SMART VERIFY

WHITEPAPER

Your assistant crosses platforms. Your governance does not.

Enterprise assistants read from one SaaS application and act in another. Each vendor governs its own platform. Nothing governs the crossing.

An assistant that touches one platform is governed by that vendor. An assistant that touches two is governed by neither.

August 2026

smartverify.ai

About this paper

This paper is written for the security leaders who have to approve an AI assistant that reaches more than one enterprise system, and for the platform engineers who have to make that approval deployable. It sets out how a cross-platform assistant creates exposure that no individual SaaS vendor is positioned to close, shows the mechanism by which untrusted text in one system drives an authorized action in another, and gives you three questions that establish whether your own estate has this problem today. It assumes no prior familiarity with SmartVerify.

Contents

In summary

This paper sets out six findings, and the sections that follow develop each one.

1. Vendor AI governance is scoped to the vendor's own platform, so an assistant that spans two platforms falls outside every one of them.
2. Native trust layers stop at their own platform boundary. That is an architectural property, not a gap any vendor intends to close.
3. Untrusted text arriving in one platform can drive a valid, authorized call into another. Both platforms behave correctly and the data still leaves.
4. Delegated identity is good design and it answers who acted. It does not record that an assistant acted rather than a person, or what instructed it.
5. The assistant's reach is the union of everything its user can do in every connected system, and nobody granted that union deliberately.
6. After an incident, no single platform can produce a record of what the assistant did across both platforms.

This is not an edge case being extrapolated. Microsoft reports the following about its own installed base.

Reported by Microsoft on its FY26 third quarter earnings call.

20M+

paid Microsoft 365 Copilot seats

up 250 percent year over year

~90%

of the Fortune 500 have active agents

built with low-code and no-code tools

tens of millions

of agents under management

across tens of thousands of companies

The people building most of those agents do not sit on the security team.

Each one chooses, at build time, whose credential it will carry.

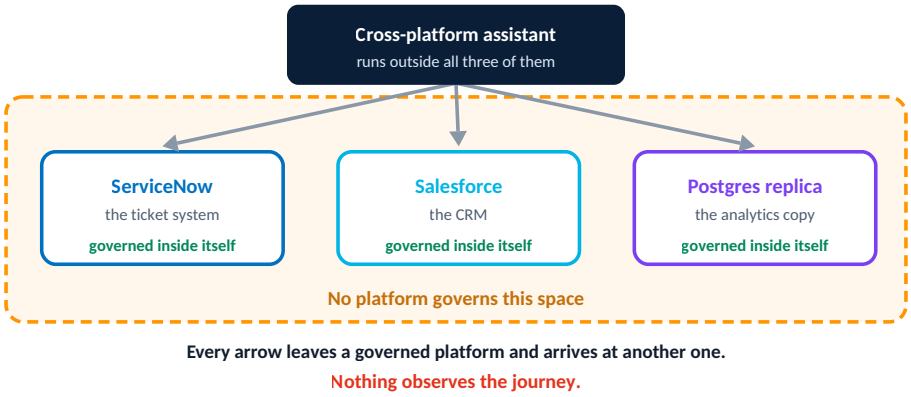
SECTION 01

An assistant that touches two platforms is governed by neither

A support lead opens Microsoft 365 Copilot and asks it to summarize an escalation. Copilot reads the ticket in ServiceNow. To explain who the customer is, it reads the account and the open opportunities in Salesforce. To check what changed recently, it queries the analytics copy both systems are replicated into. One question, one turn, three platforms.

Nobody decided to allow that. The ServiceNow connector arrived in a Microsoft rollout, and the tenant control Microsoft provides is a switch to disable default connectors rather than one to approve each new one. The Salesforce connection was authenticated once by whoever built the agent. The analytics copy was already there. No change ticket describes the combination and no access review examines it, because access reviews look at one system at a time.

Each vendor secured its own platform. The assistant works across all of them.



Each platform governs itself. The journey between them has no owner.

