

Agent Name Collision Attacks in Multi-Agent Systems

Adithyan Arun Kumar
Independent Security Researcher
adioffsec@gmail.com

September 2026

Abstract

Multi-agent hosts turn remote Agent Cards into local agents, tools, workflow targets, and broker routes. A2A defines the card’s **name** as human-readable metadata, not as a stable identity, and specifies no collision semantics. The security failure begins when a host nevertheless uses that remote name as a local routing identifier. We traced registration through dispatch and ran isolated regression tests at seven pinned open-source revisions. Six client-style integrations selected an attacker-controlled peer’s client or loopback endpoint for a request addressed to a trusted peer’s name. A seventh, brokered implementation collapsed both peers onto one name-derived route; queue and access-control state determine whether the result is interception or denial. The common result is wrong-peer dispatch, not universal privilege inheritance. Synthetic credential and tool tests found no A-specific credential transfer in the tested client bindings and no direct transfer of A-owned tools. The broker path forwards a caller-configuration object; delegated identity or tokens reach B only if present and B can consume the route. Two other paths expose a later, model-mediated decision rather than direct execution authority. The necessary conditions assign different responsibilities to the protocol, implementations, and deployments. Hosts should route by an origin-bound stable identity, keep names presentational, and reject ambiguous aliases. The evidence establishes a recurring implementation vulnerability class, not a universal A2A protocol exploit or a count of vulnerable deployments.

1 Introduction

Agent discovery converts a remote description into a local object that can receive tasks. The host starts with an endpoint, registry record, authenticated subject, or broker principal and then retrieves an Agent Card. A secure design keeps that admitted identity bound to the transport used at dispatch. The failure begins when a host takes a peer-controlled display name, uses it as a local agent, tool, workflow, registry, or broker selector, accepts a collision, and silently chooses one peer by order or shared route. This resolves a name to a resource outside the intended control sphere, consistent with CWE-706[14].

The instruction “send this task to **payments**” can appear to preserve the operator’s choice even after the resolver has rebound **payments** to another principal. The attacker does not need to forge A’s transport, signing key, or credentials. It needs control of an admitted peer B’s display name and a resolver that lets B win the collision.

We evaluated 7 open-source implementations maintained across 6 organizations at exact revisions. The corpus spans agent-tree lookup, generated function tools, client maps, workflow registries, and broker topics. It is purposive rather than statistically representative. The result is breadth across independently implemented sinks, not an estimate of deployment prevalence. The affected source

paths are pinned in the repositories for Google ADK Python and TypeScript, UiPath LangChain, BeeAI Framework, Solace Agent Mesh, Mozilla Any-Agent, and AutoDev[9, 10, 23, 6, 20, 15, 5]. In controlled tests, 6 client-style integrations selected B’s client or loopback endpoint for an A-selected request. The brokered target produced 1 name-derived route collision; its final delivery outcome depends on broker configuration. These results establish wrong-peer dispatch, not universal credential theft or code execution.

This work contributes:

1. a precise identity invariant and a conjunctive threat model that distinguish an exploitable collision from a harmless duplicate;
2. registration-to-dispatch traces and controlled tests across seven implementation paths, including a separate authority-transfer analysis;
3. an evidence-calibrated division between protocol semantics, implementation behavior, and deployment admission; and
4. a fail-closed design in which routing uses a stable identity bound to enrollment origin while names remain presentation metadata.

The evidence supports a recurring implementation vulnerability enabled by an identity-semantics gap in A2A. It does not show a break in wire framing, automatic enrollment, vulnerability in every A2A SDK, or seven remotely exploitable production deployments.

2 System and Threat Model

2.1 Identity layers

Each admitted peer has a stable identity bound to its admitted origin or transport. A remotely supplied card name is presentation metadata; it must not replace or modify that binding when the host resolves a request.

A collision becomes a vulnerability when the host instead uses the card name as an authority-bearing selector, permits two distinct peers to share it, and then uses the ambiguous value to select a transport or execution object. For a trusted A and attacker-controlled B with the same name, the selected object can depend on registration order, list order, normalization, or broker binding state rather than the identity selected at admission.

