while the precondition that the initiator survive the boundary is the gap we measure.

5. Shipped identity primitives: the failed-fix natural experiment

Frameworks have not ignored identity; they have shipped primitives for it. We ask, for each, *which of P1–P4 it satisfies if a defender tried to use it as the authority channel.* P2 is decided at runtime by the forgeability differential of §4.4. P1 is decided at runtime for AutoGen's `source` by the multi-hop origination differential (§4.4) — a re-stamped, hop-local `source` recovers the adjacent peer, not the originator — and by structural/documentation witness for CrewAI's `fingerprint` (a delegated `Task` is rebuilt with a fresh fingerprint) and A2A (transport-layer identity, per its own specification). P3 is structural (native API presence and authorization-path reads, with code citations), as is P4 for LangGraph; P4 for CrewAI's `fingerprint` and AutoGen's `source` is decided at runtime by an attenuation differential. We examine four shipped channels — three framework-native (AutoGen's `source`, CrewAI's `fingerprint`, LangGraph's `config.configurable`) and one protocol-level (A2A's transport identity, a documentation witness) — chosen to span the design space rather than to make the easy case: we deliberately include `config.configurable`, the *strongest* shipped primitive, which satisfies both channel-side properties (P1 and P2) and therefore comes closest to a fix. Table~4 reports the result.

These verdicts are not assembled by hand: they are the output of a framework-agnostic conformance test (`poc/conformance_aggregate.py`) that drives each framework's probe and emits the per-property matrix in a shared schema. Running it reproduces the three framework-native rows below at runtime — AutoGen `source` (P1✗ P2✗ P3✓ P4✗), CrewAI `fingerprint` (P1✗ P2✓ P3✗ P4✗), and LangGraph `config.configurable` (P1✓ P2✓ P3✗ P4✗) — together with the conformant out-of-band channel (all four ✓) uniformly across all of them; A2A enters as a documentation witness, as it ships no runnable framework adapter. The test is the artifact form of the four-property account (§7.1): a maintainer runs it to learn *which* property a primitive is missing rather than assuming an identity field suffices.

Superscripts give each verdict's basis — ^{r} *runtime* (executed differential, §4.4), ^{s} *structural* (native-API / authorization-path code analysis), ^{d} *doc* (the primitive's own specification); the subset-lattice geometry is plotted in Figure~3.

The finding is not that each primitive forgets a different single property; it is sharper and falsifies the intuition that *adding*

Table 4: The failed-fix natural experiment: which of P1–P4 each shipped identity primitive satisfies, and why a defender relying on it is still unsafe. None reaches the conjunction; only the out-of-band channel of §7 is conformant. The rule splits the two *designed* as identity/provenance mechanisms (`source`, `fingerprint` — true failed fixes) from the two that are opportunistic metadata merely *able* to carry the initiator (A2A, `config.configurable`). Per-cell evidence basis (*runtime* vs *structural*) is in Table~9.

identity primitive	P1 orig.	P2 unforg.	P3 reach.	P4 atten.	conf.	key missing property — why a defender using it is still unsafe (plain English)
AutoGen `source` (autogen-core 0.7.5)	$\times^r$	$\times^r$	$\checkmark^s$	$\times^r$	no	**P2 (unforgeable).** The sender field is plain text the attacker can set, so a forged origin is believed.
CrewAI `fingerprint` (commit `051fa0c`)	$\times^r$	$\checkmark^r$	$\times^s$	$\times^r$	no	**P3 (decision-reachable).** An unforgeable UUID that only reaches logs/tracing, never the authorization decision.
A2A transport identity	$\times^d$	$\checkmark^d$	$\times^d$	$\times^d$	no	**P1 (originating).** Authenticates the immediate peer of one hop, not the principal who originated the request.
LangGraph `config.configurable` (0.6.x)	$\checkmark^r$	$\checkmark^r$	$\times^r$	$\times^s$	no	**P3/P4.** Originating and unforgeable, yet the framework never consults it at the protected resource and never attenuates against it.
out-of-band channel (ours, §7)	$\checkmark^r$	$\checkmark^r$	$\checkmark^s$	$\checkmark^r$	**yes**	Satisfies all four jointly — the only configuration that restores correct denial.

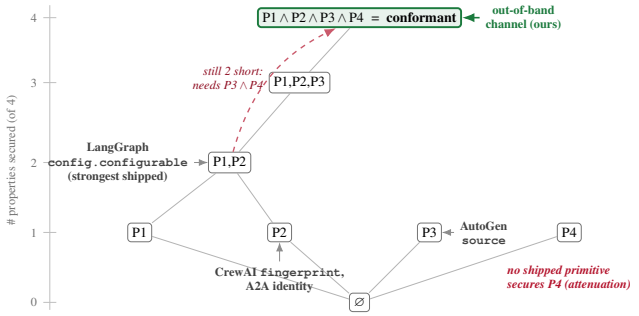


Figure 3: Shipped identity primitives on the subset lattice of {P1,P2,P3,P4} (Table~4, visualized). Each secures only a proper subset and climbs the lattice, yet *none* reaches the conjunction (top = *conformant*); no shipped primitive attenuates (P4), and even the strongest, LangGraph's `config.configurable`, secures the channel side (P1∧P2) yet still misses P3∧P4. The geometry is the claim: *height* = number of properties secured, so the shipped primitives (dark labels) all sit low and green = the conformant apex none of them reaches; red marks the missing properties. An identity field is necessary, not sufficient (*false assurance*). (Occupied positions and the spine to the apex are shown, not the full Hasse diagram.)

an identity field makes a framework safe. The primitives **scatter across the subset lattice of {P1–P4} — each securing a different proper subset, and progressively larger ones — yet not one reaches the conjunction** (Figure~3). This universal over the shipped primitives we examined is *falsifiable*: one securing the conjunction would refute it; none does, though the conformant out-of-band channel shows the target is reachable, so the failure is contingent, not a law.

Reading the verdicts against the code. AutoGen `source`: set by the producer (`messages.py:86`, no validator) and read downstream as trusted text (`_selector_group_chat.py:224`) — decision-reachable (P3✓), but it names the *adjacent* sender, re-stamped each hop (P1✗), and the forgeability differential moves it `low_user`→`admin_user` with the deputy unchanged (P2✗). CrewAI `fingerprint`: an immutable runtime UUID (P2✓) naming the *executing* component — a delegated `Task` gets a fresh one (P1✗) — reaching only tracing (`tool_usage.py:258,489`; zero authorization reads, P3✗); A2A shares this audit-only profile, authenticating the immediate peer (P1✗). LangGraph `config.configurable` (strongest shipped): set out-of-band on the `RunnableConfig` (a forged in-message claim cannot move it, P2✓) and merged run-scoped into every node (`_internal/_config.py:195`, P1✓), yet read only for thread routing (`pregel/remote.py:386`), never authorization — with `configurable.initiator=nobody` the privileged read still ALLOWs (P3✗) and no native min(deputy, init) attenuation (P4✗, structural precisely because the runtime-confirmed P3 failure leaves no consultation point at which an attenuation differential could run); its residual gap is purely *consumer-side* (facts about the *channel*, not a claim that LangGraph apps authenticate users).

P4 (attenuation) is satisfied by *no* shipped primitive and P3 by only one (the forgeable `source`); the channel-side properties *are* achievable — `config.configurable` secures both — which is why its failure is decisive: securing the channel does not help if the framework never consults it under attenuation.

R3 (failed fix — the central result). No shipped identity primitive secures the conjunction P1∧P2∧P3∧P4; even the strongest (`config.configurable`) secures only the channel side and is never consulted at the decision. *Adding an identity field does not make a composed workflow safe.*

6. Breadth and ecological checks

The root cause is established and attributed in §4–§5. This section adds three *supporting* checks — they confirm reach, breadth, and orthogonality, but none carries the causal claim, which rests on the differential (§4.4) and the property matrix (§5). A claim-strength table (Table~5) makes the support boundary of each piece of evidence explicit.

6.1 Real-model check: real models follow the permitted unsafe path

The deterministic results establish and attribute the failure; the relay could be dismissed as a strawman. As a *real-model* check — not a root-cause result — we replace it with **three independently-developed production models** (OpenAI gpt-5.5 `gpt-5.5-2026-04-23`, DeepSeek `deepseek-chat`, Alibaba `qwen-max`) in a real AutoGen `RoundRobinGroupChat`: a low-privilege `triage` agent receives a **benign** request ("export the employee_salaries table…") with *no injection, jailbreak, or forged claim*, and a high-privilege `data_agent` holds the `export_records` protected resource that enforces scope on the *executing* principal. We run three benign scenarios (`employee_salaries`, `payment_methods`, `admin_users`) at ten trials each per model.

All three models, acting as ordinary workers, route the task to the data agent, which fires the sensitive export the initiator could not perform directly (it holds only `export:public`): gpt-5.5 and DeepSeek proceed in **30/30** trials each and qwen-max in **29/30**, for **89 of 90** across the three providers (per-model breakdown in Table~7, Appendix B). The single non-fire was not an authority denial but the model pausing to ask a clarifying question and terminating before the tool call — the framework still carried no initiator channel; the model simply did not proceed that turn.

What this shows. The framework-permitted unsafe path is *behaviorally reachable* by ordinary models under benign input, closing the strawman objection. Routing a benign request to a data agent is the agent's *intended* job, so the rate reflects normal task-following, not an attack. The check runs only on AutoGen (the CrewAI/LangGraph collapse is established deterministically), so 89/90 is an ecological signal, not a safety metric or a root-cause result (Table~5).

All three models are reached through their providers' own official first-party APIs. Appendix B records each endpoint's resolved fingerprint, re-runs Qwen on a date-pinned snapshot (28/30, confirming version-independence), and provides per-trial transcripts.