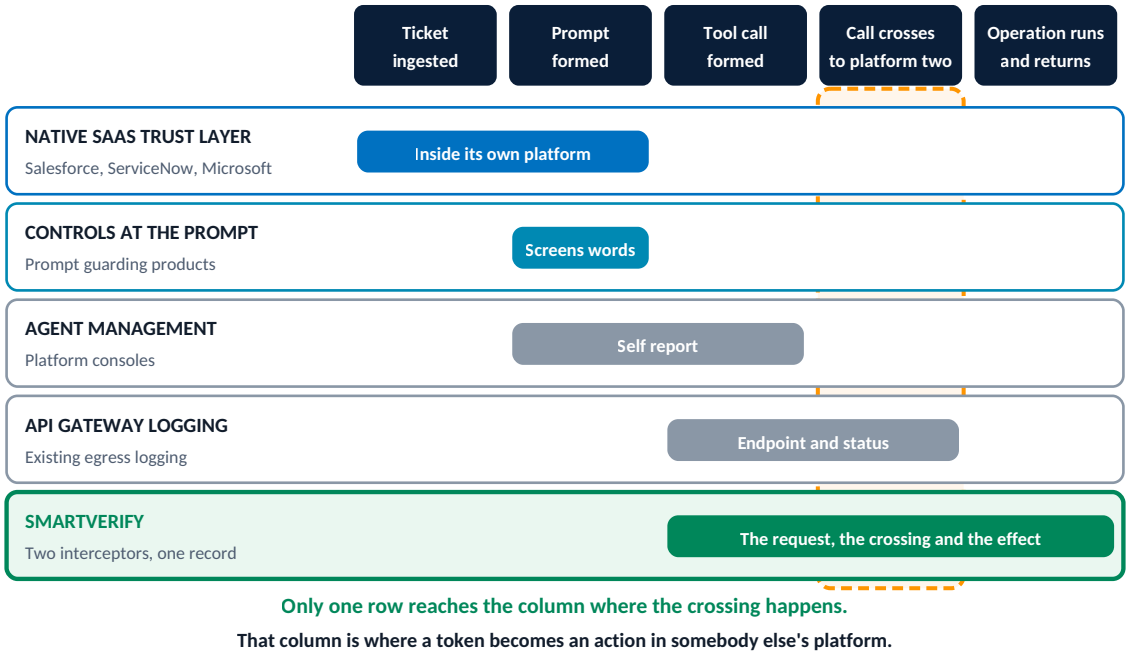
The exposure is not inside any platform. It is in the hop between them, and the hop has no owner, no budget line, and no control.

SECTION 02

Each vendor secured its own platform while the assistant works across all of them

Salesforce governs what happens in Salesforce. ServiceNow governs what happens in ServiceNow. Microsoft governs what happens in Microsoft. A cross-platform assistant is a client to all three, and a client is exactly what none of those trust layers is built to constrain.

The path a cross-platform request takes, and where each control stops.



Coverage stops before the crossing.

Prompt guarding covers a different part of the same flow and every serious deployment should have it. It reads the words of a request and declines the ones that look wrong. It is not positioned on the connection where a request becomes an action in somebody else's platform, so it cannot bound what an allowed request does once it crosses.