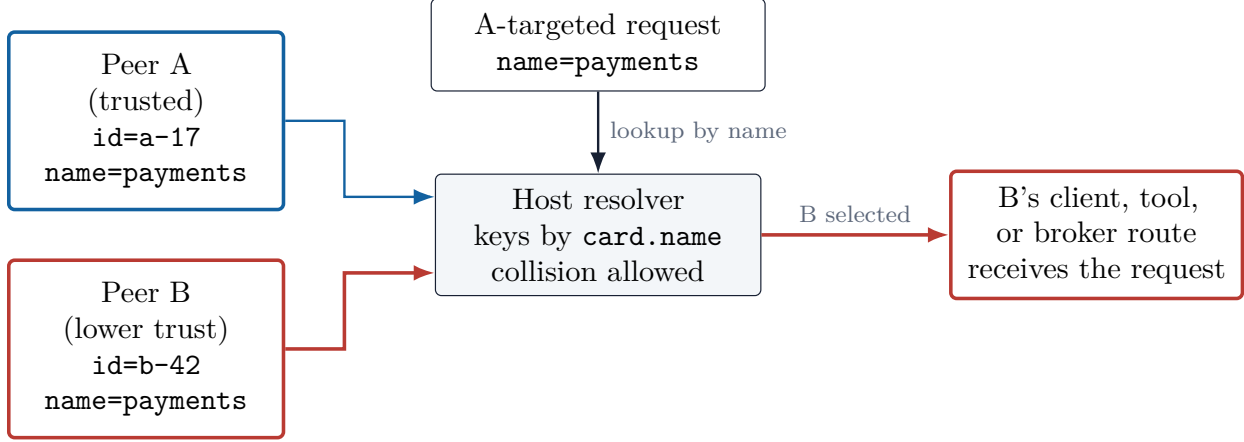


Figure 1: A name collision breaks the binding between the admitted peer and the selected route. The resolver accepts both cards under one display name and sends A’s request to B.

The selector can be a dictionary key, a tree lookup, a generated function-tool name, a workflow node, a request topic, or a queue identity. The data structure is incidental. The invariant fails at the point where presentation metadata determines which principal receives a task.

2.2 Attacker capabilities and necessary conditions

A confidentiality or integrity attack requires all of the following:

1. **Shared routing domain.** A and B are both admitted to the same host, registry, tool set, workflow, or broker namespace. The name collision does not itself enroll B.
2. **Name control.** B can publish or refresh its own card, or an attacker has compromised an admitted B and can change its card name.
3. **Collision precedence.** B is placed first in a first-match resolver, registered last in a replacement map, normalized into A’s tool identifier, or able to bind or consume the broker route.
4. **Name-derived use.** A caller, workflow, or model later selects the ambiguous local name. A client that sends directly to a fixed endpoint has no collision surface.
5. **Trust differential.** A may receive data or exercise authority that B may not. If A and B are fully interchangeable for every routed task, the impact can shrink to correctness or availability.

The attacker is inside the agent admission boundary but remains outside A and the host. This model covers registries, marketplaces, partner integrations, multi-account orchestration, and brokered meshes. It does not cover an unauthenticated Internet attacker.

2.3 Security outcomes

We separate four outcomes that are often collapsed in vulnerability descriptions:

Message diversion

B receives task or caller context intended for A and may respond on the selected route.

Credential transfer

A-specific secret material appears at B’s request boundary.

Authority transfer

B receives a delegated identity or token, an A-owned tool object, or an equivalent direct capability.

Secondary execution

B’s response influences a later model or policy decision that may invoke a host tool.

Only the first follows directly from client-side wrong-peer dispatch. The remaining outcomes depend on what the host attaches after routing and what decisions occur after B responds.

3 Methodology

3.1 Inclusion criteria

We retained a public-source implementation only when all six conditions held:

1. normal framework or product code accepts multiple remote agents or cards;
2. the peers remain distinguishable by endpoint, resource ID, registry record, or equivalent input;
3. a remote card name becomes an authority-bearing local selector;
4. exact or normalization-equivalent duplicates do not fail closed;
5. a later production path resolves through that selector; and
6. a focused test or complete source trace reaches the selected client, endpoint, tool, topic, or queue.

A dictionary assignment or code-search hit was insufficient. Copied samples were not counted as independent implementations. These criteria retained 7 implementations across 6 maintainer organizations. Table 1 gives the exact revisions and evidence types.

3.2 Controlled experiments

The client-style tests created two distinct synthetic or loopback peers. Trusted A and lower-trust B advertised the same target name; the test then invoked the framework’s selected agent or tool and recorded which client or endpoint received the request. Controls differed by target. AutoDev changed B’s remote card name; Any-Agent assigned B a distinct local tool name while both cards retained the same name; and the Google tests checked credential or callback binding with the collision intact. The preserved UiPath and BeeAI tests did not include matched distinct-name controls.

The first-party sample test did not run a model. It compiled the pinned source paths and replaced only network and SDK dependencies with inert in-process doubles, isolating name-to-route behavior from probabilistic model selection. The test was replayed on 9 September 2026. Both affected paths selected B, both one-variable controls selected A, and duplicate rejection stopped dispatch.

