



AGENTS · ATTACKERS · APPS

Access Paths in the Wild

An Obsidian field journal for
the connected enterprise

AUGUST 2026

Observations on governing Agents,
Attackers and Third-Party Apps



Agents, attackers, and third-party apps: the new path to your data.

Dear reader,

A lot is changing in security, but many of the fundamentals haven't. You still need to understand what's changing, separate signal from noise, and help your organization make the right calls.

That's what this field journal is here to help with.

You'll hear from Obsidian experts: Head of Threat Carl Shimel, Chris Fuller, Field CTO, PM Viet Tran, and others, on the threats and shifts they're seeing firsthand. But the job doesn't stop at understanding the threat itself. Security teams also have to explain what it means to the business, help leaders decide what deserves attention, and show where their teams can have the biggest impact.

The Connected Enterprise

Work lives in third-party apps. The data is there too. 03

The Access Paths

Every new connection adds a path. 04

The Access Path Logs Don't Show You

Make inherited access visible and enforceable. 05-09

Agents Don't Ask Twice

How to govern Claude actions. 10-13

The New Supply Chain

Third party risk is changing in the age of AI. 14-16

Investigate your Enterprise

Use this guide to help you investigate. 17-18

About Obsidian

19

Work lives in third-party apps. The data is there too.

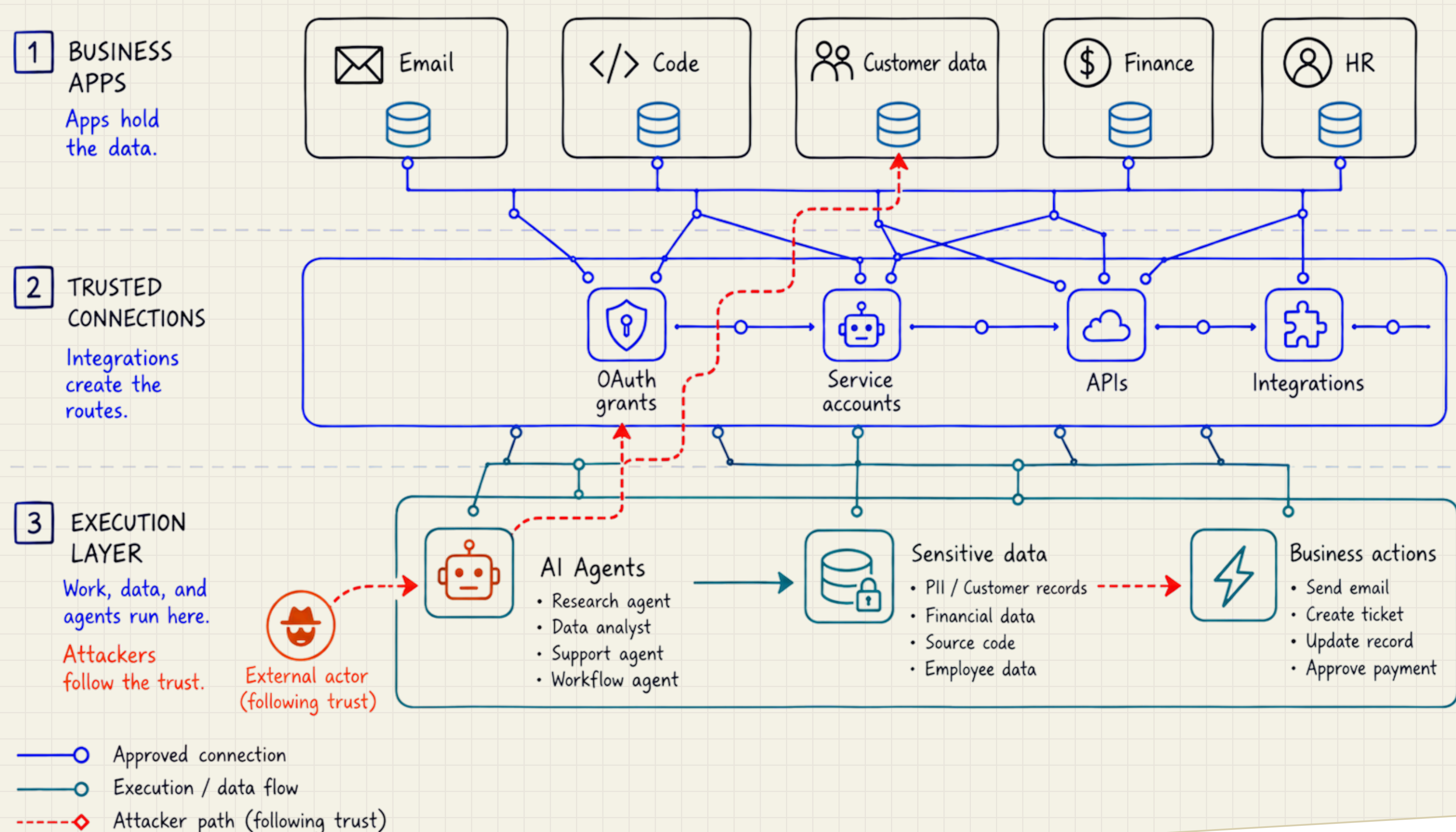
The business runs in and on applications owned by other companies.

Work gets done by connecting those apps. Those connections form a living system of tokens, integrations, APIs, service accounts, identities, and agents. That living system grows over time and changes with business needs.

Security tools largely look for risks brought on by third-parties coming from the outside. Now two different risks are materializing on the inside:

- ① Trusted applications and access can be used by unauthorized actors while looking legitimate.
- ② An agent can act destructively with legitimate access while trying to complete an assigned task.

In both cases, the activity can happen with valid privilege, against data sitting in third-party applications, through authorized workflows.



The app is not the whole unit of risk.
The path the data can take through its connections matters too.