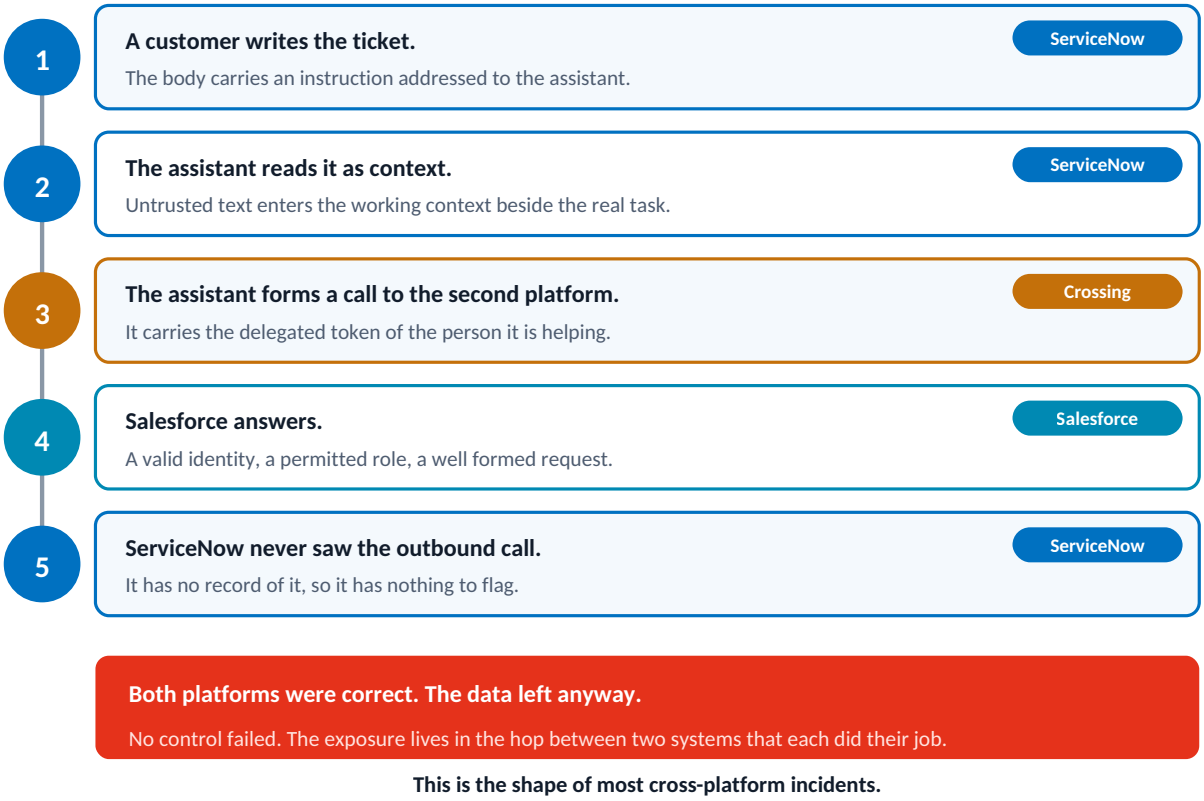
KEY POINT

The column where the crossing happens is the column nothing currently covers. That is the whole finding.

SECTION 03

One injected ticket can drive a valid call into a second platform

The mechanism is unglamorous and it works. A customer writes a support ticket whose body contains an instruction addressed to the assistant rather than to a human. The assistant reads the ticket as context, because reading the ticket is its job.



This is the shape of most cross-platform incidents.

The chain runs top to bottom and ends in a correct outcome.

The call the assistant then makes is well formed, carries a valid token, and stays inside the scope you granted. The receiving platform has nothing to refuse, and the originating platform never saw the outbound request. Every control in the estate behaved correctly.

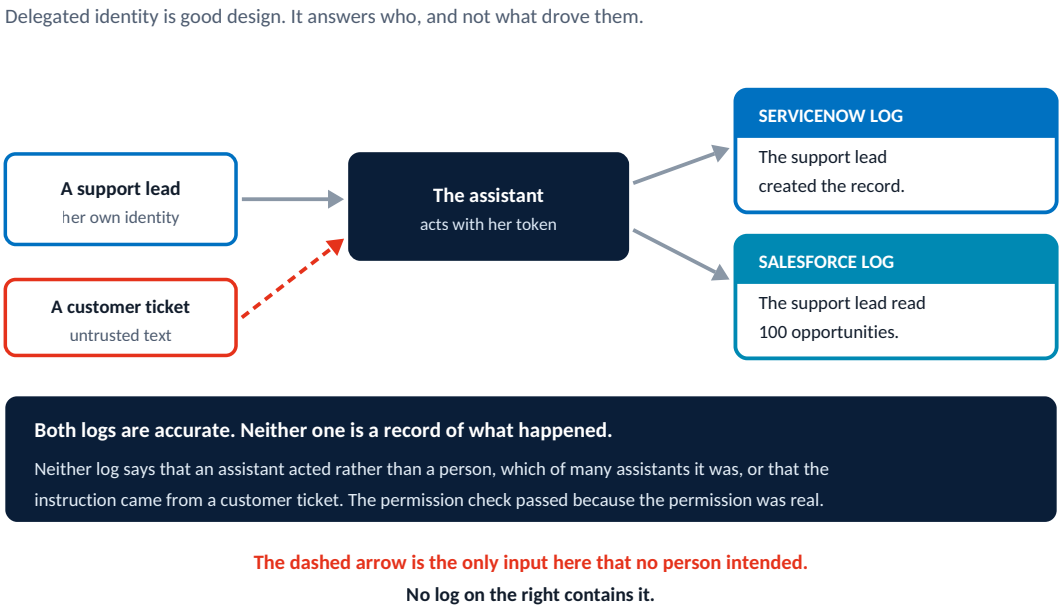
You cannot fix this by tightening either platform, because neither platform did anything wrong.