Solace Agent Mesh required a different test because its authority-bearing identity spans a registry, topic, and queue. The test established registry replacement and production publication to the shared name-derived route. We did not simulate a production broker topology, so final consumption by B remains conditional.

3.3 Tests for stronger consequences

After proving route substitution, we tested stronger consequences separately:

- an A-only synthetic transport credential appearing at B’s request boundary;
- an A-owned in-process tool becoming directly callable by B;
- a privileged action occurring without a new model or policy decision; and
- B-controlled output reaching a model that also has a host-only tool.

The fourth test confirms only that the data can reach a later decision point. Scripted models made the dataflow deterministic; their behavior does not measure exploit reliability against a production model.

3.4 Evidence integrity and safety

All repository revisions, test patches, source locators, and preserved results were hashed. On 9 September 2026, the SHA-256 manifests for the evidence bundle, pinned source, seven reproducer patches, and seven authority tests all passed. Runtime traffic was limited to recording doubles and study-controlled loopback peers. No third-party deployment, account, agent, or production data was contacted.

4 Cross-Implementation Results

4.1 Protocol boundary

At the pinned A2A revision, `AgentCard.name` is required and described as a human-readable name. The card does not carry a stable agent identifier, and the discovery guidance recognizes curated registries while leaving their API and identity model outside the specification[1]. The protocol neither requires name uniqueness nor instructs a host to build `routes[card.name]`. This omission permits the vulnerable mapping but does not mandate it.

4.2 Affected paths

Table 1 summarizes the seven affected paths. The tests exercise production classes with recording clients or real loopback HTTP endpoints. The Solace test instead runs the production publish path and records the broker destination before delivery. Source anchors are pinned at the exact revisions[9, 10, 23, 6, 20, 15, 5]. The Any-Agent composition used OpenAI Agents SDK 0.20.0 under its default `warn` collision policy, which retains the last colliding function tool[16].

Table 1: Affected collision-to-dispatch paths at pinned revisions. “First” and “last” describe resolver precedence, not attacker privilege.

Implementation	Name becomes	Resolution	Observed consequence
Google ADK Python 2.8.0	local agent name via Agent Registry helper	warning; first match	selected B agent’s A2A client called
Google ADK TypeScript 2.0.0	normalized local agent name via registry helper	first match	selected B agent’s <code>sendMessage</code> called
UiPath LangChain 0.17.3	generated A2A function-tool name	later replacement	surviving tool used B resource transport
BeeAI Framework 0.1.83	handoff tool and requirement selector	first match	client created from B card URL and sent
Solace Agent Mesh 1.28.8	card registry key, request topic, and queue identity	update/shared route	publish used collided route; consumption conditional
Mozilla Any-Agent 1.18.0	<code>call_{card-name}</code> function tool	later tool retained	JSON-RPC request reached only B loopback endpoint
AutoDev 2.4.1 lineage	card and client map key	later replacement	JSON-RPC request reached only B loopback endpoint

In all six client-style integrations, dispatch selected B’s bound client or endpoint while the legitimate A binding was not invoked. Solace instead collapsed A and B onto one broker address. The source proves publication to that shared route; queue access type, ACLs, and binding state decide whether B intercepts, becomes an active or standby consumer, or causes denial.

4.3 Four implementation forms

The targets reach the same invariant violation through four constructions:

1. **Agent-tree ambiguity.** Google ADK retains duplicate local names and returns the first matching sub-agent.
2. **Tool identity collapse.** UiPath, BeeAI, and Any-Agent derive tool or handoff identifiers from card names; the graph or backend then chooses one duplicate.
3. **Client-map replacement.** AutoDev and multiple first-party samples rekey configured endpoints by card name, so a later peer replaces both card and transport.
4. **Broker identity collapse.** Solace uses the name across discovery, registry lookup, request topics, and queue identity.

These forms show why “the host uses a dictionary” is too narrow. The security property is the binding from admitted identity to final execution target, regardless of storage structure.

4.4 Google ADK reachability boundary

The Google tests use two real remote-agent objects with duplicate local names and prove that the agent tree warns or accepts, resolves one duplicate, and invokes the selected B client. They do not, by themselves, prove remote control of the local name. In Python, direct `RemoteA2aAgent` construction requires a caller-supplied `name` documented as unique; TypeScript similarly receives the

local name through `RemoteA2AAgentConfig`. The affected Agent Registry helpers independently derive that local name from embedded Agent Card content[9, 10].

The tests confirm unsafe duplicate resolution and wrong selected-client dispatch. Both registry integrations also contain a production card-to-name path. This study did not demonstrate that a low-privilege agent can introduce or refresh the required record in a deployed Google Cloud project. Direct-constructor misuse may be invalid operator configuration rather than a remote attack. The Google ADK result is therefore a vulnerable routing primitive with conditional attacker reachability, not an end-to-end remotely exploitable zero-day.