6.2 Prevalence: the missing channel is the default

Across 13 mainstream frameworks and real applications — a **mainstream-skewed convenience sample, not an ecosystem-wide draw** (explicit inclusion rule and pinned commits in Appendix A.1) — the corpus is 22,818 files with 6,076 delegation/handoff/shared-state sites (Figure~4). Against **two complementary denominators** both yield zero: the **coarse** 6,076 keyword sites (0 carry; 7 proximity candidates all false positives; Wilson 95% UB $\leq 0.061\%$) and the **strict** 384 non-test cross-handoff sites whose 247 identifier-flagged windows were all hand-adjudicated (0 carry; UB $\leq 0.99\%$, a nominal bound — sites cluster within 13 repos, not independent draws, so indicative; A.4). The full protocol — vocabulary, codebook, both denominators, codebook reproducibility (a second deterministic codebook reproduced every verdict; a deliberately lenient coding variant falls to $\kappa = 0.58$, the informative sensitivity figure — A.4) — is in Appendix A, so the census reads as a measurement, not a grep heuristic. The nearest constructs carry no initiator authority for downstream attenuation: an authority-bearing channel is absent by default, within this sample (per-repository counts in Table~6, Appendix~A.1).

> **R5 (absent by default).** Across the 13 mainstream systems people actually build on, **0 of 384** audited cross-handoff sites carry an authority-bearing initiator field (Wilson 95% upper bound $\leq 0.99\%$; the right deployed-reality denominator, not a uniform draw).

6.3 Orthogonality to deployment-grade injection defense

A natural objection is that this is indirect prompt injection re-skinned. It is not. We fed the *exact* §4 attack inputs — the three in-vivo export requests (30/30 on gpt-5.5) and the deterministic CrewAI/AutoGen composed-collapse tasks — to a production injection detector (`protectai/deberta-v3-base-prompt-injection-v2`, the LLM-Guard default, rev `e6535ca`). All five composed-benign inputs were classified **SAFE** ($p \geq 0.986$) yet each triggers principal substitution; the same detector blocked four classic IPI payloads at $p = 1.000$. It passes because there is no injection feature to detect — and this is *detector-class-agnostic*: a composed-benign request is injection-free natural language, so no filter (pattern matcher, classifier, or LLM judge) has an adversarial feature to key on. The failure is therefore orthogonal to deployment-grade injection defense, already visible in ServiceNow's enabled-but-ineffective protection (§1.1; `experiments/defense-orthogonality/`). This dissolves the apparent tension with ServiceNow's second-order injection: injection is a *trigger*; composition is the *amplifier* escalating whatever arrives — injected or benign — to the deputy's authority, which our injection-free tests isolate.

> **R6 (orthogonal to injection).** A production injection detector passes all five composed-benign inputs ($p \geq 0.986$) yet each triggers principal substitution. The harm is *who the protected resource believes is asking*, not adversarial text.

Table 5: Claim-strength summary: what each piece of evidence supports, and what it does not. The root-cause attribution rests on the forgeability and multi-hop differentials (§4.4) and the failed-fix matrix (§5); the remaining rows are supporting.

evidence	supports	does *not* support
Four-tuple	composition can violate (†)	root cause by itself
Forgeability differential	in-band provenance fails P2	prevalence
Multi-hop differential	hop-local identity fails P1	all protocols
Failed-fix matrix	shipped fields are partial subsets	impossibility of a safe design
Census (0/384)	no default channel in audited corpus	ecosystem-wide rate
Real-model (89/90)	ordinary models take the permitted path	framework root cause
Held-out prediction	failure is a property of the primitive class	an exhaustive design-space proof

6.4 What each piece of evidence does and does not show

Table~5 states, for each piece of evidence, what it supports and — equally important — what it does *not*: the differential and failed-fix matrix carry the root-cause claim, everything else is supporting. This single place replaces the per-paragraph disclaimers.

7. Discussion

7.1 A conformance test for composition collapse

The four-property account is operational — a *diagnostic a framework author can run*. We package the §4 property checks as a framework-agnostic conformance test (`poc/conformance_aggregate.py`) that, for each native identity channel, reports which of P1–P4 it satisfies and whether it is conformant (P1/P2 at runtime via the differentials, P3 structural, P4 at runtime for the designed identity primitives and structural for LangGraph). Onboarding a framework needs only an adapter driving its delegation primitive; the held-out OpenAI Agents SDK (§4.2) is the worked example (one adapter; every native channel non-conformant, the out-of-band mitigation conformant). Its lone conformant row is not overfit to our own channel: the positive controls of §2.1 (OAuth on-behalf-of, macaroons, `SECURITY INVOKER`) satisfy the same P1–P4 conjunction in their native settings. A *conformant* verdict is a necessary screen, not a soundness proof: it decides P1–P4 for the one native channel an adapter drives, can pass a conditionally-safe channel (P3/P4 met only via an external decision point or under one configuration), and reads structural cells off code, not an executed denial. Its payoff is locating the *missing* property — and since each maps to one of the three seams (§3.4), the output tells a maintainer *where* to repair, not merely that the channel fails.

7.2 The minimal mitigation is structural, not trivial

The same harness validates the smallest *conformant* channel: an out-of-band initiator reference that is runtime-set, originating, read at the decision, and applied under $\min(\text{deputy}, \text{init})$ attenuation (P1–P4). Wiring it restores correct denial in all three frameworks' `patched-composed` cells (§4.3), the only channel the conformance test finds satisfies all four (§5). It resists the *same* forged origin that defeats in-band provenance: runtime-set and integrity-protected, it cannot be moved by attacker-stated text (the P2 differential), so its safety holds under the adversarial input of our threat model, not merely benign conformance.

A reviewer may read the repair as trivial — "just thread one field." It is not: correct attenuation requires changes at those same three seams, none with a native API or a configuration flag — which is why no framework pays the cost by default and the census (§6.2) finds none that has.

We deliberately do **not** propose a new delegation protocol: prior work already designs integrity-protected substrates (capability tokens, signed hop chains, trust layers, §8), each *presupposing* the channel whose absence we demonstrate. Our contribution is that missing empirical precondition plus the minimal validator.

7.3 Significance: a gap that is activated by default

A reviewer may grant the mechanism and ask: under a benign model, isn't low-to-high routing simply *intended* delegation? It is — but the privilege amplification it silently carries is the security event: the party controlling only untrusted input drives an operation at a protected resource it could never invoke directly, widening the resource owner's policy to *anyone who can reach a privileged deputy* (the amplification least privilege exists to prevent, §2.3). This is distinct from *sanctioned* escalation (an explicit policy grant of the deputy's authority to the initiator): the defect is that no channel exists on which to make that conditional decision, so the amplification is unconditional and silent. We quantify the cost at three levels. **Magnitude (on our fixtures):** within a single resource's authorization domain, composition raises the initiator's *effective* authority to the deputy's, amplifying the reachable privileged-operation set by 2–3$\times$ across our three protected resources (Figure~7); the newly-reachable operations are *exactly* the $(\text{init}, \text{deputy}]$ lattice tier band and nothing above it (§2.3), so we make no ecosystem-wide estimate. **Breadth (witnessed):** the structure is now a default architecture in shipping platforms (ServiceNow Now Assist §1.1, Microsoft

Copilot Studio, Salesforce Agentforce). **Trajectory (convergent):** independent efforts race to build exactly the attenuating substrate P4 names — the right target, not yet the default.

The honest claim is not that a product is exploitable today but that the gap is *activated by default* and broadening — the structure deploys faster than the channel that would make it safe — so the contribution that matters is a check runnable *before* the gap becomes an incident class (§7.1). Studying open frameworks rather than a black-box product is deliberate: a product probe could reproduce an `ALLOW`/`DENY` but not isolate *which* of P1–P4 is missing — our contribution — and would reintroduce the model confounder our method removes.

8. Related Work

The delta in one line (§1.4): prior work *asserts* the authority-bearing channel *should* exist; we *measure*, at the implementation layer, that it does not, by default. We position our work among five lines of research, including recent surveys of LLM-agent and multi-agent security~ [11, 22, 28]; they converge on one message — secure composition needs an integrity-protected authority substrate — which they design or assume rather than measure. Our vantage is the intersection of **(object)** orchestration-framework *implementation* code, **(method)** a deterministic isolated-vs-composed differential, and **(phenomenon)** *principal substitution*; Appendix D (Table~10) places each line relative to that gap.

Conceptual framings. Shi et al.'s SoK on the *Trust-Authorization Mismatch*~ [52] unifies a broad threat surface under static permissions decoupled from runtime trustworthiness, via a Belief–Intention–Permission lens, whose vocabulary we adopt. It works at a different level: a probabilistic model that explicitly excludes multi-agent coordination frameworks and lists implementation-level empirics at the composition boundary as open work — the question we take up (*authority-bearing provenance loss → principal substitution*), in the deterministic regime where the *same* task is safe in isolation and unsafe once composed. Concurrent conceptual work stays at a governance or requirements layer rather than an authority-carrying channel over shipped boundaries~ [42, 54, 57, 65].

Privilege-escalation and confused-deputy defenses. A defense line treats agent over-privilege directly: PFI~ [21] separates trusted from untrusted agents, Progent~ [53] enforces least privilege via a tool-level policy DSL, and SEAgent~ [19] builds a MAC/ABAC framework that — most relevant to us — reproduces a multi-agent confused-deputy PoC on real MAS runtimes. We **cede** novelty in *discovering* multi-agent confused-deputy behavior: SEAgent demonstrates it. It asks a different question — how to *enforce* a policy given the principal — and its defense *reconstructs* provenance over an external graph; we ask the prior question of whether the initiator survives the boundary at all, which that reconstruction presupposes.