SECTION 04

Delegated identity names the person, not the assistant

Modern integrations handle identity well. When a Microsoft Copilot acts in ServiceNow it typically uses on-behalf-of delegation, passing the signed-in user's own token, and ServiceNow applies that individual's roles. Nothing is impersonated and no shared account is involved.

That is the right design, and it answers a different question than this paper asks. It establishes which person the action was for. It does not establish that an assistant took it, which of several it was, or what text instructed it.



The deputy acts with real authority, directed by something else.

Security literature calls this a confused deputy. The deputy acts with the principal's authority while something else decides what it should do. The permission check is not bypassed. It passes, correctly, because the permission is real.

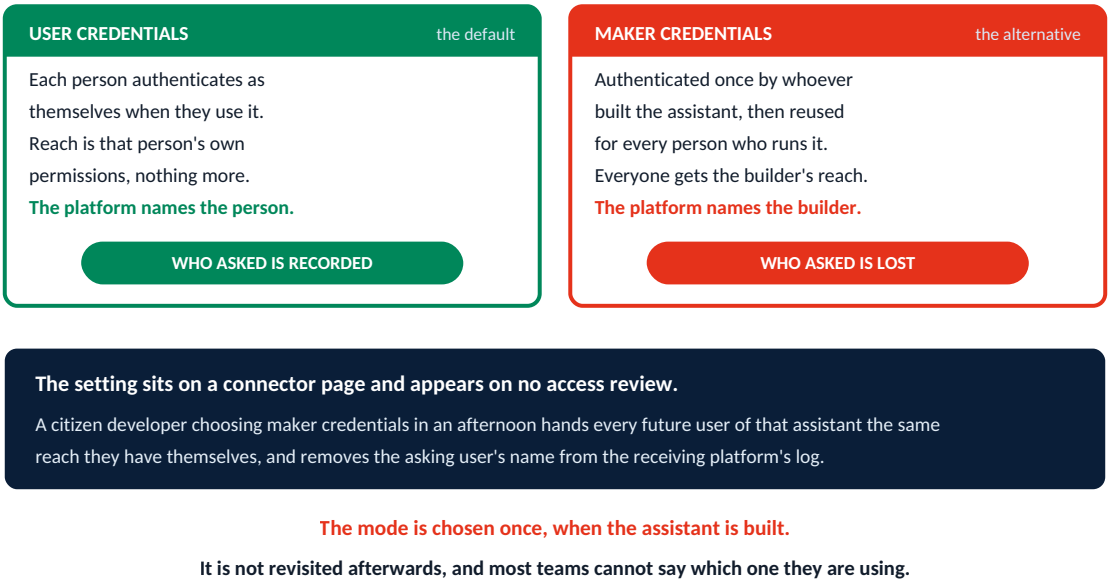
An authorized action driven by untrusted content looks like ordinary work in every log that records it.

SECTION 05

The connection can authenticate as its builder instead of its user

The delegated case in the previous section is the one everybody describes. It is not the only option. Microsoft's own documentation for Copilot Studio connectors offers two credential modes, and the choice between them is made by whoever built the assistant.

Microsoft documents two credential modes for a Copilot Studio connector. The builder picks one.



Two credential modes, and only one names the person who asked.