5 Impact and Authority Boundary

5.1 What follows from wrong-peer dispatch

When the necessary conditions in Section 2 hold, B receives a request that the host selected under A’s name. The request can include prompts, document fragments, task metadata, caller context, or conversation state. B can also supply a response on the selected path. That is a confidentiality and response-integrity failure whenever A and B have different authorization for the content, even if no credential crosses the boundary. The controlled tests record the transport or endpoint that received each request. Whether B can enter that routing domain, and whether the diverted data crosses a meaningful trust boundary, remains deployment-specific.

5.2 Credentials and delegated context

The collision usually selected B’s complete client, agent, or tool closure rather than merging A’s configuration into B. Across the tested ADK Python, ADK TypeScript, BeeAI, Any-Agent, and AutoDev bindings, an A-only transport credential did not appear at B. The observed UiPath bearer was shared host authentication at the managed proxy client boundary; it was not A-specific, and downstream delivery to B was not shown. The number of demonstrated direct A-specific credential transfers in these tested bindings was 0. The shared negative result does not imply one retention mechanism. BeeAI’s handoff clone rebuilt the selected agent without its original HTTP parameters, while the tested AutoDev path exposed no per-agent credential setting[6, 5].

Solace differs from the client-style cases. Its publish path places the entire `a2aUserConfig` on the request sent to the name-derived route[20]. That structure may contain caller identity, scopes, or delegated tokens. B receives that authority only if the deployment populated it and B can consume the collided route. Exclusive and non-exclusive queue modes change the delivery behavior[19]. We therefore report 1 conditional delegated-context case, not a universal token leak.

5.3 Tools and execution

No tested target transferred an A-owned in-process tool object to B or deterministically executed a privileged A or host action solely because of the name collision. UiPath and BeeAI did demonstrate 2 second-stage dataflows: B-controlled output became model input in a context where a host-only tool was available, and a scripted model then chose that tool. This validates reachability to a new decision point. It is a compound prompt-injection path, not direct authority inheritance, and it does not measure production model reliability.

Table 2: Evidence boundary after the routing decision.

Outcome	Status	Interpretation
A-intended request reaches B	Proven in six client paths	Direct confidentiality and response-integrity impact when trust differs
Name-derived broker route	Proven; delivery conditional	Interception, standby, or denial depends on broker controls
A-specific credential reaches B	Not observed	No A-specific transfer was observed; credential behavior differed by target
Delegated caller context	One conditional path	Solace forwards configuration, but token population and B consumption are required
A-owned tool or direct privileged action	Not observed	The collision alone does not grant an in-process capability
B output reaches a later model/tool decision	Plumbing proven in two paths	Separate model, policy, and authorization decisions determine final action

5.4 Severity

No single severity score accurately represents every composition. If B can join a host that routes sensitive A-selected content by name, wrong-peer disclosure and response substitution can be high impact. If every peer is mutually trusted for every task, the impact may be correctness or availability. Credential and execution impact depends on attachments and sinks after routing. A report should therefore score the concrete deployment path, not the abstract existence of a duplicate name.

6 Ecosystem Propagation and Counterexamples

6.1 Reference-code propagation

The first-party A2A samples contained at least 6 paths that fetched cards from configured addresses, stored connection objects under `card.name`, and later dispatched by that key[2]. A bounded GitHub code search for the exact assignment returned the first 100 indexed file matches across 83 distinct public repositories. That number measures propagation, not prevalence. The set includes forks, tutorials, workshops, notebooks, and descendants of the same sample. It is neither a count of independent implementations nor a count of live deployments. It shows that the unsafe identity pattern entered reusable ecosystem reference code.

6.2 Negative controls

We also reviewed 7 paths that did not exhibit the exact wrong-peer substitution behavior. Their controls were not identical, but each prevented a duplicate display name from silently choosing another transport.

Table 3: Negative-control designs on inspected paths.

Implementation	Relevant control
Microsoft Semantic Kernel .NET	duplicate function insertion fails closed[13]
Microsoft Agent Framework Python	duplicate workflow executor identifiers are rejected[12]
Google ADK Java	remote agents use explicit local names; duplicate sub-agent names are rejected[8]
Google ADK Go	remote agents use explicit local names; the inspected multi-agent loader rejects duplicates[7]
Strands Agents SDK TypeScript	exact and normalized-equivalent tool names are rejected[21]
python-a2a	later collisions receive distinct suffixed aliases[18]
Agno	routing uses an operator-controlled agent identifier rather than the card display name[4]

These designs are not equally strong. Suffixing prevents silent replacement but can create unstable aliases; a stable authenticated identifier is preferable. Collectively, they establish that A2A compatibility does not require wrong-peer dispatch and that implementation-level remediation is practical.