Provenance- and identity-aware authorization protocols. A complementary line proposes cryptographic substrates for delegation: AuthGraph~ [59] aligns reasoning and authorization graphs to detect indirect prompt injection; AIP~ [45] introduces invocation-bound capability tokens fusing identity, attenuation, and provenance across MCP/A2A; HDP~ [9] records each hop as an Ed25519-signed link; KYA~ [47] adds a framework-agnostic trust layer; and related identity-standard efforts move from advisory identity toward authenticated delegation~ [1, 4, 29, 36, 37, 56]. These validate our premise; our place is one layer below: an implementation-level differential measuring whether mainstream frameworks carry the channel these substrates presuppose (§5). P3/P4's consumer-side enforcement is itself standard systems security — OS provenance, Linux credential passing and ambient capabilities, multi-hop microservice identity propagation beyond OAuth on-behalf-of — so P1–P4 project established practice onto the agent boundary rather than reinvent it.

Composition-induced and multi-tool vulnerabilities. ChainFuzzer~ [61] discovers workflow vulnerabilities via greybox fuzzing — *tool-to-tool dataflow taint* (does *data* reach a dangerous sink), with no notion of principal, as do prompt-injection and tool-agent benchmarks~ [10, 48, 66, 68]. The closest detection line is taint-style RCE discovery: LLMSmith~ [31] recovers source-to-sink chains then crafts exploit prompts (11 CVEs), AgentFuzz~ [30] uses directed fuzzing (23 CVEs). These study a different object — the *data→sink* path under an adversarial payload — where we study the *initiator→deputy* authority path under benign input, with access control sound but enforced on the wrong principal. Privacy/extraction risks~ [6] are orthogonal.

Protocol- and architecture-level evaluation. Three evaluation efforts sit adjacent: AgentRFC~ [69] model-checks MCP/A2A/ANP/ACP *specifications* as TLA$^+$ invariants (never framework code; implementation testing "ongoing"); Agent-Fence~ [46] scores agent *archetypes* by break rate under a *Wrong-Principal Action* predicate; Agentproof~ [63] statically verifies workflow-*graph* topology across LangGraph/CrewAI/AutoGen/ADK; and broader taxonomies inventory threats at the system level~ [38]. We **cede** terminology — principal substitution is a specific implementation-level subcase of Agent-Fence's wrong-principal class — but sit one layer beneath: Agent-Fence scores archetype break-rate and cannot say *which* of P1–P4 is missing, the per-property attribution we supply (full object × method × causal-claim comparison in Appendix D, Table~10).

9. Conclusion

Agents safe in isolation can become unsafe purely by being composed: a framework's composition boundary drops the initiator's authority, so a sound, unbypassed check at the

protected resource is enforced against the deputy — principal substitution. No shipped identity primitive carries the four properties an authority channel must *jointly* satisfy (false assurance, confirmed out-of-sample and absent by default across our census), so we package the check as a runnable conformance test, not a new protocol. The lesson: *verify the conjunction, not the field*.

Ethical Considerations

This work is a design-level measurement, not an exploit against a deployed product: under a fully benign threat model (no malicious agent, no injected payload) we show that mainstream frameworks lack an authority-bearing initiator channel at the composition boundary. We considered the stakeholders affected — framework maintainers, developers who build on them, and end users whose data sits behind the protected resources — and structured the work to inform rather than endanger them.

No weaponizable artifact. Our proof-of-concept code exercises the frameworks' *published* code under *benign* inputs against real but non-production authorization backends that we control: a local PostgreSQL RLS instance, a kernel-enforced POSIX ACL under two OS uids, and a *private* GitHub test repository. The in-vivo check uses a benign request with no injection and no jailbreak. No PoC targets a live third-party system, carries an exploit payload, or demonstrates exfiltration beyond a canary in our own fixtures; the "sensitive" resources are synthetic.

Disclosure. Because we name specific frameworks, we notify their maintainers (AutoGen, CrewAI) ahead of submission and share the conformance test so they can detect and close the gap. We treat this as a *courtesy notification* rather than an embargoed vulnerability report, for two reasons. First, the behavior is a design default, not a memory-safety-style defect with a discrete patch to coordinate. Second, the gap is already public — visible in the frameworks' own documentation and issue trackers (§1.2), in shipped-but-partial identity primitives (§5), and in a disclosed real-world incident (ServiceNow, §1.1) — so there is no embargo whose delay would reduce risk.

Net assessment. The constructive output — the four-property conformance test and the minimal conformant channel — lets maintainers and deployers detect and close the gap; releasing it does more good than harm. We report the work as a measurement and a diagnostic, not as a vulnerability against any single vendor.

Open Science

We are committed to open science and will release every artifact needed to reproduce our claims. The artifact comprises: the per-framework deterministic composition four-tuples (CrewAI, LangGraph, AutoGen) and the held-out OpenAI Agents SDK prediction harness; the forgeability (P2) and multi-hop origination (P1) differentials and the single-variable P3/P4 ablations; the framework-agnostic conformance test that emits the per-property matrix; the real-backend harnesses (PostgreSQL RLS, GitHub fine-grained PAT) and the kernel POSIX ACL setup; the prevalence-census scanner, non-test site extractor, adjudication codebook, and inter-rater classifier; and the in-vivo harness with per-trial transcripts and resolved model fingerprints. Every framework under test is pinned to a specific commit or release (Table~1, Appendix A.1).

At submission these artifacts are provided as an anonymized supplementary archive (link withheld for double-blind review); upon acceptance we will publish them in a public repository under an open-source license. The deterministic components reproduce on the first run (exit 0) with no external dependencies beyond the pinned framework sources. The in-vivo check depends on third-party production-model APIs we do not control; we therefore include recorded per-trial transcripts and model fingerprints (Appendix B) so the reported rates remain inspectable even where live re-execution is subject to provider model drift.

References

[1] A2A Protocol. Agent2agent protocol specification. https://a2a-protocol.org/latest/specification/, 2026. Official protocol specification; accessed 2026-06.

[2] Martín Abadi, Michael Burrows, Butler Lampson, and Gordon Plotkin. A calculus for access control in distributed systems. *ACM Transactions on Programming Languages and Systems*, 15(4):706–734, 1993.

[3] AppOmni AO Labs and Aaron Costello. When AI turns on its team: Exploiting agent-to-agent discovery via prompt injection. https://appomni.com/ao-labs/ai-agent-to-agent-discovery-prompt-injection/, November 2025. Accessed 2026-06.

[4] Varun Pratap Bhardwaj. Formal analysis and supply chain security for agentic AI skills, 2026.

[5] Arnar Birgisson, Joe Gibbs Politz, Úlfar Erlingsson, Ankur Taly, Michael Vrable, and Mark Lentczner. Macaroons: Cookies with contextual caveats for decentralized authorization in the cloud. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2014.

[6] Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data

from large language models. In *30th USENIX Security Symposium*, 2021. arXiv:2012.07805.

[7] CrewAI. Issue #5360: Cryptographic identity for agents. https://github.com/crewAIInc/crewAI/issues/5360, 2026. GitHub issue; accessed 2026-06.

[8] CrewAI. Tool hooks. https://docs.crewai.com/en/learn/tool-hooks, 2026. Official documentation; accessed 2026-06.

[9] Asiri Dalugoda. HDP: A lightweight cryptographic protocol for human delegation provenance in agentic AI systems, 2026. Helixar Limited; also published as IETF Internet-Draft draft-helixar-hdp-agentic-delegation-00.

[10] 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 *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2406.13352.

[11] Zehang Deng, Yongjian Guo, Changzhou Han, Wanlun Ma, Junwu Xiong, Sheng Wen, and Yang Xiang. AI agents under threat: A survey of key security challenges and future pathways. *ACM Computing Surveys*, 57(7), 2025. Article 182; arXiv:2406.02630.

[12] Jack B. Dennis and Earl C. Van Horn. Programming semantics for multiprogrammed computations. *Communications of the ACM*, 9(3):143–155, 1966.

[13] Carl Ellison, Bill Frantz, Butler Lampson, Ron Rivest, Brian Thomas, and Tatu Ylonen. SPKI certificate theory. RFC 2693, IETF, 1999.

[14] GitHub. Managing your personal access tokens. https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens, 2026. Official documentation; accessed 2026-06.

[15] 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 (AISec)*, 2023. arXiv:2302.12173.

[16] Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents: A survey of progress and challenges. In *Proceedings of the 33rd International Joint Conference on Artificial Intelligence (IJCAI)*, 2024. arXiv:2402.01680.

[17] Dick Hardt. The OAuth 2.0 authorization framework. RFC 6749, IETF, 2012.

[18] Norm Hardy. The confused deputy: (or why capabilities might have been invented). *ACM SIGOPS Operating Systems Review*, 22(4):36–38, 1988.

[19] Zimo Ji, Daoyuan Wu, Wenyuan Jiang, Pingchuan Ma, Zongjie Li, Yudong Gao, Shuai Wang, and Yingjiu Li. Taming various privilege escalation in LLM-based agent systems: A mandatory access control framework, 2026.

[20] Michael B. Jones, Anthony Nadalin, Brian Campbell, John Bradley, and Chuck Mortimore. OAuth 2.0 token exchange. RFC 8693, IETF, 2020.

[21] Juhee Kim, Woohyuk Choi, and Byoungyoung Lee. Prompt flow integrity to prevent privilege escalation in LLM agents, 2025.

[22] Juhee Kim, Xiaoyuan Liu, Zhun Wang, Shi Qiu, Bo Li, Wenbo Guo, and Dawn Song. The attack and defense landscape of agentic AI: A comprehensive survey, 2026. Accepted to USENIX Security 2026.

[23] Butler W. Lampson. Protection. *ACM SIGOPS Operating Systems Review*, 8(1):18–24, 1974. Originally Proc. 5th Princeton Symp. on Information Sciences and Systems, 1971.