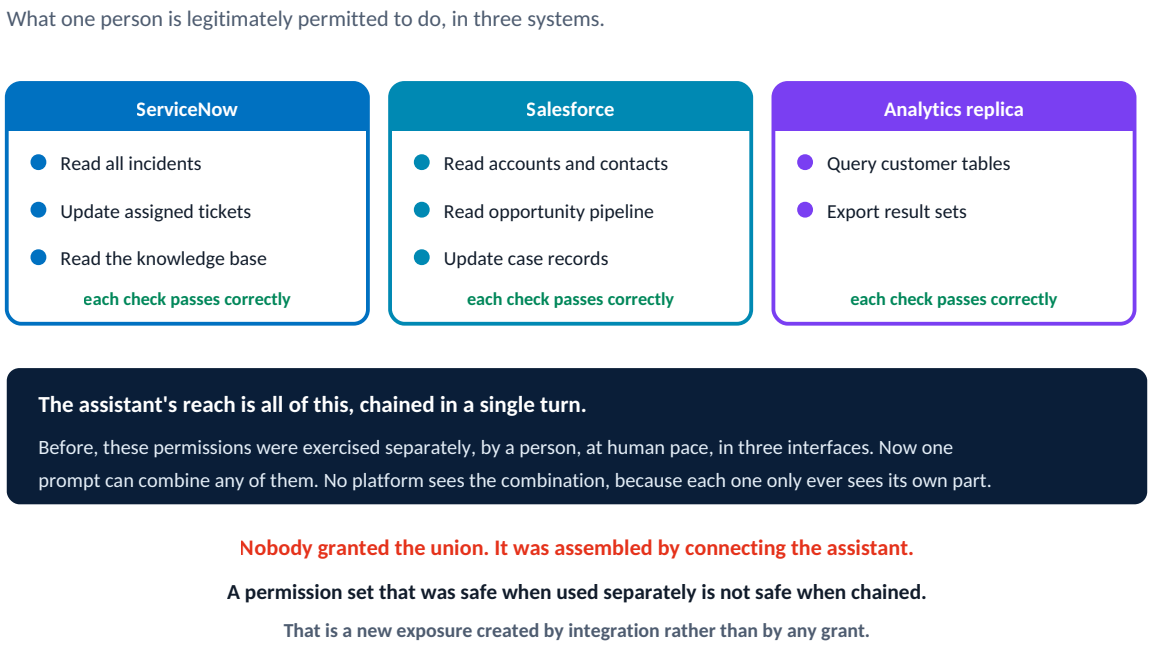
User credentials are the default. The alternative is maker-provided credentials, where the connection is authenticated once by the person who built the assistant and then used for everyone who runs it. Every user reaches as far as the builder can reach, and the receiving platform records the builder rather than the person who asked.

Ask which credential mode each of your assistants uses. The answer is often that nobody knows, and the setting is on no access review.

SECTION 06

The reach is the union of what that identity can do everywhere

Take the safest case, an assistant acting for one support lead with entirely reasonable access. She can read incidents in the ticket system, read accounts and pipeline in the CRM, and query the analytics replica. Each grant was reviewed and each is appropriate on its own.



Three reviewed permission sets, exercised as one.

Before the assistant existed, chaining those permissions took intent and effort. Now one prompt combines them in a single turn, and the combination was never reviewed, because no access review looks across three platforms at once.

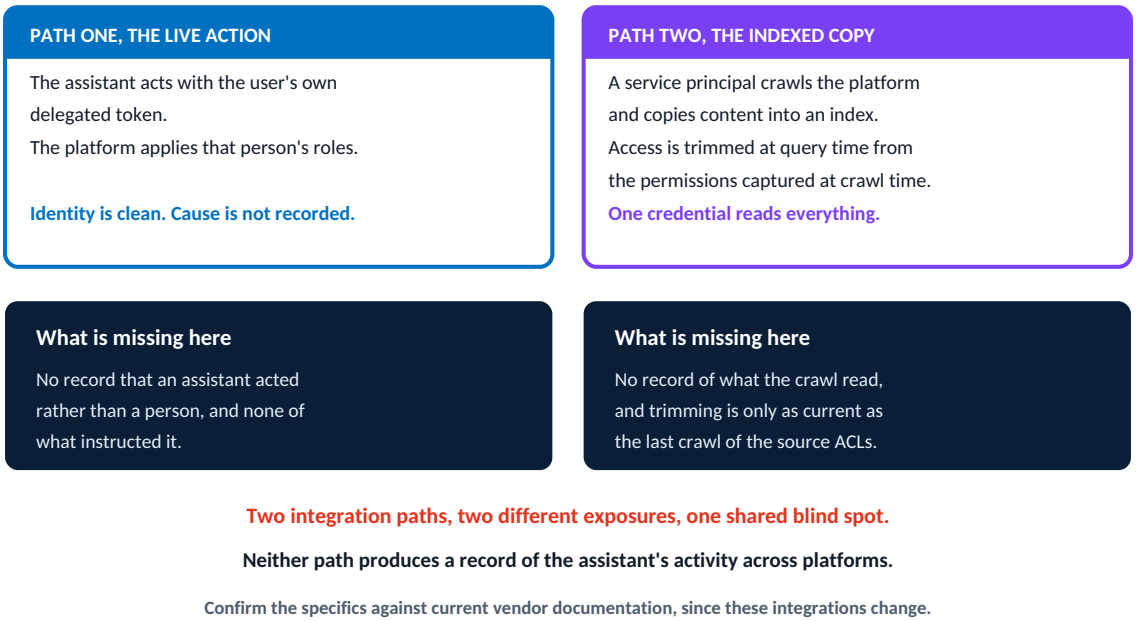
In the maker-connection case the union is not hers at all. It is the builder's, exercised by everyone who uses the assistant.