7 Protocol, Implementation, and Deployment Responsibility

A2A supplies a required human-readable card name but no stable agent identifier or collision semantics at the pinned revision[1]. That is a protocol-level identity gap: interoperable discovery lacks a standard field and rule for the identity that should survive registration, refresh, selection, and dispatch. The name collision is not a wire-protocol attack. The protocol does not require hosts to key routes by `AgentCard.name`, and safe conforming implementations exist. Each affected framework owns the point where it discards an endpoint or registry distinction, accepts ambiguity, and dispatches through a fail-open resolver. Deployments separately control who may enter that resolver’s domain.

Table 4: Responsibility follows the layer that controls each condition.

Layer	Observed gap	Action
A2A protocol	no interoperable stable identity or duplicate-name semantics	declare names non-authoritative; define or profile an origin-bound stable identifier and refresh rules
Host/framework	display name promoted to route, tool, workflow, topic, or queue identity	preserve enrolled identity; reject ambiguity; test the final dispatch binding
Registry/deployer	attacker-controlled B may enter A’s routing domain	authenticate publishers; scope registration and refresh; separate broker namespaces and ACLs
Credential/policy layer	authority may be attached after an ambiguous route resolves	resolve authenticated identity before attaching credentials, scopes, or privileged sinks

Calling the issue only a deployer misconfiguration ignores framework code that silently replaces a stable distinction with remote text. Calling it a universal protocol vulnerability ignores implemen-

tations that never perform that replacement. It is a protocol-enabled implementation vulnerability class whose reachability and impact depend on the deployment.

8 Defenses

In the first-party sample test, the one-variable controls restored dispatch to A when B no longer collided, and duplicate rejection stopped dispatch before either peer received the request. The durable fix is to preserve identity rather than manage winner order.

Hosts, frameworks, and registries should apply the following controls:

1. **Route by an enrolled identifier.** Preserve the operator- or registry-controlled ID associated with the configured endpoint. Do not rekey the object after card retrieval.
2. **Bind identity to origin.** Associate the ID with the configured endpoint, authenticated registry subject, workload identity, or expected signing key. A stable but self-asserted ID can still be stolen.
3. **Keep names presentational.** Names may appear in user interfaces and model context, but workflow edges, tool registries, authorization, and broker topics should resolve through the stable ID.
4. **Fail closed on aliases.** If compatibility requires a name-only API, reject exact and normalization-equivalent duplicates. Never choose a peer by insertion or list order. Return all candidates only if a trusted caller can disambiguate by stable ID.
5. **Attach authority after identity resolution.** Select credentials and delegated scopes from the authenticated identity, not a card-derived name. Refuse a refresh that unexpectedly changes the bound identity.
6. **Carry identity into the broker.** Topic, queue, and consumer identity must use the stable principal. An in-memory stable registry does not help if two peers can bind the same name-derived broker route.
7. **Test the invariant.** Register two distinct origins or resources with identical and normalization-equivalent names. Registration must fail clearly, or stable-ID dispatch must remain bound to the intended endpoint regardless of order.

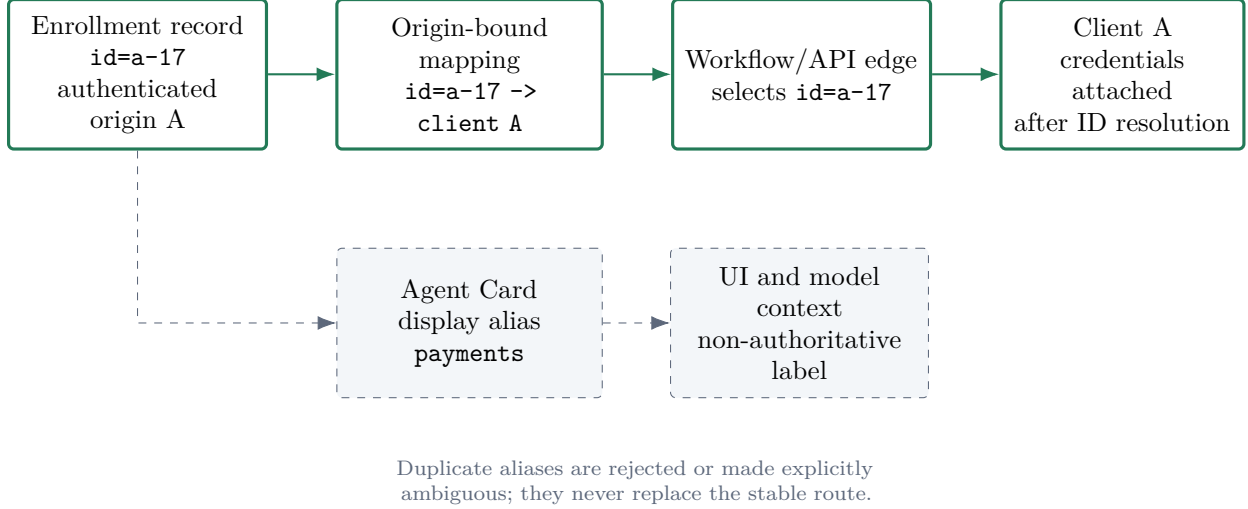


Figure 2: Safe routing keeps the enrollment identity on the authority path and moves display names outside it.