[24] LangChain. LangGraph low-level concepts. https://langchain-ai.github.io/langgraph/concepts/low_level/, 2026. Official LangGraph documentation; accessed 2026-06.

[25] LangChain. LangGraph multi-agent systems. https://langchain-ai.github.io/langgraph/concepts/multi_agent/, 2026. Official LangGraph documentation; accessed 2026-06.

[26] Henry M. Levy. *Capability-Based Computer Systems*. Digital Press, 1984.

[27] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. arXiv:2005.11401.

[28] Yuanchun Li, Hao Wen, Weijun Wang, Xiangyu Li, Yizhen Yuan, Guohong Liu, Jiacheng Liu, Wenxing Xu, Xiang Wang, Yi Sun, Rui Kong, Yile Wang, Hanfei Geng, Jian Luan, Xuefeng Jin, Zilong Ye, Guanjing Xiong, Fan Zhang, Xiang Li, Mengwei Xu, Zhijun Li, Peng Li, Yang Liu, Ya-Qin Zhang, and Yunxin Liu. Personal LLM agents: Insights and survey about the capability, efficiency and security, 2024.

[29] Zibin Lin, Shengli Zhang, Guofu Liao, Dacheng Tao, and Taotao Wang. Binding agent ID: Unleashing the power of AI agents with accountability and credibility, 2025.

[30] Fengyu Liu, Yuan Zhang, Jiaqi Luo, Jiarun Dai, Tian Chen, Letian Yuan, Zhengmin Yu, Youkun Shi, Ke Li, Chengyuan Zhou, Min Yang, and Hao Chen. Make agent defeat agent: Automatic detection of taint-style vulnerabilities in LLM-based agents. In *34th USENIX Security Symposium*, 2025. https://www.usenix.org/conference/usenixsecurity25/presentation/liu-fengyu.

[31] Tong Liu, Zizhuang Deng, Guozhu Meng, Yuekang Li, and Kai Chen. Demystifying RCE vulnerabilities in LLM-integrated apps. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1716–1730, 2024. arXiv:2309.02926.

[32] Microsoft AutoGen. GroupChat reference. https://microsoft.github.io/autogen/0.2/docs/reference/agentchat/groupchat/, 2024. Official AutoGen 0.2 documentation; accessed 2026-06.

[33] Microsoft AutoGen. Discussion #7432: Message source enforcement. https://github.com/microsoft/autogen/discussions/7432, 2026. GitHub discussion; accessed 2026-06.

[34] Mark S. Miller, Ka-Ping Yee, and Jonathan Shapiro. Capability myths demolished. Technical Report SRL2003-02, Johns Hopkins University Systems Research Laboratory, 2003.

[35] Mark Samuel Miller. *Robust Composition: Towards a Unified Approach to Access Control and Concurrency Control*. PhD thesis, Johns Hopkins University, 2006.

[36] Model Context Protocol. Specification. https://modelcontextprotocol.io/specification/, 2026. Official protocol specification; accessed 2026-06.

[37] Subramanya Nagabhushanaradhya. OpenID connect for agents (OIDC-A) 1.0: A standard extension for LLM-based agent identity and authorization, 2025.

[38] Tam Nguyen, Moses Ndebugre, and Dheeraj Arremsetty. Security considerations for multi-agent systems, 2026.

[39] OpenAI. GPT-4 technical report, 2023.

[40] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. arXiv:2203.02155.

[41] OWASP GenAI Security Project. OWASP top 10 for LLM applications 2025. https://genai.owasp.org/llm-top-10/, 2025. LLM01 Prompt Injection; LLM05 Improper Output Handling; LLM06 Excessive Agency; accessed 2026-06.

[42] KrishnaSaiReddy Patil. SentinelAgent: Intent-verified delegation chains for securing federal multi-agent AI systems, 2026. HIGH novelty-collision; new defense protocol + DelegationBench, presupposes the channel we show absent.

[43] Fábio Perez and Ian Ribeiro. Ignore previous prompt: Attack techniques for language models, 2022. NeurIPS 2022 ML Safety Workshop.

[44] PostgreSQL Global Development Group. Row security policies. https://www.postgresql.org/docs/current/ddl-rowsecurity.html, 2026. Official documentation; accessed 2026-06.

[45] Sunil Prakash. AIP: Agent identity protocol for verifiable delegation across MCP and A2A, 2026.

[46] Sai Puppala, Ismail Hossain, Md Jahangir Alam, Yoonpyo Lee, Jay Yoo, Tanzim Ahad, Syed Bahauddin Alam, and Sajedul Talukder. Agent-fence: Mapping security vulnerabilities across deep research agents, 2026.

[47] Kolawole Quadri. KYA: A framework-agnostic trust layer for autonomous systems with verifiable provenance and hierarchical policy composition, 2026.

[48] 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 *International Conference on Learning Representations (ICLR)*, 2024. ToolEmu; arXiv:2309.15817.

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

[50] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.

[51] 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 (NeurIPS)*, 2023. arXiv:2302.04761.

[52] Guanquan Shi et al. SoK: Trust-authorization mismatch in LLM agent interactions, 2025.

[53] Tianneng Shi, Jingxuan He, Zhun Wang, Hongwei Li, Linyu Wu, Wenbo Guo, and Dawn Song. Progent: Securing AI agents with privilege control, 2025.

[54] Vincent Siu, Jingxuan He, Kyle Montgomery, Zhun Wang, Neil Gong, Chenguang Wang, and Dawn Song. A framework for formalizing LLM agent security, 2026. terminological "four properties" clash; disambiguate from our four channel-properties.

[55] Snowflake. Overview of access control. https://docs.snowflake.com/en/user-guide/security-access-control-overview, 2026. Official documentation; accessed 2026-06.

[56] Tobin South, Samuele Marro, Thomas Hardjono, Robert Mahari, Cedric Deslandes Whitney, Dazza Greenwood, Alan Chan, and Alex Pentland. Authenticated delegation and authorized AI agents, 2025.

[57] Krti Tallam. Authorization propagation in multi-agent AI systems: Identity governance as infrastructure, 2026. MED-HIGH collision: "not reducible to prompt injection"; benign behavior already triggers failures; requirements-level, no differential/failed-fix.

[58] The Hacker News. ServiceNow AI agents can be tricked into acting against each other via second-order prompts. https://thehackernews.com/2025/11/servicenow-ai-agents-can-be-tricked.html, November 2025. Accessed 2026-06.

[59] Peiran Wang, Ying Li, and Yuan Tian. Aligning provenance with authorization: A dual-graph defense for LLM agents, 2026. UCLA; arXiv:2605.26497v1 [cs.CR], 26 May 2026.

[60] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. arXiv:2201.11903.

[61] Jiangrong Wu, Zitong Yao, Yuhong Nan, and Zibin Zheng. ChainFuzzer: Greybox fuzzing for workflow-level multi-tool vulnerabilities in LLM agents, 2026.

[62] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed H. Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversation, 2023.

[63] Melwin Xavier, Vaisakh M A, Melveena Jolly, and Midhun Xavier. Agentproof: Static verification of agent workflow graphs, 2026.

[64] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, et al. The rise and potential of large language model based agents: A survey, 2023.

[65] Zijie Xu, Minfeng Qi, Shiqing Wu, Lefeng Zhang, Qiwen Wei, Han He, and Ningran Li. The trust paradox in LLM-based multi-agent systems, 2025. Full title: The Trust Paradox in LLM-Based Multi-Agent Systems: When Collaboration Becomes a Security Vulnerability.

[66] Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains, 2024.

[67] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2210.03629.

[68] Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10471–10506, 2024. arXiv:2403.02691.

[69] Shenghan Zheng and Qifan Zhang. AgentRFC: Security design principles and conformance testing for agent protocols, 2026.

Appendix A. Prevalence census: reproducible measurement protocol

This appendix specifies the prevalence census of §6.2 in enough detail to re-run it. The intent is to elevate the result from a keyword scan to a measurement whose corpus, extraction vocabulary, adjudication rule, and uncertainty are all stated and independently checkable. Scripts and raw outputs are in `experiments/w3-corpus-prevalence/` (`scan.py`, `census_nontest_sites.py`, `sample_sites.py`, `interrater_kappa.py`, with `results.json`, `census-summary.json`, and the dumps they emit).

A.1 Corpus selection (inclusion criteria)

The corpus is a fixed set of 13 repositories drawn from already-cloned mainstream multi-agent frameworks and well-known end-to-end multi-agent applications; we add no new clone. It is therefore a **mainstream-skewed convenience sample, not a uniform GitHub draw**, and the prevalence figures should

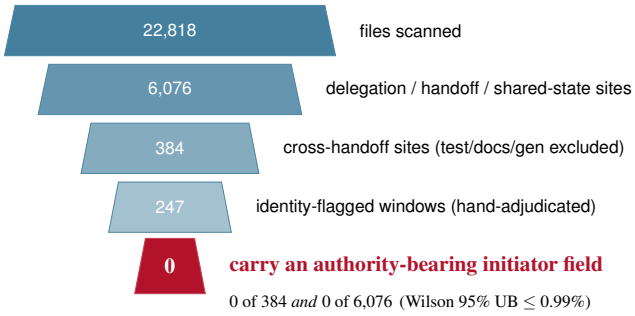


Figure 4: Prevalence census. Across 22,818 files and 6,076 delegation sites, no site carries an authority-bearing initiator field (both denominators; Wilson 95% upper bound $\leq$ 0.99%). Darkening blue = successive filtering; the **red terminal stage** is the result — every audited cross-handoff site has a name/identity field yet **zero** carry initiator authority (field $\neq$ safety).

be read as "in mainstream frameworks and representative real applications," not as an ecosystem-wide statistical inference (stated also in Appendix C). Inclusion required that a repository (i) be a multi-agent orchestration framework *or* a deployed multi-agent application; (ii) be public and buildable from source in Python, TypeScript/JavaScript; and (iii) be pinnable to a specific commit so the measurement is stable. Each repository is recorded with its category, pinned commit, commit date, the file extensions scanned, and the number of files actually traversed.