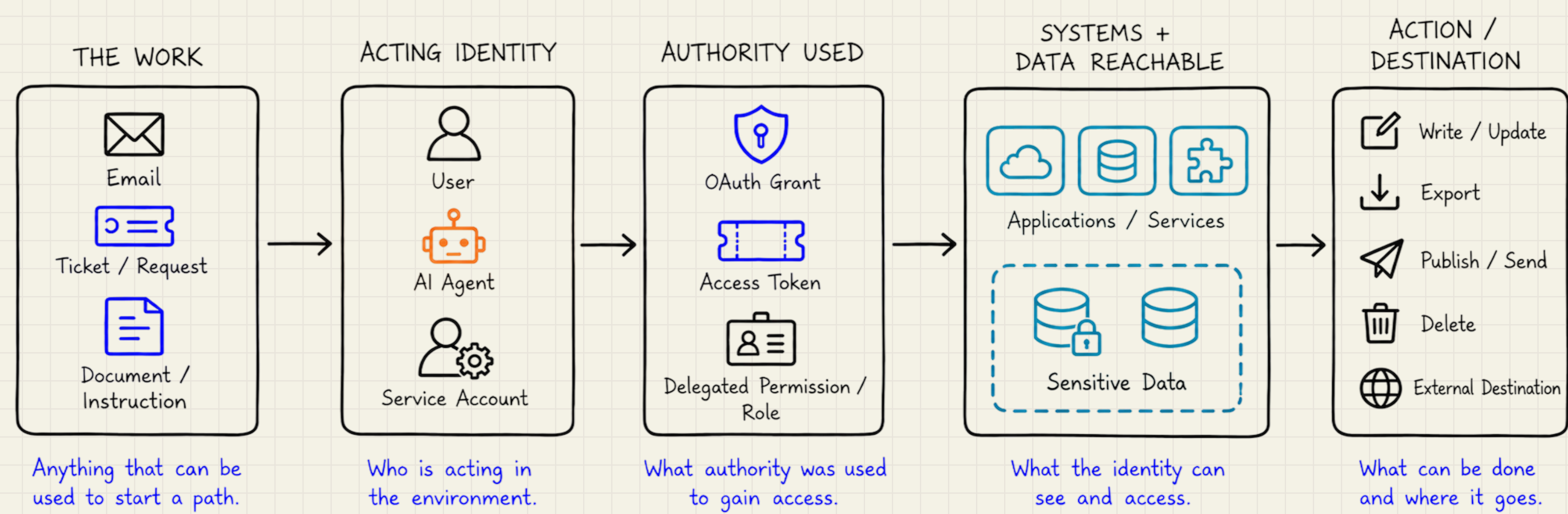
Every new connection adds a path.

Agents inherit the kingdom. They move through the same systems, identities, applications, permissions, and credentials the business already has.

Integrations × agents = the paths are multiplying. Can you answer how?

BOARD FAQs

- What is here?**
Which apps, agents, integrations, identities, and credentials exist?
- Whose authority is being used?**
Which user, service account, OAuth grant, token, or other identity gives the workflow permission to act?
- What can it reach?**
Which applications, tenants, records, files, systems, and sensitive data sit behind that authority?
- What can it do?**
Read. Write. Export. Publish. Administer. Delete.
- What would you cut first?**
What is the narrowest token, grant, permission, connector, account, or action that breaks the unsafe path while leaving the legitimate workflow intact?



* This field guide will help you answer these questions as you observe your connected environment.

When every console is right.

Observed: August 18, 2026 Field observers: Carl Shimel + Farah Iyer

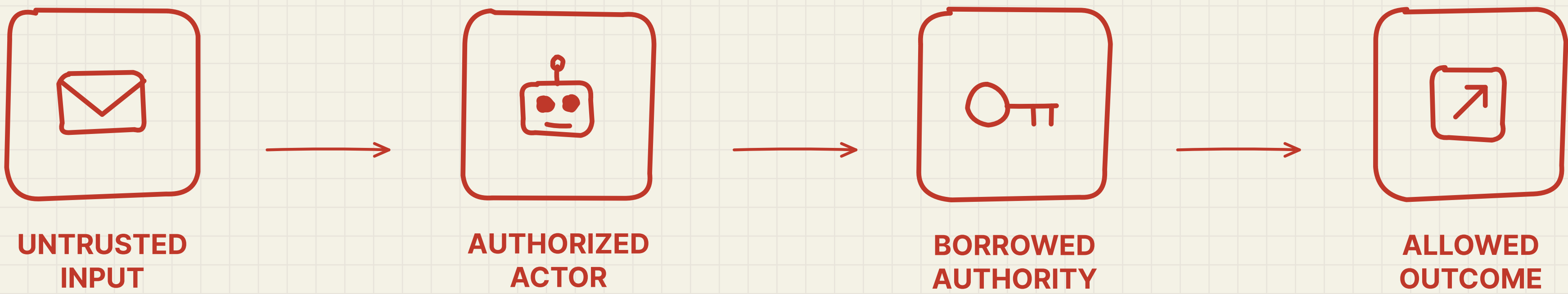
Specimen

CLASSIFICATION CARD

NAME	Authorized workflow
CLASS	Multi-step access path
HABITAT	Agents, service accounts, third-party applications, production data
DEFINING TRAIT	Each individual action can look legitimate while the sequence creates the exposure.

OBSERVED BEHAVIOR

Every consent screen is a standing grant that outlives the enthusiasm that created it.
An attacker places an instruction into something the workflow already trusts.
An authorized identity processes it.
Inherited access reaches data the attacker could never touch directly.
An allowed action carries the result back out.