A warning is not a security control if the ambiguous graph remains usable. Similarly, reversing insertion order only changes the winner. Remediation is complete when an untrusted card field can no longer select another principal’s transport or authority.

9 Limitations and Ethical Scope

The attack model begins after B is admitted or compromised. We did not establish a universal enrollment bypass. Actual exploitability depends on registry permissions, discovery topology, configuration ownership, and broker ACLs, as well as the name collision itself.

Google ADK requires special caution. The focused tests manually supplied duplicate local names, while the Agent Registry helpers provide the source-established card-to-name path. This study did not test a Google Cloud project or prove that a low-privilege agent can create or refresh the needed registry record. That missing admission edge prevents an end-to-end remote-exploit claim for Google ADK.

Solace final delivery was not exercised in a production broker topology. The source and focused test establish registry replacement, route derivation, and publication of caller configuration to the collided route. Exclusive versus non-exclusive queues, bind order, and ACLs determine whether B receives the message.

The authority study used synthetic credentials and harmless action counters. It did not demonstrate production secrets, business actions, or one universal severity. Scripted-model tests prove that B’s output can reach another decision point; they do not estimate whether a real model, prompt, approval rule, or guardrail would invoke a privileged tool.

The seven-target corpus was purposive. Package popularity and repository search were used to find and contextualize implementations, not to estimate a population rate. The 83-repository exact-pattern snapshot cannot be converted into vulnerable products, exposed installations, or independent codebases.

The prior-art review covered public, indexed material available by 9 September 2026. Private reports, renamed issues, and unindexed content may exist. We found no earlier public description of the complete mechanism within those bounds; this is not a claim of novelty or priority.

All runtime traffic was confined to in-process doubles or study-controlled loopback peers. No third-party endpoint or account was probed. Results report exact pinned revisions rather than present-tense vulnerability status.

10 Directly Related Work

The closest protocol work is A2A issue 1014, which states that `agent_card.name` is not guaranteed unique and is vulnerable to branding changes, and proposes a persistent identifier[11]. That work identifies the missing primitive. It does not document the complete path from duplicate display name through a fail-open host resolver to deterministic wrong-peer dispatch.

LevelBlue’s Agent-in-the-Middle experiment shows that a compromised peer can exaggerate Agent Card capabilities so a model selects it for tasks[22]. The attacker capability and confidentiality consequence are adjacent, but the mechanism is model-mediated description steering. The collision studied here instead corrupts a deterministic local identity binding after peers are admitted and can be tested without model choice.

Palo Alto Networks’ A2A security guidance discusses agent impersonation and Agent Card shadowing, including similar names, capabilities, identities, and endpoints[17]. That establishes broad shadowing and impersonation as prior art. The contribution here is narrower: cross-implementation evidence for exact or normalized name collision entering an authority-bearing resolver and reaching the wrong transport, tool, or broker route.

CWE-706 provides the closest general software-weakness category: a name or reference resolves to a resource outside the intended control sphere[14]. Agent-name collision applies that pattern to multi-agent discovery, where a remote presentation field displaces an enrolled principal at dispatch.

These works rule out broad novelty claims about malicious Agent Cards, agent impersonation, name collisions, or task capture. The bounded search found no earlier public description of the complete chain from duplicate name, through fail-open resolution, to wrong-peer dispatch.

11 Conclusion

Agent-name collision is a security vulnerability when a host discards a stable admission distinction, promotes remote presentation metadata into routing identity, and silently resolves an ambiguity to another principal. Seven independently maintained implementation paths reached that failure through agent lookup, tool registration, client maps, and broker naming.

Responsibility spans three layers. The protocol leaves an identity-semantics gap, affected frameworks implement the unsafe resolver, and deployments determine who can enter the routing domain and what authority follows the selected route. Framework behavior makes this more than an intentional operator misconfiguration, while safe conforming implementations rule out a universal A2A protocol exploit.