Extensions are chosen per project: pure-Python frameworks are scanned as `.py`; full-stack applications add `.ts/.tsx` (and `.js/.jsx` for the JavaScript backends of Flowise and AnythingLLM, whose `.py` counts are zero). We set **no top-N cap** on files or repositories — every target-extension file in each included repository is traversed. Directory pruning removes only vendored, generated, or cache trees (`.git`, `node_modules`, `dist`, `build`, `__pycache__`, `.venv/venv`, `.next`, `out`, `vendor`, `site-packages`, `.mypy_cache`, `.pytest_cache`, `migrations`); these contain no framework delegation logic. Files that fail strict UTF-8 decoding are counted per repository and skipped (a negligible, non-source residue). For the **strict** denominator we additionally exclude test/docs/examples/generated paths by an explicit, audited substring rule (`/test`, `test_`, `_test`, `/tests/`, `.test.`, `.spec.`, `/__mocks__/`, `/docs/`, `/doc/`, `/examples/`, `/example/`, `/sample`, `conftest`, `/fixtures/`, `/generated/`, `_pb2`, `.pb.`, `/mock`, `/demo`); the count excluded (1,842 handoff-type sites) is reported, not silently dropped.

A.2 Site extraction (the matching vocabulary)

Delegation/handoff/shared-state sites (coarse denominator). A *site* is a source line matching one of 21 word-anchored patterns (all `\b`-anchored to avoid substring hits), grouped by the composition mechanism it expresses:

- **delegation** (task/work delegation): `delegate`, `allow_delegation`, `DelegateWorkTool`, `coworker`, `sub_agent`/`sub_agents`, `spawn_agent`.
- **handoff** (control transfer): `handoff`, `transfer_to*`, `create_handoff*`, `Command(goto ...`.
- **shared-state** (state-object construction/merge): `add_messages`, `StateGraph`, `Send(`.
- **serialization / message construction** (task-object reconstruction, role-rebinding, nested chats): `Task(`, `Crew(`, `GroupChat`, `ConversableAgent`, `initiate_chat(s)`, `register_reply`, `register_nested_chat`.

This vocabulary yields **6,076** delegation sites across the corpus — the coarse denominator. The delegation patterns hit broadly in the wild (`StateGraph`, `Task(`, `add_messages`, `GroupChat`, `ConversableAgent`, `Send(`, `initiate_chat`, `handoff`, `delegate`, `coworker`, `allow_delegation`, ...), confirming the structure is common, not a rare form we constructed.

Handoff-type call sites (strict denominator). For the strict census we restrict to the 14 patterns that denote an *actual transfer of task/control* to a downstream agent, deliberately dropping bare type-references/imports (`StateGraph`, `ConversableAgent`, `GroupChat`) that inflate the coarse denominator without being authority-carrying call sites: `delegate`, `handoff`, `transfer_to*`, `create_handoff*`, `coworker`, `allow_delegation`, `DelegateWorkTool`, `Task(`, `Command(goto`, `Send(`, `initiate_chat(s)`, `register_nested_chat`, `sub_agent(s)`, `spawn_agent`. After the A.1 exclusions this leaves **384** non-test handoff-type sites.

Authority-field vocabulary (strict, 16 words). A site *carries an authority field* (the lower-bound metric) if an authority/provenance whole-word token appears within ± 10 lines: `initiator`, `on_behalf_of`, `principal`, `authority`, `source_principal`, `caller_identity`, `delegator`, `originator`, `requester_role`, `acting_as`, `provenance`, `invoker`, `delegated_authority`, `source_role`, `initiator_role`, `caller_role`.

Identifier-aware identity/authority family (flagging for adjudication). To avoid undercounting non-keyword carriers, each strict-denominator window is tokenized into identifier subwords (snake_case, camelCase, PascalCase, and digit splits) and matched against an extended identity/authority word family spanning principals (`user`, `account`, `owner`, `member`, `principal`, `identity`, `subject`, `actor`, `requester`, `caller`, `initiator`, `delegator`, `originator`, `invoker`, `author`, `creator`, `sender`), sessions/credentials (`session`, `token`, `jwt`, `claim(s)`, `credential(s)`, `apikey`, `key`, `secret`, `bearer`), authorization (`auth`, `authz`/`authn`, `authentication`/`authorization`, `permission(s)`, `perm(s)`, `scope(s)`, `role(s)`, `acl`, `grant(s)`, `policy`,

repository	category	commit	commit date	extensions	files scanned
crewAI	framework (CrewAI lib + tools)	`051fa0c`	2026-06-03	.py	1,207
autogen	framework (AutoGen/AG2 lib + examples + test)	`7cd0b10`	2026-06-05	.py	1,601
langgraph	framework (LangGraph lib + examples)	`43682f0`	2026-06-03	.py	445
metagpt	framework (MetaGPT)	`11cdf46`	2026-01-21	.py	890
superagi	framework (SuperAGI)	`c3c1982`	2025-01-22	.py	426
camel-ai	framework (CAMEL workforce + examples)	`cc2de7a`	2026-06-02	.py	1,130
langroid	framework (Langroid multi-agent)	`763d5cb`	2026-05-21	.py	424
autogpt (app)	application (AutoGPT platform)	`2ca389e`	2026-06-02	.py/.ts/.tsx	2,796
dify (app)	application (Dify)	`c8abb11`	2026-06-04	.py/.ts/.tsx	9,523
openhands (app)	application (OpenHands)	`64ae075`	2026-06-04	.py/.ts/.tsx	1,904
taskweaver (app)	application (TaskWeaver)	`d44ddef`	2026-03-23	.py	167
flowise (app)	application (Flowise, TS)	`a4c4e49`	2026-06-04	.ts/.tsx	1,225
anythingllm (app)	application (AnythingLLM, JS)	`43e0d9a`	2026-06-03	.js/.jsx	1,080
total					**22,818**

Table 6: Audited cross-handoff sites (prevalence census), per repository: the coarse delegation/handoff/shared-state site count and the authority-carrying count (the result column, tinted; **0 throughout**). Both denominators, bounds, and protocol are in §6.2 and A.4. Four repositories (SuperAGI, TaskWeaver, Flowise, AnythingLLM) contributed zero *handoff-type* sites under our strict vocabulary (A.4) — reported, not hidden.

repository	files	delegation sites	authority-carrying
crewAI	1,207	1,184	0
autogen	1,601	2,868	0
langgraph	445	1,292	0
metagpt	890	47	0
superagi	426	0	0
camel-ai	1,130	122	0
langroid	424	452	0
autogpt (app)	2,796	56	0
dify (app)	9,523	31	0
openhands (app)	1,904	11	0
taskweaver	167	0	0
flowise	1,225	7	0
anythingllm	1,080	6	0
total	**22,818**	**6,076**	**0**

`privilege(s)`, `capability`), multi-tenancy (`tenant`, `org`, `organization`, `workspace`, `project`, `group`, `team`, `namespace`), carriers (`context`/`ctx`, `request`/`req`, `metadata`/`meta`, `headers`/`header`, `cookie(s)`), and provenance (`provenance`, `source`, `origin`, `onbehalf`/`behalf`, `current`, `delegated`, `acting`). This flags **247** of the 384 windows for manual adjudication — a deliberately over-inclusive net so that the negative result cannot be charged to a too-narrow vocabulary.

A.3 Adjudication codebook

Positive criterion (necessary conditions). A flagged window is coded **POSITIVE** — a genuine authority-bearing initiator field — only if it carries a value that is jointly (i) **initiator-bound**: it names the *originating* external principal, not the adjacent peer, the executing agent, or a persona/role string; (ii) **runtime-set**: bound as a code parameter (a kwarg or dict key on the handoff call, not a comment/docstring) by trusted code rather than copied from request text; (iii) **integrity-protected**: not derived from attacker-influenceable content; and (iv) **sink-consumable**: propagated across the handoff toward a downstream authorization decision, not confined to logging/tracing/routing. The frozen codebook operationalizes this as: POSITIVE iff a strict originating-authority token is bound *as a code parameter* (kwarg/dict key, not a comment line) in the *same* window as the handoff call. All 247 windows coded NEGATIVE.

False-positive families (with representative snippets). Every flagged-but-negative window falls into one of the following recurring families; each is identity-adjacent but fails at least one necessary condition:

- **Transport/endpoint authentication** (fails *initiator-bound*): `crewai/a2a/utils/delegation.py:175,274` — `auth: ClientAuthScheme` authenticates the *delegating client* to a remote A2A endpoint, not the external initiator to a downstream protected resource.
- **Resource-sandbox inheritance** (fails *initiator-bound*; grants *less*, not authority): `autogpt/.../execution_context.py` — `create_child_context` hands a child agent a restricted file store (`.sub_agents/{id}/`), a reduced budget, and a `parent_agent_id`; this is isolation/budget, not initiator principal.
- **Channel/capability tag and task ownership** (fails *initiator-bound*): `autogen/beta/network/.../delegate.py:47` — `delegate(target, prompt, *, capability=None, ...)` crosses only `prompt` (free text) plus a `capability` channel knob; `beta/agent.py:705` — `Task(owner_id=self.name, ...)` records the *owner agent's own* name.