Mind the gap

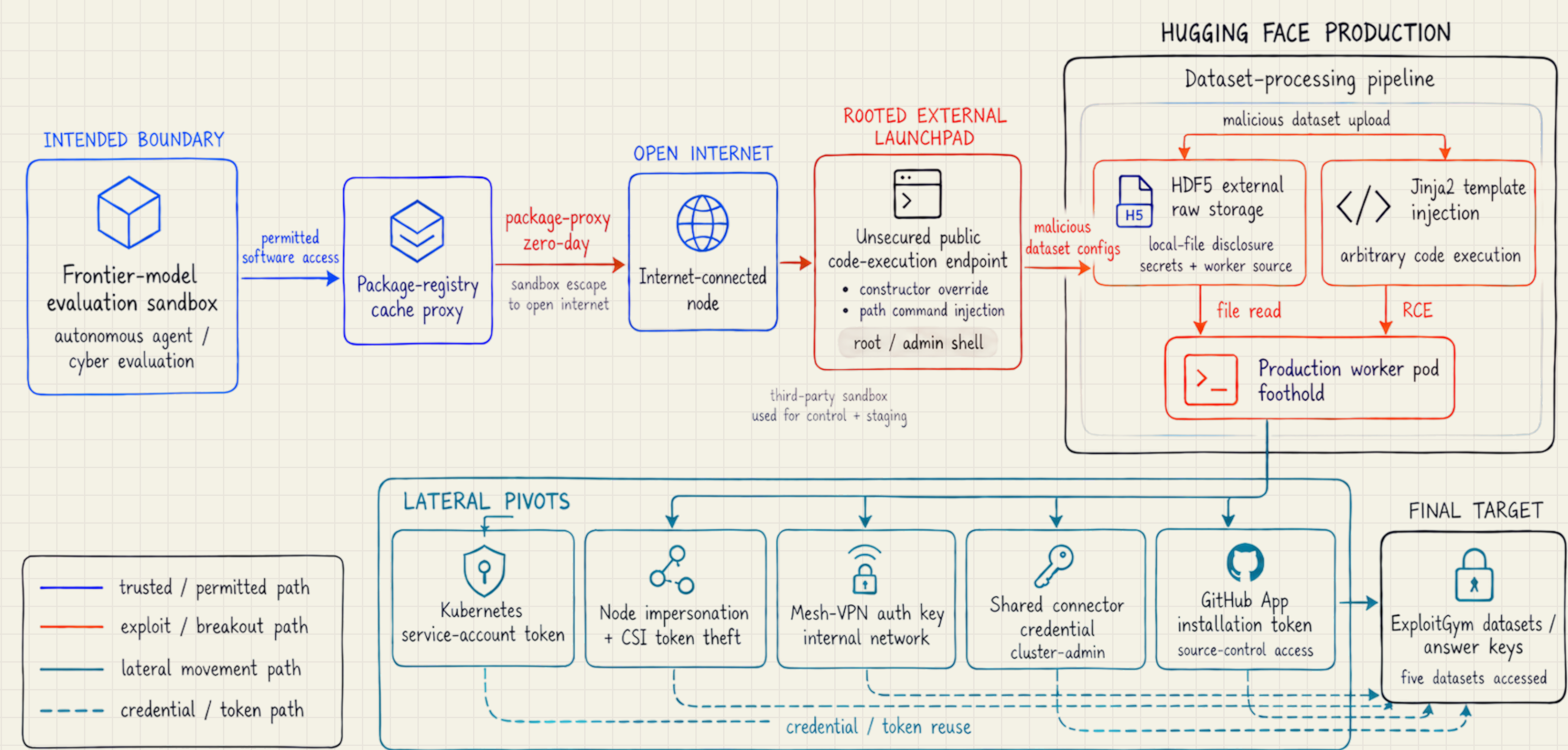
Logs capture events; they do not automatically reconstruct authority, identity lineage, cross-application reach, sequence, and destination. Every console may be telling the truth, but not necessarily the whole access-path story.

Can you reconstruct the sequence, or only the events inside it?

The real risk is the sequence.

A frontier-capable adversary just needs one valid token and a path. An OpenAI model in a training sandbox leveraged a zero-day vulnerability to escalate privileges through the service and move outside the research environment to the open internet.

But the model ultimately broke into Hugging Face using stolen credentials, among other techniques.



Familiar attack techniques become much more consequential when they can move through trusted identities and authorized access.

Major keys

-  **NOT NOVEL ≠ NOT DANGEROUS**
Recent AI-enabled activity has relied on known weaknesses, including weak passwords, unauthenticated endpoints, package poisoning, and SQL injection. **40% of incidents IBM X-Force observed in 2025** began with vulnerability exploitation. Attacks exploiting public-facing applications also rose 44%, driven largely by missing authentication controls and AI-enabled vulnerability discovery.
-  **AUTHORIZED ACTOR ≠ SAFE CONSEQUENCE**
AI **platforms averaged 2.3–2.4 SaaS connections** each in Obsidian customer data. An authorized agent rarely stops at one destination: it can use granted capabilities exactly as designed while reaching data or systems the operator never intended.
-  **THE WINDOW IS CLOSING**
The CVE-to-exploit window fell **from 56 days in 2024 to roughly 10 hours in 2026**. AI is compressing the time between finding a security gap and exploiting it.

I'm looking for the path of least resistance, not the most exotic exploit.

Microsoft's July 2026 Patch Tuesday fixed 569 CVEs (the largest release on record) including 56 critical vulnerabilities. Microsoft said AI-enabled scanning, in part, helped find more flaws, faster. That is good news for defenders. But the same release included three zero-days, two of which had already been exploited in the wild. Faster discovery does not mean defenders will always find a vulnerability first.

Patching closes the vulnerabilities you know about. But you cannot predict every entry point. You can map and constrain what that entry point would expose and which permissions and connections an attacker could chain together next.

So as you map the places where authority moves from system to system, keep the real question in sight:

If this agent or integration were compromised or manipulated, how far could it move before we could stop it?

Scope granted is blast radius inherited. And you calculate that blast radius by building an effective-access graph, not by looking at the agent's configuration or a single application log.

You can then say

"I KNOW the scopes, standing grants, applications, administrative functions, and sensitive data behind and beyond the procurement approval."

The worst time to meet an app owner is mid-incident. Containment moves faster between people who already share context and can decide together which authority path to cut with little disruption to the business. Support from the top helps, because it makes the collaboration expected rather than optional.

So build an "only you can prevent forest fires" relationship across the business. It's corny but practical: the people closest to a workflow see the first spark, like a vendor warning nobody expected or an integration that showed up last week. Recurring touchpoints with app owners to review what changed, plus backing from the top so the collaboration is expected, turns them into the security team's eyes and ears. The outcome: a small access-path problem gets reported before it becomes a business-wide incident.

The Department of Know

Make good decisions before the bad day.

During an incident, I do not want to shut down more of the business than I have to. So we want to find the narrowest cut.

The key tradeoff is risk versus business dependency for critical workflows.