Nobody granted the union. It was assembled by connecting the assistant, and it appears on no access review anywhere.

SECTION 07

Indexed knowledge is a second exposure with a different shape

Live actions are only one integration path. The other is indexed knowledge. A connector crawls the source platform using a service principal, copies content into a search index, imports the source access control lists, and trims results at query time so a user sees only what they are entitled to see.



Two integration paths that fail in different ways.

Two things follow. The crawling credential does have broad access, because crawling everything is its job, and there is no record of what it read into the index. And trimming is only as current as the permissions captured at crawl time, so a change in the source system is correct in the source and stale in the index until the next crawl.

KEY POINT

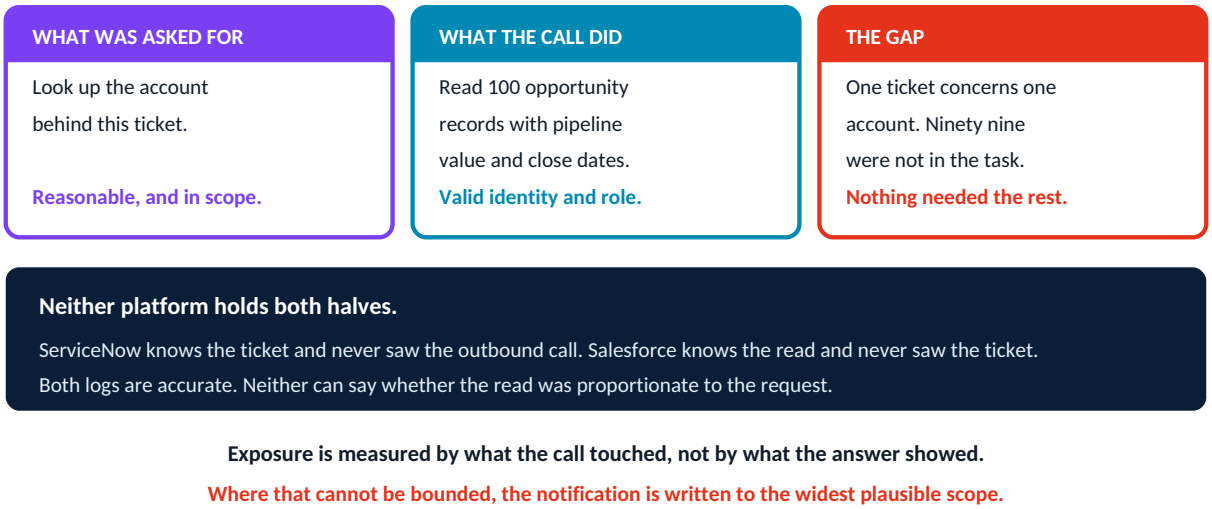
The live path has clean identity and no record of cause. The indexed path has a broad credential and a permissions snapshot. They fail differently and neither produces a cross-platform record.

SECTION 08

Nobody can produce the cross-platform record after an incident

Put the question the way it will actually be asked. For one named assistant, on one named day, what did it read and change in each platform it touched, and was any of it beyond what the task required.

One escalation. Two observations. The distance between them is the finding.



Exposure is measured by what the call touched, not by what the answer showed.

Where that cannot be bounded, the notification is written to the widest plausible scope.

The request and the effect sit in different platforms, so nobody compares them.