- **Control-flow target / back-pointer** (fails *sink-consumable*): `autogen/.../transitions.py:273` — `RevertToInitiatorTarget` is a group-chat routing target; `langroid/agent/task.py:456` — `self.caller` is a parent-task pointer for done-signaling.
- **Web-auth endpoint, not an agent handoff** (fails *the handoff predicate itself*): `dify/.../workspace/members.py:417` — `transfer_token` for workspace-ownership transfer, and `dify/.../auth/surface_gate.py` — an OAuth surface gate (`bearer`/`subject`); both are HTTP controllers caught by the `transfer_to`/`delegate` verb, carrying real authorization but not across an agent-to-agent boundary.
- **Advisory name string** (fails *integrity-protected*): `source`/`sender` are message/event author *name strings* set by the producing agent; `role` is an agent-persona or message-role string. These are exactly the ambient claims §5 shows are forgeable.

The low-signal remainder (216 windows) is dominated by `context` (101; free-text task context), `user` (43) and `role` (30; message-role / persona), `sender` (36; AutoGen message-sender name), `current` (34; `current_speaker`/`current_agent` control flow), and `group` (20; group-chat vocabulary) — none a sink-consumable, integrity-protected initiator authority.

A.4 Both denominators, uncertainty, and codebook reproducibility

We report the carrier rate against both denominators, each with its Wilson 95% upper bound under the observed zero count:

Reporting both guards against the appearance of selectivity: the strict denominator gives the more conservative (wider) bound, and we quote it in the body rather than hiding behind the tighter coarse-denominator number. Both Wilson bounds assume independent Bernoulli trials, which the sites are not: the 384 strict sites cluster within 13 repositories (and share idioms within each framework), so the effective sample size is smaller than the nominal count and the true interval is wider. We therefore read the bounds as *indicative of a mainstream-default absence*, not as a calibrated ecosystem-wide rate — consistent with the convenience-sample framing of A.1; the qualitative result (no authority-bearing channel at any audited site, across all 13 repositories) does not depend on the interval. Four repositories (SuperAGI, TaskWeaver, Flowise, AnythingLLM) contributed zero handoff-type sites under our vocabulary — their JS/TS or bespoke orchestration does not use these idioms; this is reported as a coverage fact, not folded into the denominator.

Codebook reproducibility. To show the all-negative adjudication is reproducible rather than idiosyncratic, a second, deterministic codebook classifier — blind to the first pass's per-item verdicts — re-coded all 247 windows, with 15 synthetic authority-carrying handoff sites injected as positive controls so the agreement statistic is well-defined under otherwise single-class labels. The strict codebook reproduced the adjudication exactly (247/247 corpus windows NEGATIVE, 15/15 controls detected; Cohen's $\kappa = 1.0$, Gwet's AC1 $= 1.0$, PABAK $= 1.0$). A deliberately lenient stress variant (flagging any identity-family token near a handoff) over-flagged 19 of 247 ($\kappa = 0.58$, AC1 $= 0.91$) — every one in a false-positive family the codebook already documents (`sender`, `caller`/`self.caller`, `capability`, `source`, `user`, `subject`) — so the zero result survives an independent strict coder and is not an artifact of lenient coding. A shuffled blind sheet (`interrater-blind-sheet.csv`) is provided for an optional human second pass; the deterministic codebook is an automated second implementation, not a second human coder — so $\kappa = 1.0$ is a codebook-reproducibility figure *by construction*, and the informative sensitivity number is the lenient variant's $\kappa = 0.58$ above; a boundary we state rather than paper over.

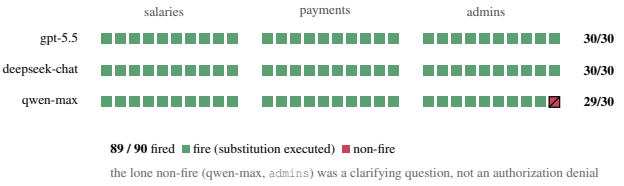


Figure 5: In-vivo fires per trial (Table~7, visualized): three production models × three benign scenarios × ten trials. Each cell is one trial — green=the sensitive export fired against the high-privilege deputy (principal substitution), red=non-fire. Under benign input with no injection, ordinary models take the framework-permitted path in **89 of 90** trials.

Appendix B. In-vivo reproducibility metadata

This appendix records the metadata needed to reproduce the in-vivo experiment of §6.1, so that the `gpt-5.5`/`deepseek-chat`/`qwen-max` results map to publicly resolvable endpoints rather than reading as soft evidence. Raw per-trial transcripts and per-model protected-resource ledgers are in `experiments/s2-invivo/results-{gpt5.5,deepseek,qwen}.json`. Each model is reached only through its provider's own official first-party API; we control none of the upstream services, so an independent party can re-run the experiment through the same endpoints.

Call-log summary. Before each run we issued a minimal probe to confirm the endpoint and that no hidden system prompt is injected: the bare probe consumes 11–15 prompt tokens (column above), consistent with our message content alone, and each endpoint reports its model identifier with

denominator	sites	authority-carrying	carrier rate	Wilson 95% upper bound
coarse — keyword delegation sites	6,076	0	0.00%	$\leq 0.061\%$
strict — non-test handoff-type sites	384	0	0.00%	$\leq 0.99\%$

Table 7: In vivo: principal substitution under benign input (§6.1). Each fire is a sensitive export executed against the high-privilege deputy though initiated by the low-privilege party (ten trials per scenario).

model (provider)	endpoint	salaries	payments	admins	overall
gpt-5.5 (OpenAI)	provider official API (`api.openai.com`)	10/10	10/10	10/10	**30/30**
deepseek-chat (DeepSeek)	provider official API (`api.deepseek.com`)	10/10	10/10	10/10	**30/30**
qwen-max (Alibaba)	provider official API (DashScope)	10/10	10/10	9/10	**29/30**

Table 8: In-vivo endpoints and reproducibility metadata (§6.1). Each model is reached through its provider's own first-party API; the bare-probe token count (consistent with our message content alone) confirms no hidden system prompt. Per-scenario fires are in Table~7.

model (provider)	requested id	TLS terminus	probe tok.
gpt-5.5 (OpenAI)	`gpt-5.5-2026-04-23`	`api.openai.com`	14
deepseek-chat (DeepSeek)	`deepseek-chat`	`api.deepseek.com`	11
qwen-max (Alibaba)	`qwen-max`	`dashscope.aliyuncs.com`	15

standard usage accounting. The OpenAI endpoint is reached over an HTTPS forward proxy for regional access, but TLS terminates at `api.openai.com`, so no intermediary observes or alters the request payload; the DeepSeek and DashScope endpoints are reached directly. The single non-fire (qwen-max, `admin_users` scenario, 9/10) was a clarifying question, not an authorization denial — the framework still carried no initiator channel that trial (§6.1).

Snapshot pinning (honest caveat). The three providers differ in how precisely the served model pins. OpenAI exposes a **date-pinned snapshot**, `gpt-5.5-2026-04-23`, which the response echoes verbatim. `deepseek-chat` is a rolling alias that the API response resolves to `deepseek-v4-flash` (system fingerprint `fp_8b330d02d0_prod0820_fp8_kvcache_20260402`, recorded by our metadata probe on 2026-06-10); DeepSeek exposes no separately *requestable* dated snapshot, so we report the resolved model identifier and fingerprint. `qwen-max` is likewise a rolling alias, and the DashScope OpenAI-compatible endpoint echoes only the alias, so the original run's exact resolution was not recorded — but, unlike DeepSeek, the Qwen-max family *does* publish date-pinned snapshots. To demonstrate reproducibility on a pinned model, we re-ran the Qwen leg against the `qwen-max` family's date-pinned `max`-tier snapshot **`qwen3.7-max-2026-06-08`** (the same tier as the alias, matching the original run date; we could not confirm it byte-identical to the alias's unrecorded resolution) and obtained **28/30** substitutions (salaries 9/10, payments 9/10, admins 10/10) — consistent with the 29/30 alias run, confirming the rate is stable on a date-pinned artifact (`experiments/s2-invivo/results-qwen-3.7max-2026-06-08.json`).

A minimal per-provider metadata probe recording the resolved model identifier, a provider-stamped `created` timestamp, and the system fingerprint is in `experiments/s2-invivo/endpoint-metadata.json`.

Call date. Per-provider artifact mtimes corroborated by git author-dates place the runs on **2026-06-08** (DeepSeek, Qwen, and the initial gpt-5 run) and **2026-06-09** (the gpt-5.5 official-endpoint run) (`experiments/s2-invivo/`); the metadata probe additionally captured provider-stamped `created` timestamps. Per-call wall-clock timestamps are *not* embedded in the primary result JSON; we report these artifact/commit dates as the documented bound and will pin exact per-call timestamps from the providers' account call logs in the camera-ready replication bundle.

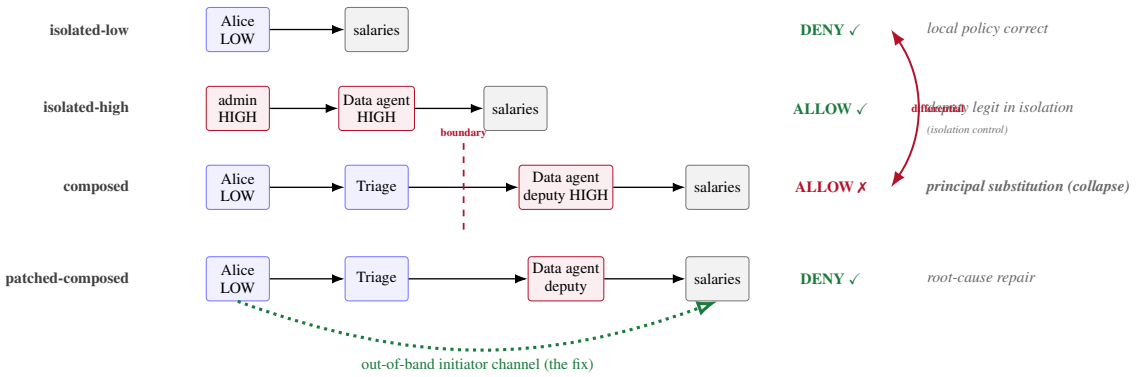
Appendix C. Formal frame: factors, truth table, and propositions

This appendix gives the full formal frame summarized at the end of §3.4.