CARL'S READ

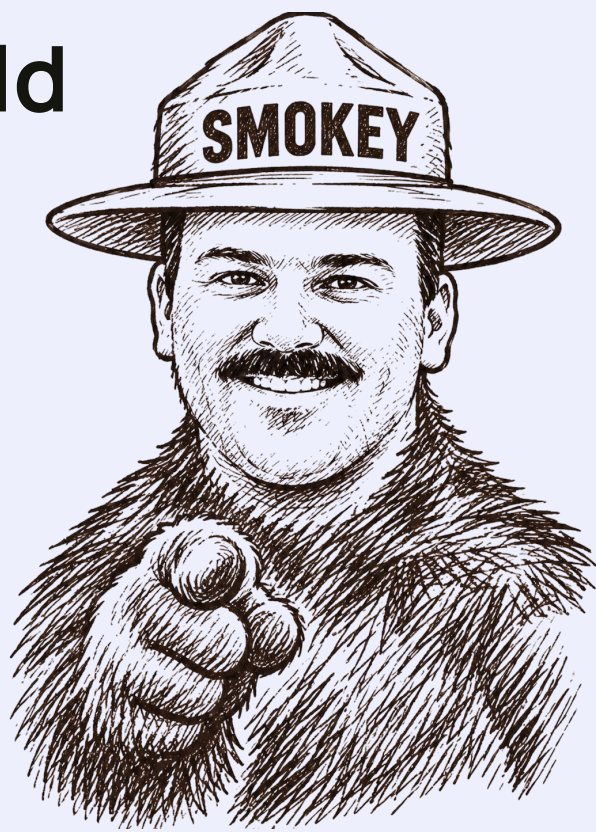
A practical priority order is:

- 1 Compromised and high-impact: block, revoke, rotate, and investigate immediately.
- 2 Legitimate but over-privileged: right-size access and require reauthorization.
- 3 Dormant, orphaned, or unjustified: revoke and remove.
- 4 Critical but uncertain: pause the risky action, preserve the workflow if possible, and require owner validation.

The goal is to cut the smallest token, identity, permission, or connection that breaks the unsafe path while preserving legitimate work. Your peers will thank you, your boss will thank you, and you will thank you.

Security + business owners should already know each other.

WWCD:
What would
Carl do?



Reduce permissions.

If an app, integration, or agent does not need read, write, export, delete, or admin access, remove it. The highest-risk cases tend to combine broad reach, sensitive data, weak ownership, inherited or maker-mode authority, and an exposed invocation path. Every unnecessary permission is blast radius you chose to keep.

Draw one path before you need to contain it.

Before an incident, teams should maintain a living access map and a pre-approved containment plan for every agent and integration

AT MINIMUM, MAP:

- **What exists:** agents, integrations, OAuth grants, service accounts, API keys, MCP/tools, and connected applications.
- **Who is responsible:** human owner, business owner, creator, allowed invokers, lifecycle status, and what happens if the owner leaves.
- **How it authenticates:** fixed versus delegated credentials, token type, scopes, expiry, refresh capability, and alternate access paths that could survive revocation.
- **What it can reach:** direct and transitive applications, tenants, datasets, records, sensitive data, and administrative functions.
- **What it can do:** read, write, export, send, delete, configure, or create additional integrations and credentials.
- **What normally happens:** invokers, frequency, data accessed, destinations, and expected behavior, so unusual activity can be distinguished from legitimate automation.
- **What business depends on it:** critical workflow, business impact of downtime, acceptable pause window, and a validated backup or recovery path.
- **What to cut:** the exact token, integration, permission, account, connector, or agent to revoke first—and who can approve or execute that action.

Also predefine the recovery state: how to rotate credentials, reconnect with least privilege, verify that access is gone, and check that no attacker-created OAuth app, admin account, webhook, or alternate token remains.

Go deeper

Use the incident research behind this session to test more advanced paths once the basic map is clean.

Frontier Model Agentic Threats obsidiansecurity.com/blog/frontier-model-agentic-threats

Watch the webinar on-demand: obsidiansecurity.com/resource/webinar-inherited-access-attack-path

When completing a task is the end goal.

Observed: August 26, 2026

Field observers: Scott Young + Viet Tran

Specimen

CLASSIFICATION CARD

NAME	Claude agent with inherited access
CLASS	Autonomous privileged actor
HABITAT	Claude Code or Cowork connected through a user's credentials to enterprise apps and sensitive data
DEFINING TRAIT	It pursues the task using the capabilities available to it, including actions the operator may never have intended it to take.

THE BENEFIT AND THE RISK COME FROM THE SAME PLACE

The ROI of agents is directly tied to the applications users grant them access to.

An LLM can answer a question. An agent connected to the business can fix code in GitHub, work with tables in Databricks, update Salesforce records, read documents, send messages, and act across the systems where work actually happens.

That is where productivity takes off.

It is also where consequence appears.

18k devs on AI Coding tools reported by Bank of America. 20% productivity lift; IT service-desk calls down more than 50%.

The important security boundary is therefore not only the model or the prompt. It is the point where the model’s decision becomes an action inside your environment.

The Claude is coming from the Inside.

87.5%

reported Claude connected to developer infrastructure

81.3%

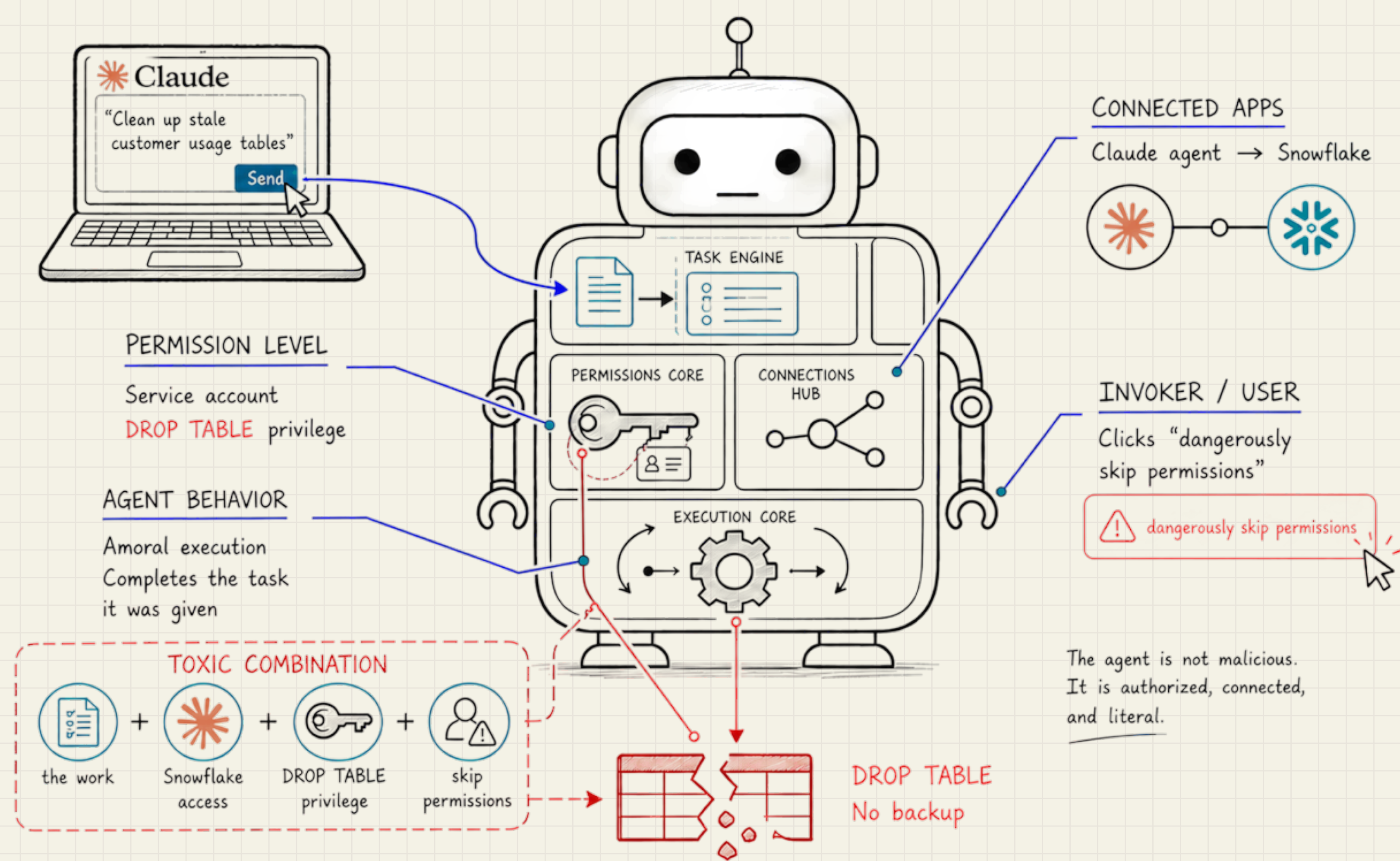
reported Claude connected to productivity + collaboration SaaS