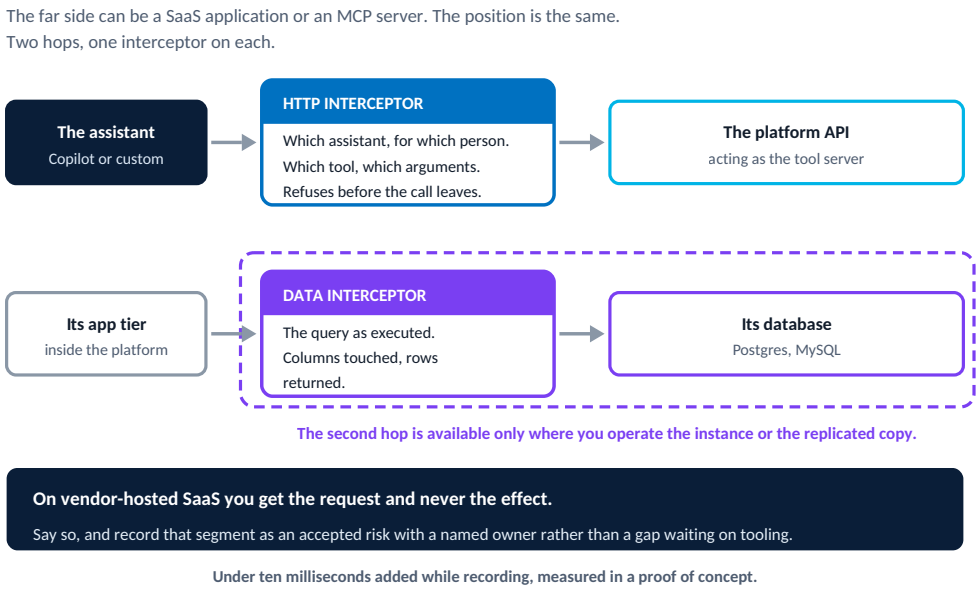
Answering that today means exporting logs from two or three vendors, aligning clocks that do not agree, and correlating on a credential that is the same for every assistant. That is reconstruction, it takes weeks, and it produces an estimate. Where the scope of an exposure cannot be bounded, counsel assumes the widest plausible scope and the notification follows that assumption.

The difference between contacting a defined set of customers and contacting all of them is frequently the entire cost of the incident.

SECTION 09

SmartVerify observes the hop that nobody owns

SmartVerify places an interceptor where the call is formed and another where the operation lands. The first sees which assistant acted, for which person, and with what arguments. The second sees what the receiving system did and what came back.



The same two interceptors, whether the far side is a SaaS platform or an MCP server.

Pairing those two observations produces one record per interaction, joined across a boundary that no single vendor can see both sides of. That record is the artifact an investigation needs, and it is the artifact a compliance reader is asking for when they ask whether agent activity is monitored.

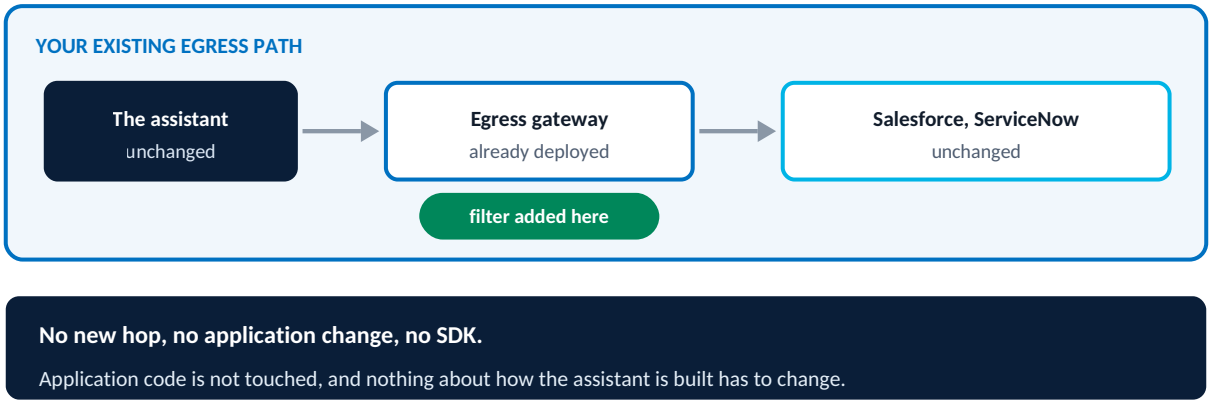
There is no second architecture here. A SaaS application receiving a tool call occupies the position an MCP tool server occupies, so ServiceNow, Salesforce, and a self-hosted MCP server are parsers on the same interceptor. So is the replicated copy your platform data is synced into, which is often the least governed store you own and the one place the effect is always observable.

- **Record.** Every outbound call and the operation it produced, joined into one entry with the fields touched and the volume returned.
- **Refuse.** Decline the call before it leaves your network, while a decision is still available, returning an error the assistant framework already handles.

SECTION 10

Platform engineering deploys this without touching application code

The reason this is adoptable is that it asks nothing of the application. The filter loads into the egress path your platform team already operates. No SDK, no rewrite, no new network hop, and no change to how the assistant is built.



Recording can be switched on before any decision about what to refuse.