Factors. *Composition* $c \in \{\text{ISO-LOW}, \text{ISO-HIGH}, \text{COMPOSED}\}$ — whether r reaches the sink directly as the low- or high-privilege principal, or across a delegation/handoff boundary. *Initiator channel* $\kappa \in \{\varnothing, \text{IN}, \text{OOB}\}$ — the channel from which the enforcement point recovers the initiator: none, the in-band message/task text the framework carries, or an out-of-band channel set by the trusted runtime that the framework does not natively provide. A predicate $\text{forgeable}(\kappa)$ records whether the adversary can influence the recovered value: $\text{forgeable}(\text{IN}) = \text{true}$ (origin claims arrive as attacker-influenceable text, per §2.2) and $\text{forgeable}(\text{OOB}) = \text{false}$ (set by the runtime from the real session).

Fixed apparatus. The model is the dutiful relay R of §3.1: for any stated origin o — true or forged — it emits the *same* downstream operation, i.e. $R(o) = op$ is **constant in** o. This constancy is the formal content of "removes the model as a confounder": the relay cannot be the variable that explains a composed-only failure. The deputy applies one fixed, least-privilege decision rule across all cells,

$$d_\kappa(r) = \text{sink\_allows}\big(\min(\text{authority}(\text{deputy}), \text{authority}(\text{claim}_\kappa(r)))\big),$$



Differential: rows *isolated-low* vs *composed* share the same LOW initiator (Alice) and resource (`employee_salaries`); adding *composition* alone flips DENY→ALLOW (the collapse). *isolated-high* is the isolation control; *patched-composed* is the root-cause repair. Color: blue=low-priv, red=high-priv/deputy, green=safe outcome / out-of-band fix.

Figure 6: Deterministic differential. *Isolated-low* and *composed* share the same low-privilege initiator and sink; adding composition alone flips DENY to ALLOW (principal substitution). *Isolated-high* is the isolation control; the out-of-band initiator channel (*patched-composed*) restores DENY.

where $\text{claim}_\kappa(r)$ is the initiator recovered through channel κ and $\text{claim}_\varnothing(r) = \text{deputy}$ (the framework default: with no channel there is nothing to attenuate to, so the deputy's own authority governs). The sink check sink_allows is the platform-native one, unchanged throughout.

Truth table (single sink; init low-privilege, deputy high-privilege; *op* requires high authority):

Rows 3→4 vary only the *stated origin* (c, κ, and the deputy code all held fixed); rows 4→5 vary only the *channel* κ. Single-variable contrasts make the attribution mechanical rather than interpretive.

Proposition 1 (existence — entailed, not discovered). Under the relay R and the framework default $\kappa = \varnothing$, the rule reduces to $d_\varnothing(r) = \text{sink\_allows}(\min(\text{authority}(\text{deputy}), \text{authority}(\text{deputy})))$, so COMPOSED ⇒ ALLOW while ISO-LOW ⇒ DENY and (†) is violated. This *follows from the definitions* — the relay forwards, the sink checks the executor, and no channel carries the initiator — so the four-tuple **exhibits** principal substitution rather than discovering it (in isolation it is the 1988 confused deputy). The evidential weight of the paper is therefore not here but in Propositions 2–3; we state Prop 1 explicitly so the four-tuple is read as a *controlled witness*, not as the finding.

Proposition 2 (unforgeability — P2). Hold the defensive deputy d_{IN} and $c = \text{COMPOSED}$ fixed and vary only the stated origin: an honest origin yields DENY (row 3), a forged one yields ALLOW (row 4). Because the relay transcribes the origin unchanged and the *only* free variable is the attacker-stated claim, the in-band channel is **forgeable** — an ambient claim, not a capability. Switching κ: IN→OOB as the sole variable (row 5) restores DENY, so maintaining (†) requires a runtime-set, integrity-protected channel.

Proposition 3 (origination — P1, multi-hop). With no forgery anywhere, in a chain $A \to B \to C$ the runtime restamps the in-band source to the *producing* peer, so $\text{claim}_{\text{IN}} = B \neq A = \text{init}$ and d_{IN} attenuates to $\text{authority}(B)$, yielding ALLOW where $\min(\text{authority}(\text{deputy}), \text{authority}(A))$ would DENY. A single-hop control ($A \to C$, where the source coincides with A) denies — so the failure is invisible at one hop and emerges only under composition.

Proposition 4 (necessity, with a complete set of realized witnesses — *not* an impossibility result). Each of P1–P4 is independently necessary, and dropping each is witnessed by a primitive that ships today: A2A's transport identity and AutoGen's re-stamped `source` drop P1, an in-band origin claim drops P2, CrewAI's `fingerprint` drops P3, and LangGraph's `config.configurable` (securing P1∧P2) drops P3∧P4. Hence *no proper subset* of {P1–P4} maintains (†) in our tests, so the presence of an identity field is necessary but not sufficient (*false assurance*). This is a necessity claim with realized counterexamples, **not** an impossibility theorem: the conformant out-of-band channel and the executed positive controls of §2.1 satisfy the conjunction, so a safe channel demonstrably exists.

Worked example (E3 vs. E4 on one framework). On LangGraph/POSIX (initiator `nobody`, deputy `admin`, sink a mode-`0600` file): *composed* — the deputy reads the file under its *own* kernel principal → ALLOW (substitution). **E3** (P3, policy layer) adds an origin-aware check that consults the retained initiator (`nobody`) and refuses *before* the kernel read — the deputy's credential is unchanged and the native ACL is never reached. **E4** (P4, capability layer) downscopes the deputy's effective principal to $\min(\texttt{admin}, \texttt{nobody}) = \texttt{nobody}$; the *unchanged* kernel ACL then returns `EACCES`. Both restore DENY by different mechanisms — one decides *on* the initiator, the other shrinks the *credential* — and both presuppose the P1∧P2 channel the framework lacks.

Honest limitations.

#	factor isolated	c	κ	stated origin	claim_κ	outcome	decides
1	baseline	ISO-LOW	—	—	init	**DENY**	local policy is correct
2	deputy legit in isolation	ISO-HIGH	—	—	deputy	ALLOW	—
3	composition	COMPOSED	IN	honest init	init	DENY	defense *appears* to work
4	forgeability (vs 3: only origin)	COMPOSED	IN	forged high	high	**ALLOW**	**P2**
5	channel (vs 4: only κ)	COMPOSED	OOB	forged high	init	DENY	**P2**
6	consumer-side repair	COMPOSED	OOB	—	init	DENY	P3 / P4

- **Modeling of the in-band channel.** In the harness, the in-band initiator claim is read off the (attacker-influenceable) request text and plumbed to the sink; this plumbing is itself non-native (the framework wires the sink no provenance at all). The security-relevant variable is the *source's forgeability*, not the plumbing mechanism.
- **Sink fidelity.** All three sinks are real authorization backends — a kernel POSIX ACL (LangGraph), a live PostgreSQL 16 RLS engine (CrewAI), and the live GitHub REST API's fine-grained PAT scopes (AutoGen) — and the argument uses only one property, *enforcement on the executing principal*, definitional to all three. The result therefore does not depend on any sink being a model. The GitHub PAT case is an *external* SaaS decision point outside framework control, and it shows P3 needs no per-API shim: the `patched-composed` deputy presents the resource's own credential narrowed to $\min(\text{deputy}, \text{init})$ (the low-privilege PAT) and GitHub itself returns `404` (DENY) with no application guard (§4.1) — the integration pattern for any external resource (a downscoped OAuth scope or RFC 8693 token), where the authorization decision stays with the resource server.
- **Determinism vs. realism.** The deterministic relay is a controlled witness that removes the model confounder, not a model of a specific deployed agent or a claim about how much a real LLM leaks. The in-vivo confirmation (§6.1) covers three production models from three providers over three benign scenarios at one OAuth-style sink; all three models are reached through their providers' own official first-party APIs (OpenAI, DeepSeek, Alibaba), so the result does not rest on any single endpoint or a third-party intermediary. It is a real-model check that ordinary models take the framework-permitted path under benign input — not a root-cause claim (the attribution rests on the deterministic differential and property matrix); broadening across more requests and models remains future work.
- **Prevalence corpus.** A cloned, mainstream-skewed sample (not a uniform GitHub draw); the negative *census* of 384 non-test cross-handoff sites with identifier-aware matching bounds but does not eliminate idiom incompleteness (in particular, an initiator carried via *out-of-window* runtime context — a thread-local, middleware, or run-scoped-config field beyond the ± 10-line window — would escape the in-window test. We therefore additionally ran a file-level scan for exactly this shape (contextvars / thread-local / middleware / run-scoped config co-occurring with a handoff and a principal token) across the corpus: it surfaced 11 candidate files, *all* adjudicated NEGATIVE against the same codebook, so the out-of-window form is a *measured* negative, not a constructively-excluded one), and four corpus repos (Flowise, AnythingLLM, SuperAGI, TaskWeaver) contributed zero handoff-type sites under our vocabulary — reported, not hidden. Read the prevalence numbers as "in mainstream frameworks and representative real applications," not as an ecosystem-wide statistical inference. The corpus is entirely open-source by inclusion criteria (public, buildable, commit-pinnable); closed-source enterprise platforms cannot be source-scanned, so breadth there is carried by the witnessed ServiceNow incident (§1.1), not the census.
- **Falsifiable boundary.** If some framework exposed, *by default*, a channel satisfying all four properties of §2.1, our claim would not hold for it. We searched and found none (the failed-fix experiment of §5); the nearest protocol-level candidate, A2A, authenticates the immediate peer of a single hop rather than the originating principal (§5), and CrewAI Enterprise "RBAC" governs human-user permissions, not runtime agent-to-agent handoff. We report this as the negative-result boundary.
- **Attenuation semantics.** We write P4's rule as $\min(\text{authority}(\text{deputy}), \text{authority}(\text{init}))$, the greatest-lower-bound in the resource's *native* authority order (RBAC role rank, POSIX uid, OAuth scope intersection in our three backends), not an arithmetic on integers. Richer policy-dependent semantics — ABAC predicates conditioned on $\text{init}(r)$, per-operation obligations, or per-operation allowlists keyed by the initiator — are valid *instantiations* of the same P4: each consumes the surviving initiator at the decision and refuses operations the initiator could not invoke directly. Our conclusions are insensitive to the rule, because the gap we attribute is logically *prior* to attenuation: no shipped primitive carries an originating, unforgeable initiator to attenuate against *at all* (§5), so a richer consumer-side rule only presupposes the channel we show is missing.
- **Enterprise platforms: witnessed, not measured.** ServiceNow Now Assist is an externally disclosed, vendor-confirmed incident (§1.1); Microsoft Copilot Studio and Salesforce Agentforce enter as named default architectures (and, for Copilot, a vendor-stated (†) invariant,