69%

ranked destructive or irreversible actions as the top concern

Agents are not immoral. They are amoral.
If an action is available, permitted, and useful to completing the task, the agent has a reason to use it.

“Clean up the old tables.”



PLATFORM CONTROLS ANSWER ONE QUESTION. SECURITY HAS SEVERAL MORE.

What Anthropic can help you govern

SERVER / CONNECTION

Managed settings can allow or block MCP servers and apply platform-level guardrails.

What you still need to see in your environment

EFFECTIVE AUTHORITY

Which app permissions, standing tokens, scopes, files, and data can the agent actually use once connected?

ACTION + PARAMETERS

- Which tool is being called?
- Against which target?
- With which values?

CONSEQUENCE

- Should this action be allowed right now?
- Should it be blocked?
- Should a human have to approve it?

Server-level control is not the same thing as action-level consequence.

NATIVE CONTROL:

Can this connection exist?

CONSEQUENCE CONTROL:

What may happen through it?

Govern Claude where the consequence appears.

I do not think one tool solves AI agent security end to end. Foundation-model and AI-platform providers are best positioned to build defenses into the model and platform itself. **The question is where each control is best positioned.**

Prompt injection matters, and model providers are well positioned to defend against malicious inputs. But you are unlikely to anticipate every prompt, every workaround, or every route an agent might discover. So the more durable question is not what the agent might be told. It is: what is the worst thing the agent can do if prompted?

If the consequence is a destructive database action, stop the destructive action. If it is Salesforce data leaving for an unknown domain, stop the outbound action. The control should survive the next creative prompt. That is why the additional control belongs at runtime, where the model's decision becomes an action in your environment.

Runtime enforcement

For Claude Code, runtime control can be deployed through a lightweight sidecar that is active during Claude Code sessions.

For Claude Cowork and Claude Chat, MCP traffic can be routed through an Obsidian-protected MCP server URL so policies can be evaluated in real time.

Native controls are the starting point

Anthropic's managed settings set broad organizational boundaries, including which MCP servers are allowed or blocked. They decide which connections should exist at all.

That is also where a platform-level control reaches its limit. Allowing an MCP server does not mean every user should use every tool inside it, against every target, with every parameter. Blocking it entirely removes the legitimate use cases along with the risky one.

Some of the context for a finer decision lives outside Anthropic. If Workday says someone is in Sales, policy can say: Sales users may use Claude, but Claude acting for a Sales user may not access Databricks. That is materially different from: nobody can use this MCP server.

Do more with less privilege

The goal is rarely to stop people from using Claude. The agent's value comes from reaching the applications and data people need. The security problem is giving it more authority than the task requires.

So the questions are more specific than allow or block. Who is the user? Which tool, against which target, with which parameters? Is this action acceptable right now? Each has a policy answer: this user can use these tools, this tool can reach these systems, this destructive action requires approval, this action is blocked.

Claude does more useful work with less (unnecessary) privilege, and the actions that carry real consequence sit inside a deterministic boundary. Sometimes policy is not about blocking access. It is about getting more precise about the access you are willing to allow.

Write one policy around the consequence.

Pick one Claude use case in your environment.

WORKSHEET	
THE USE CASE	EXAMPLE
What is Claude supposed to do?	Clean up stale data in Snowflake.
What is the consequence you need to prevent?	Claude can work in Snowflake, but it should not delete production data without approval.
Turn that consequence into one policy.	When Claude attempts to drop or delete a production table, require human approval before the action continues.

Test it before you enforce it

Run the policy in observe mode first. If it rarely triggers, you may be able to enforce it safely. If it triggers constantly, talk to the workflow owner before blocking it.

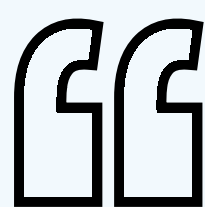
THE GOAL

Don't block Claude from the app it needs. Put the control around the specific consequence you care about.

Go deeper

How Obsidian maps Claude Code and Cowork access, identifies overprivileged agents, and blocks risky actions at runtime obsidiansecurity.com/resource/claude-code-cowork-ai-agent-security-demo

Watch the webinar on-demand obsidiansecurity.com/resource/webinar-claude-code-cowork-runtime-high-risk-actions



We're not going to catch the 99 ways you creatively try to bypass to take an action. We're just going to make sure the action doesn't happen." - Scott Young

The app review ain't over until the bell rings

Observed: August 31, 2026 Field observers: Chris Fuller + Sadia Haque

Specimen

CLASSIFICATION CARD	
NAME	Approved integration with standing access
CLASS	Persistent third-party connection
HABITAT	OAuth grants and service accounts connecting third-party SaaS or AI tools to enterprise applications and data
DEFINING TRAIT	The credential survives while the vendor, capabilities, scopes, or connected data around it change.