The proven common impact is wrong-peer dispatch. Stronger outcomes require separate evidence. The tested client bindings did not transfer A-specific credentials or A-owned tools, while one broker path conditionally forwarded delegated caller context and two framework paths exposed a separate model-mediated chain. Remote presentation metadata must not select the security principal or transport binding. Preserving an origin-bound stable identity, rejecting ambiguous aliases, and attaching credentials only after identity resolution removes the measured failure without constraining human-readable names.

A Reproducibility Record

The study pinned one revision per tested path and retained the patches, exact commands, result transcripts, and SHA-256 manifests. Table 5 records the principal executions; Table 1 and the bibliography provide the source locations. The public artifact repository contains the manuscript source, normalized results, pinned source maps, focused reproducer patches, recorded outputs, and an archive-integrity verifier[3].

Table 5: Focused test record.

Target	Evidence type	Recorded result
Google ADK Python	production agents and recording clients	two tests passed; duplicate warning retained
Google ADK TypeScript	production agent tree and A2A clients	two tests passed
UiPath LangChain	production tool construction and graph node	three tests passed; one warning
BeeAI Framework	A2A agent, handoff tool, and scripted model	one focused test passed
Solace Agent Mesh	production registry and publish method	two tests passed; broker delivery not asserted
Mozilla Any-Agent	two loopback A2A peers; OpenAI Agents SDK 0.20.0 (warn)	collision and local tool-name override tests passed; two warnings
AutoDev	production Kotlin consumer with loopback HTTP peers	collision and distinct-name controls passed
First-party samples	pinned source paths with inert doubles	two affected paths, two distinct-name controls, and duplicate rejection passed

All six client-style collision tests recorded selection of B’s client or endpoint. Controls differed by target: AutoDev changed B’s remote card name; Any-Agent used a local tool-name override; and the Google tests checked credential or callback binding with the collision intact. The preserved UiPath and BeeAI tests did not include matched distinct-name controls. The separate first-party sample test changed only B’s name and exercised duplicate rejection. The broker test stops at the production publish boundary, so final peer consumption remains conditional.

A.1 Evidence integrity

The archived results, source corpus, collision-test patches, and authority-boundary test patches each had an independent manifest. All four manifests passed during verification on 9 September 2026. The deterministic sample test also rechecked its pinned source hashes before execution. The reported counts are generated from a structured results file rather than re-entered manually.

A.2 Reproduction safety

Reproduction requires only disposable local checkouts, recording clients, or loopback HTTP peers. Tests should use synthetic messages and credentials. No external agent, shared broker, production registry, or real user data is required to confirm the resolver behavior. Testing a deployment’s admission or broker-consumption edge requires explicit authorization and is outside this artifact.

Generative AI Usage

To help manage and cross-check the repository-scale evidence corpus, OpenAI Codex assisted with repository inspection, orchestration and checking of experiment scripts, evidence normalization, citation-metadata collection, drafting and revision of manuscript text and $\text{\LaTeX}$ figures, and build and visual verification. The author determined the research questions, scope, methods, evidentiary boundaries, claims, and conclusions; reviewed the cited sources, experimental records, controls, citations, and final manuscript; and made all substantive decisions. Codex outputs were not treated as independent evidence or validation. The author takes full responsibility for the paper.

References

- [1] A2A Project contributors. A2A Protocol Specification at commit 98853be. <https://github.com/a2aproject/A2A/commit/98853be376c88df25e1704771cd3ea9ef8823a96>, 2026. Retrieved 2026-09-09T07:25:32Z.
- [2] A2A Project contributors. A2A Samples at commit 6603ba3. <https://github.com/a2aproject/a2a-samples/commit/6603ba3f2c31a7ef33e70b9d8b5b5f8be42ac9a3>, 2026. Retrieved 2026-09-09T07:25:32Z.
- [3] Adithyan Arun Kumar. Agent Name Collision Attacks: Public Research Artifacts. <https://github.com/adithyan-ak/agent-name-collision-attacks>, 2026. Retrieved 2026-09-22T21:09:49Z.
- [4] Agno contributors. Agno 3.0.9 at commit 6c5112e. <https://github.com/agno-agi/agno/commit/6c5112ea26cc0c1b1171d90b99568967998f3fbc>, 2026. Retrieved 2026-09-19T04:05:25Z.
- [5] AutoDev contributors. AutoDev at commit 23777fe. <https://github.com/phodal/auto-dev/commit/23777fe2d3f1526694f3dd324d82595f651f28bc>, 2026. Retrieved 2026-09-09T07:25:32Z.
- [6] BeeAI Framework contributors. BeeAI Framework at commit 0e7e3e5. <https://github.com/i-am-bee/beeai-framework/commit/0e7e3e58620e66b75dc862dd8c742ea8aded6ca5>, 2026. Retrieved 2026-09-09T07:25:32Z.
- [7] Google ADK contributors. Google Agent Development Kit for Go at commit d56d1a8. <https://github.com/google/adk-go/commit/d56d1a808c46d70b3f2a12d7aa6b4ff9421e45c8>, 2026. Retrieved 2026-09-19T04:05:25Z.
- [8] Google ADK contributors. Google Agent Development Kit for Java at commit 9bcfffd. <https://github.com/google/adk-java/commit/9bcfffd0fd868f4bc63e1e73bd21d8c442693023>, 2026. Retrieved 2026-09-19T04:05:25Z.
- [9] Google ADK contributors. Google Agent Development Kit for Python at commit b018062. <https://github.com/google/adk-python/commit/b0180620f4c2f4f4467a89c37a30f75bf849700b>, 2026. Retrieved 2026-09-09T07:25:32Z.