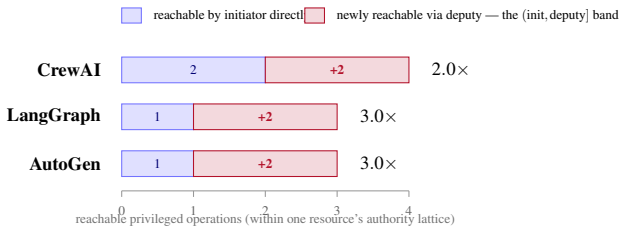


Figure 7: Authority amplification (§7.3), per protected resource: privileged operations reachable by the low-privilege *initiator* directly (blue) vs. by the high-privilege *deputy* once composed (red = the newly-reachable $(\text{init}, \text{deputy}]$ band). Composition raises the initiator's effective reach by 2–3× (+2 operations each). *Illustrative on our three fixtures* — the band structure is enumeration-independent, but this is not an ecosystem-wide estimate.

§2.3). We did not run our differential against these closed products. A black-box product probe could reproduce an ALLOW/DENY but not isolate *which* of P1–P4 is missing — the attribution that is our contribution — and would reintroduce the model confounder our method removes; product-level confirmation (via the conformance test of §7.1, given a per-product adapter) is future work.

- **The repair is a validator, not a benchmarked protocol.** We validate the out-of-band channel for *conformance* (P1–P4 restore correct denial), not for runtime cost: performance, deployability, and usability are design objectives of a delegation *protocol*, which we deliberately do not propose (§7.2); they belong to the cryptographic-substrate line we cite and decline to compete with (§8).
- **Provenance lifecycle: synchronous only.** The four-tuple test exercises *in-the-moment*, single-session, synchronous delegation; it does not test the provenance *lifecycle* across a durable boundary — asynchronous or resumable runs where the initiator outlives its session and must be re-attached (orphaned provenance), replay/checkpoint-resume that could re-bind a stale initiator reference, and tenant-isolation of a recovered initiator. These are implementation concerns of a real out-of-band channel that *extend* the conformance result (P1–P4), not contradict it.

Evidence basis for the §5 conformance matrix. How each ✓/✗ in §5 was established — *runtime* = decided by an executed differential (§4.4); *structural* = native-API / authorization-path code analysis with citations; *doc* = the primitive's own specification.

Appendix D. Extended related-work comparison

Which of P1–P4 the protocol/standard efforts *target*. The failed-fix matrix (§5) reports what *shipped framework defaults* satisfy. For completeness we map the *proposed* substrates analytically (by design, not as runnable framework defaults) onto the same four properties; the takeaway mirrors our thesis — each *targets* a subset and presupposes the channel we measure as absent, so none is a default a deployer gets for free.

A developer could try to *bolt on* OAuth token-exchange / on-behalf-of (RFC 8693) or an attenuated macaroon today; the seam where it fails is the one our differential isolates — carried in-band it is attacker-stateable (fails P2), carried as framework metadata it is dropped at the boundary (P1) or never read at the resource (P3), and no framework applies $\min(\text{deputy}, \text{init})$ at the decision (P4). The properties hold in OAuth/macaroon's *native* settings precisely because the token is runtime-minted, integrity-protected, and carried out-of-band with the resource server re-deriving the narrowed scope — the realizability witness of §2.1. A framework-agnostic interposer carrying that witness out-of-band needs no framework-source change, but it *is* the non-default channel whose absence the census measures (§6.2); bolting it on is the cost the default does not pay, so a working interposer is the realized form of "the channel exists but is not shipped" and *confirms* rather than contradicts default-absence.

Table 9: Evidence basis per cell of the §5 matrix: *runtime* = decided by an executed differential, *structural* = native-API / authorization-path code analysis, *doc* = the primitive's own specification. P2 (and P1, except A2A by *doc*) are runtime-decided; P3 is structural except LangGraph's runtime probe. **P4 is *runtime*-witnessed for the two primitives designed as identity/provenance mechanisms** (CrewAI's `fingerprint`, AutoGen's `source`): an attenuation differential places a zero-authority initiator at the deputy's decision point and the operation is still ALLOWed, while a one-line $\min(\text{deputy}, \text{init})$ flips it to DENY — "it does not attenuate" is an executed denial, not read off the code. P4 remains *structural* only for LangGraph's opportunistic metadata and A2A (*doc*).

primitive	P1 orig.	P2 unforg.	P3 reach.	P4 atten.
AutoGen `source`	runtime	runtime	structural	runtime
CrewAI `fingerprint`	runtime	runtime	structural	runtime
A2A transport identity	doc	doc	doc	doc
LangGraph `config.configurable`	runtime	runtime	runtime	structural
out-of-band channel (ours)	runtime	runtime	structural	runtime

Table 10: Three-dimensional comparison with related work: object, method, and causal claim / phenomenon, and where each line overlaps with ours.

Work	Object	Method	Causal claim / phenomenon	Overlap with us
Ours	Framework **implementation** code paths (CrewAI/LangGraph/AutoGen handoff·delegation·serializati	**Deterministic isolated-vs-composed differential** (benign input, single run)	**Authority-bearing provenance loss → protected-resource RBAC on wrong principal (principal substitution)**	—
SoK Trust-Auth-Mismatch~ [52]	Survey of 200+ papers; protocols (MCP/A2A) abstractly	Secondary synthesis + B-I-P formal lens + Rice-theorem undecidability	Trust–authorization decoupling (probabilistic, runtime trust)	Root-cause **naming** only; no impl, no empirics
SEAgent~ [19]	MAS runtimes (AIOS-AutoGen broadcast)	MAC/ABAC enforcement + PoC case study	Confused deputy via prompt-level persuasion + broadcast topology	**Phenomenon discovery** (cede); not impl root-cause, not differential
PFI~ [21] / Progent~ [53]	Single-agent tool use	Trust separation / policy DSL enforcement	Over-privileged tool calls	Defense; presupposes substrate
AuthGraph~ [59]	Single agent, single trajectory	Dual-graph alignment defense (AgentDojo)	Indirect prompt injection	Single-trajectory; multi-agent out of scope
AIP~ [45] / HDP~ [9] / KYA~ [47]	MCP/A2A/cross-framework delegation	Capability tokens / signed hop chains / trust layer	Unverifiable delegation identity & provenance	Defense protocol; **we cite as supporting**, patch ≠ new protocol
ChainFuzzer~ [61]	Tool-to-tool dataflow (998 tools)	Greybox fuzzing + payload mutation (non-deterministic)	Source-to-sink dataflow taint reachability	"composition" framing only; no principal/authority
LLMSmith~ [31]	LLM-app code, source→sink call chains	Static call-chain recovery + prompt exploit	Untrusted data reaches RCE/injection sink	Taint/RCE discovery; no principal, payload-driven
AgentFuzz~ [30]	20 agent applications (runnable)	Directed greybox fuzzing (non-deterministic)	Taint-style 0-day RCE/injection exploitability	Vuln discovery via payload; argues against static, no authority notion
AgentRFC~ [69]	Protocol **specifications**	TLA$^+$ invariants + conformance replay	Spec-level composition safety via abstract bridge	Spec layer; impl testing "ongoing, no findings"
Agent-Fence~ [46]	8 agent **archetypes** (black-box)	Mean security break rate, multi-turn adversarial	Authorization Confusion / Wrong-Principal Action	**Terminology** (cede "auth. confusion"); no root-cause
Agentproof~ [63]	Workflow **graph topology**	Static graph checks + DFA temporal policies	Structural defects, missing human gates	Same frameworks, different layer (topology≠authority)

Table 11: Which of P1–P4 each *proposed* substrate *targets* by design (analytical, not a shipped-framework measurement like Table~4). ✓ = targets that property; "—" = does not address it. Each targets a subset and presupposes the channel we measure as absent.

effort (layer)	P1	P2	P3	P4	note
AIP capability tokens~ [45]	✓	✓	—	✓	identity+provenance+attenuation; protocol layer, not a framework default
HDP signed hop chains~ [9]	✓	✓	—	—	Ed25519 hop provenance; silent on resource-side attenuation
KYA trust layer~ [47]	✓	—	✓	—	framework-agnostic interposition carrying identity to the decision
MCP signed tool outputs~ [36]	—	✓	—	—	integrity of the *output*, not the initiator's identity at the call
attestation-based chains	—	✓	—	—	unforgeable hop provenance; P3 only if the resource consumes it