Nothing in the path changes except what the gateway now inspects.

Recording can be switched on before anyone decides what to refuse, which is what makes the first conversation with a change board short. Refusal at the outbound call is the safer surface to enforce on first, because the assistant framework already expects a typed error from a tool call and nothing in the business depends on that path behaving a particular way.

SECTION 11

The records answer what an investigator actually asks

Regulatory frameworks converging on agentic AI ask for records of what an automated system did, and for a mechanism to intervene. They do not ask for a product category. The useful test is whether you can answer the questions an investigator asks.

What an investigator asks after a cross-platform incident, and where each answer comes from.

✓ Which assistant acted, and for which person	At the call
✓ What it asked the second platform to do	At the call
✓ What that platform actually returned	At the effect
✓ Whether the two were proportionate	The comparison
✓ Whether a refusal happened, and why	The decision
! Whether the model itself was manipulated	Prompt layer, not here

Five of the six are answerable from one record.

The sixth belongs to the prompt layer, and we say so.

Where each answer comes from.

SmartVerify produces the record and the refusal decision. It does not observe prompts or model behavior, so it says nothing about whether a model was manipulated. That belongs to the prompt layer, and stating the boundary plainly is what makes the rest of the claims checkable.

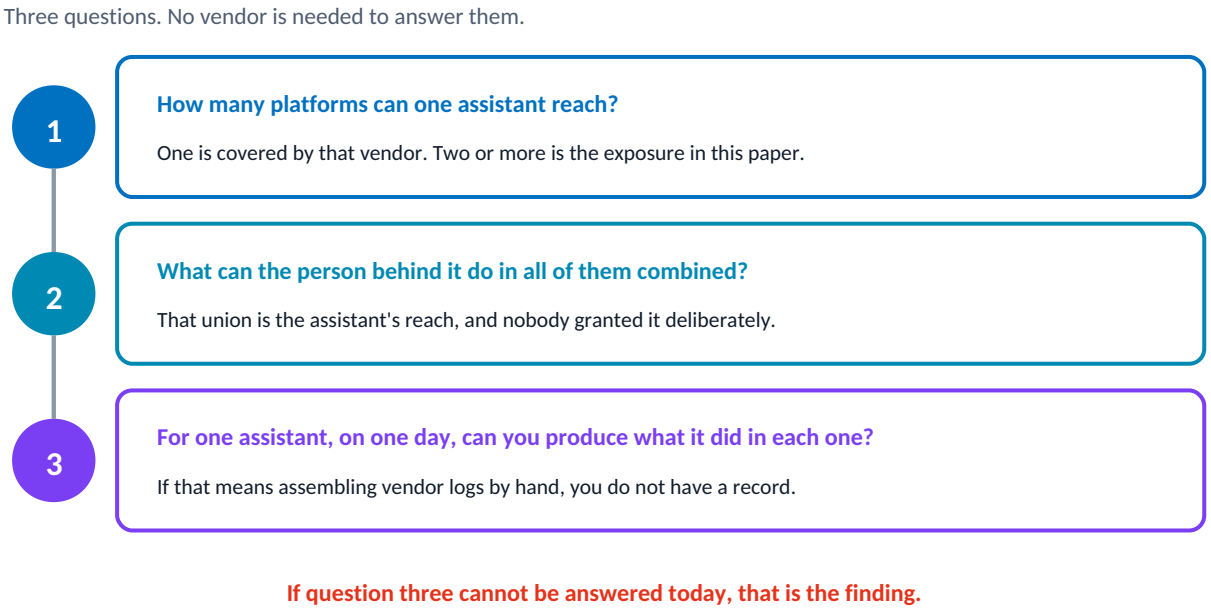
KEY POINT

A control that produces records an investigator can use is worth more than one that produces a posture score.

SECTION 12

Three questions establish whether you have this exposure today

None of these needs a vendor to answer, and the third is the one that matters.



A fixed scope assessment on one assistant, observing only, on real traffic, is the usual starting point. The deliverable is a report on your own cross-platform activity in two parts. What crossed, and what crossed beyond what the task required. It states the share of traffic that could not be classified, because a coverage figure a reader cannot see is not a coverage figure.

- **Record the crossings first.** It changes nothing about how the assistant behaves and it can go on real traffic quickly.
- **Refuse once the records show something worth refusing.** The outbound call is the surface to start on, because it carries the new risk and no legacy dependency.