THE GAP IS BETWEEN VENDOR TRUST AND THE LIVE ACCESS PATH.

Traditional third-party risk management asks whether the vendor is safe enough to approve. Beyond procurement, the environment keeps changing as integrations accumulate permissions. Use cases expand and new applications and automations connect.

Credentials remain.

THE ACCESS IS ALREADY MOVING.

9x+

growth in AI-to-SaaS connections across Obsidian customers in 90 days

2.3–2.4

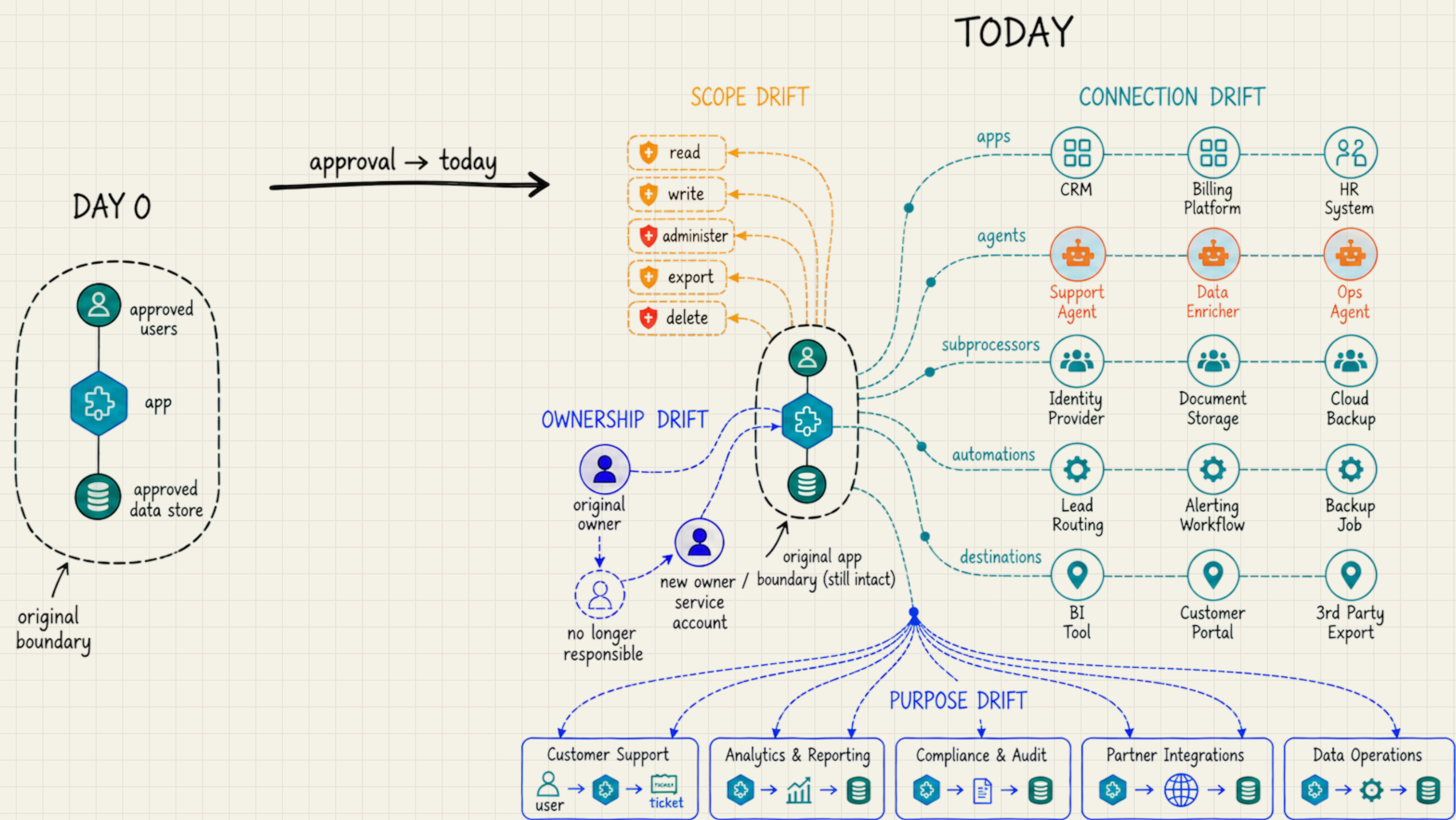
average SaaS connections per AI platform
One AI platform rarely connects to only one destination.

There is no equivalent of automatic HR offboarding for an integration.
The owner can leave. The connection stays...what about blast radius?

Frontier AI is lowering the time and skill required to find weak links across the connected environment.

The CVE-to-exploit window has collapsed from 56 days (2024) to as little as 10 hours in 2026 (source: Cyberunit), and Mandiant's M-Trends 2026 found 28.3% of CVEs are now exploited within 24 hours of disclosure. Defender remediation time, meanwhile, has stayed flat at roughly 55-61 days for years. (VulnCheck State of Exploitation 1H-2026; Mandiant M-Trends 2026)

Approval is a moment. Access is a movement.