- [10] Google ADK contributors. Google Agent Development Kit for TypeScript at commit ac84bb4. <https://github.com/google/adk-js/commit/ac84bb49675e7c7b749e9c97c9f4d5243140acb1>, 2026. Retrieved 2026-09-09T07:25:32Z.
- [11] kthota-g. Unique identifier for an agent in Agent Card. <https://github.com/a2aproject/A2A/issues/1014>, 2025. Retrieved 2026-09-09T07:25:32Z.
- [12] Microsoft Agent Framework contributors. Microsoft Agent Framework at commit 540ad6b. <https://github.com/microsoft/agent-framework/commit/540ad6baf0959aced104b1ba355f80529926312a>, 2026. Retrieved 2026-09-09T07:25:32Z.
- [13] Microsoft Semantic Kernel contributors. Microsoft Semantic Kernel at commit 872d29e. <https://github.com/microsoft/semantic-kernel/commit/872d29ea75c6d65c88630c0298c7c8b8396a25e9>, 2026. Retrieved 2026-09-09T07:25:32Z.
- [14] MITRE. CWE-706: Use of Incorrectly-Resolved Name or Reference. <https://cwe.mitre.org/data/definitions/706.html>. Retrieved 2026-09-09T07:25:32Z.
- [15] Mozilla AI contributors. Mozilla Any-Agent at commit 58938fe. <https://github.com/mozilla-ai/any-agent/commit/58938fe0692f8355275deeb42ddc736300124344>, 2026. Retrieved 2026-09-09T07:25:32Z.
- [16] OpenAI. OpenAI Agents SDK 0.20.0 at commit d2bda3f. <https://github.com/openai/openai-agents-python/commit/d2bda3f3110415bf02e526a3983b0d0fa903e0d7>, 2026. Retrieved 2026-09-12T06:15:05Z.
- [17] Palo Alto Networks. Safeguarding AI Agents: An In-Depth Look at A2A Protocol Risks and Mitigations. <https://live.paloaltonetworks.com/t5/community-blogs/safeguarding-ai-agents-an-in-depth-look-at-a2a-protocol-risks/ba-p/1235996>, 2025. Retrieved 2026-09-09T07:25:32Z.
- [18] python-a2a contributors. python-a2a at commit 40b969a. <https://github.com/themanojdesai/python-a2a/commit/40b969a465f85e8156793b7cfac16c5e23ed0b86>, 2025. Retrieved 2026-09-19T04:05:25Z.
- [19] Solace. Queue access types. <https://docs.solace.com/Messaging/Guaranteed-Msg/Queues.htm#Queue-Access-Types>. Retrieved 2026-09-09T07:25:32Z.
- [20] SolaceLabs contributors. Solace Agent Mesh at commit f71e744. <https://github.com/SolaceLabs/solace-agent-mesh/commit/f71e744842a66331a7dbb33e5fba8319dc892d12>, 2026. Retrieved 2026-09-09T07:25:32Z.
- [21] Strands Agents contributors. Strands Agents SDK TypeScript at commit 00e0488. <https://github.com/strands-agents/sdk-typescript/commit/00e04880c30c5ce1f76e40c164d32ce52f7b6dca>, 2026. Retrieved 2026-09-09T07:25:32Z.
- [22] Tom Neaves. Agent In the Middle: Abusing Agent Cards in the Agent-2-Agent Protocol to Win All the Tasks. <https://www.levelblue.com/blogs/spiderlabs-blog/agent-in-the-middle-abusing-agent-cards-in-the-agent-2-agent-protocol-to-win-all-the-tasks>, 2025. Retrieved 2026-09-09T07:25:32Z.
- [23] UiPath contributors. UiPath LangChain Python at commit 5331f6e. <https://github.com/UiPath/ui-path-langchain-python/commit/5331f6e7699575e67aaa4c9ed9b029bc67282804>, 2026. Retrieved 2026-09-09T07:25:32Z.