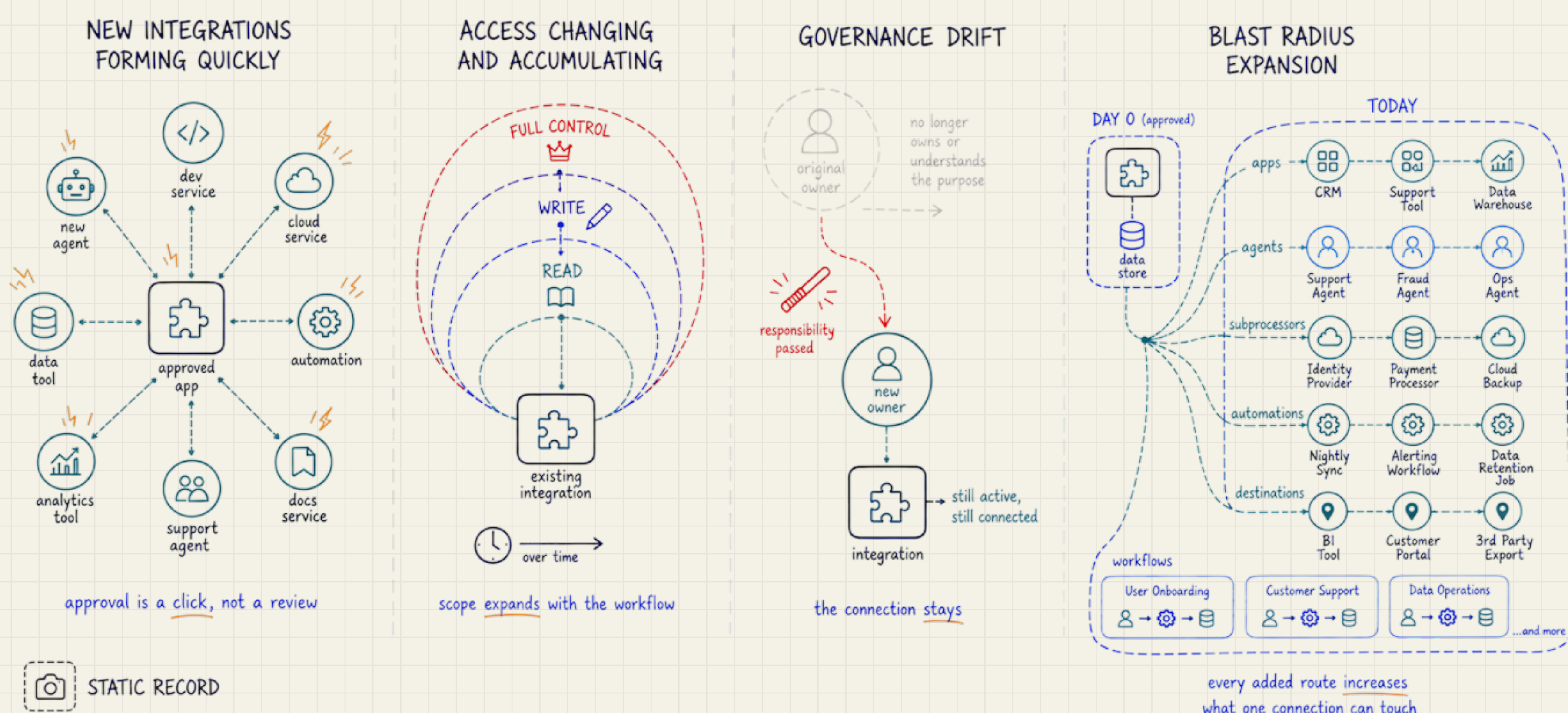
CATCH MY DRIFT?

- Scope drift**
The connection can read, write, or administer more than the original use case required.
- Connection drift**
New apps, agents, subprocessors, automations, or destinations attach themselves to the path.
- Ownership drift**
The person who understood why the connection existed is no longer the person responsible for it.
- Purpose drift**
A connection built for one workflow quietly becomes infrastructure for several.
- Tokyo drift**
A high-speed driving technique of sliding a car sideways around corners.

The smallest vendor can become your biggest risk. Their access to Salesforce, GitHub, or Snowflake can matter more than a much larger vendor with little meaningful reach.

Your vendor record does not calculate exposure nor does your questionnaire detect it.

Static lists don't calculate tomorrow's risk, contextualize activity, or cross-app exposure. A low-risk vendor on paper could be the biggest tinderbox.



Risk is a property of the access path that exists today and tomorrow.

If your monitoring stops at inventory and scope changes, that activity can look perfectly healthy. But if you also go behind the inventory, you can catch structural changes: a new permission, an ownerless account, a dormant connection that never went away.

Boring can quickly become terrifying.

The most worrisome vendor may be the small, boring one that barely does anything in your environment. But their standing access into Microsoft 365, Okta, Salesforce, GitHub, or Snowflake can create a more consequential path than a large vendor with little meaningful access.

So I follow the permissions, not the procurement record.

Review effort naturally concentrates on the large, visible vendors but if we take a look at Klue, that tiny unheard vendor was connected to 9 different surfaces and ended up with a blast radius affecting 200 firms.

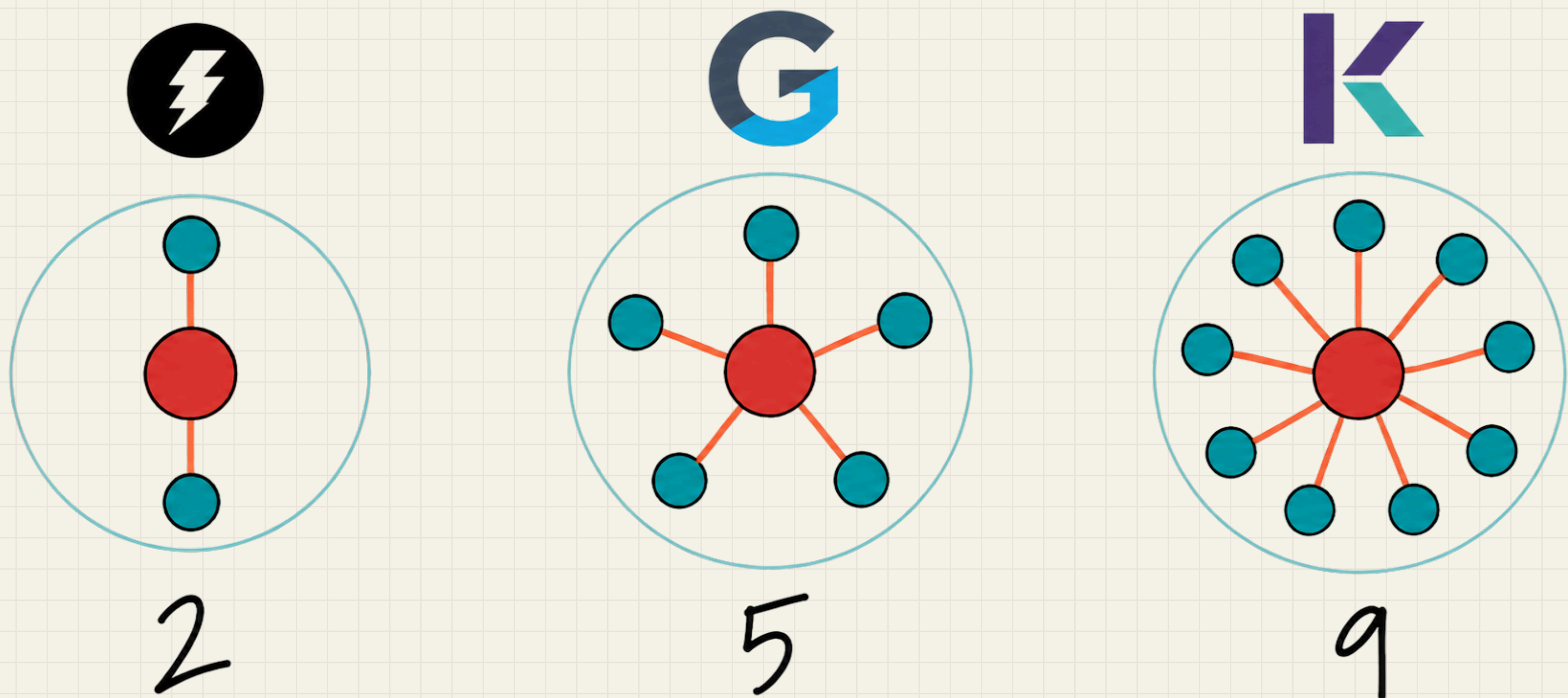
Structural monitoring asks:
Did the connection change?

Behavioral monitoring asks:
Did the way the connection is being used change?

Which category of change do your tools expose?

GET A KLUE:

Inspect your most boring app.



* Pick a vendor outside your obvious top five or top ten.

Go deeper

Look for four capabilities in the system you use to manage third-party integrations:

CONTINUOUS SIGNALS

What is changing over time?

CROSS-APP CONTEXT

What else does this connection reach?

BEHAVIORAL BASELINING

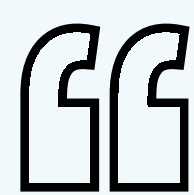
Is it behaving differently than usual?

NETWORK CONTEXT

Is this pattern abnormal only for you, or across multiple environments?

Discover how agents and integrations create hidden risks obsidiansecurity.com/resource/inheriting-access-no-one-is-watching

Watch the webinar on-demand obsidiansecurity.com/resource/webinar-frontier-ai-third-party-integration-risk



Your vendor risk process answers a question about the past. The risk that actually materializes lives in what changed since, and whether anyone was watching when it did.” - Chris Fuller

Use these to calculate the cost of an integration rogue vs securing it. Hardening is the best practice we've always known.

YOUR NEXT STEPS

Gather the evidence for your own environment.

Pick a small sample of integrations or agents and try to answer these questions with the tools you already have:

Specimen

QUESTIONS TO ASK

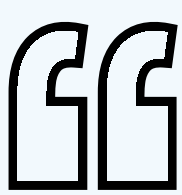
1. What is connected?	Which integrations are active, stale, or no longer owned?
2. What can each one reach?	Trace scopes, credentials, connected apps, sensitive data, and downstream access.
3. How many places did you have to look?	Count the consoles, app owners, tickets, spreadsheets, or teams needed to reconstruct the answer.
	Time it. Then extrapolate.
4. How long did one investigation take?	Manual effort = findings per month × minutes per investigation × analyst cost Surface evidence to show where the operational burden and risk are accumulating.

What to look for in a solution

If this exercise is hard to repeat, that is part of the finding. Look for a way to:

- continuously inventory integrations, agents, and their owners
- connect permissions to the data and apps they can actually reach
- identify stale, over-scoped, or risky access without checking every console manually
- investigate one access path without stitching together evidence by hand
- enforce or remediate the risky action once you find it

Use this to translate your findings into evidence of the risk and to prove how much work it takes to understand and control the risk you already have.

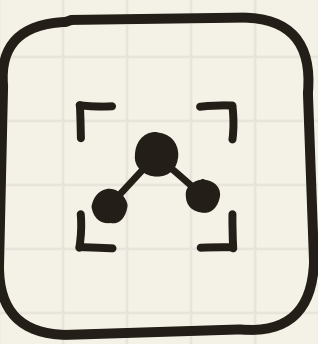


I could see in Obsidian what the vendor was reporting as a potential problem. I didn't know what it meant yet, but I could see it happening. We solved the case in the first five minutes with Obsidian – a very, very clear picture. No joke, probably five minutes."
- [Wyndham Case Study](#)

Work lives in third-party apps. The data is there too.

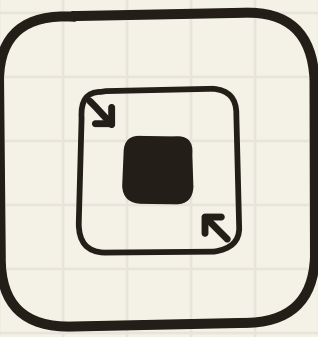
Integrations connect the business around that data.
Those connections keep changing after approval.
Agents move through the same routes autonomously.
Attackers exploit the trust already built into them.

OBSIDIAN SECURES THE CONNECTED ENVIRONMENT AROUND YOUR DATA.



SEE THE CURRENT PATH

Understand which apps, agents, identities, integrations, tokens, scopes, and permissions can reach sensitive data across third-party applications.



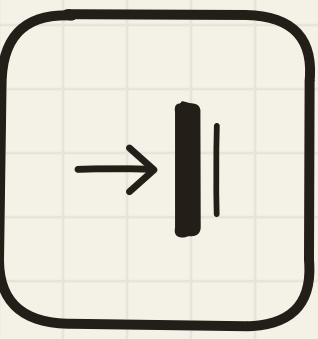
REDUCE THE ATTACK SURFACE

Right-size excessive privilege, standing access, risky integrations, and drift that no longer matches the business purpose.



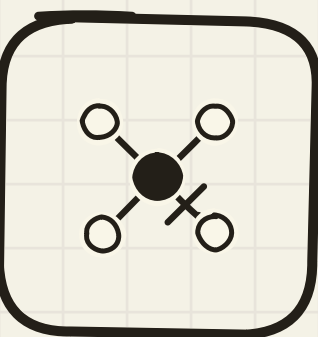
GOVERN THE ACTIONS

Identify activity and allowed actions in your environment from the integration, user, or agent.



ENFORCE THE CONSEQUENCE

Apply policy to risky agent actions at runtime, including blocking or requiring approval before destructive or sensitive actions execute.



CONTAIN THE NARROWEST EDGE

Identify the token, grant, permission, connector, account, or action path that can be cut without unnecessarily breaking the workflow around it.

The goal is not fewer apps, fewer integrations, or fewer agents.
It is to make the environment the business already depends on safe to run.

Apps hold the data.
Integrations connect the work to apps.
Agents move through them.
Attackers follow the trust.
Obsidian shows you the path.

TRY OBSIDIAN FOR FREE