reco

The CISO's Guide to Claude Risk in the Enterprise

A CISO's guide to the six Claude surfaces, the risks each one introduces, and the mitigation strategies that close them.



At the start of 2026, three changes to Claude's footprint happened simultaneously:



1. The Compliance API launched, creating a distinct governance surface.
2. Managed Agents shipped, enabling autonomous workflows at scale.
3. MCP servers went mainstream, giving Claude access to internal systems.

Together, they expanded Claude's attack surface with no governance baseline to match. There's no widely adopted Claude-specific benchmark comparable to a CIS Benchmark, NIST mapping, or formal audit checklist. Every other major infrastructure category your team governs has at least one.

Until one exists for Claude, this guide is designed to bridge that gap.

Claude's Core Attack Surfaces

Claude's risk is spread across six distinct surfaces. Each one has its own identity path, permission model, audit trail, and failure mode. Mapping them separately gives security teams a clearer view of where Claude can access data, retain context, reach tools, execute code, act through delegated identities, and mint the credentials that agents run on. Below, we break down each surface, the primary risks it introduces, and the audit source that covers it.

SURFACE	WHAT IT IS	PRIMARY RISK	AUDIT SOURCE
Claude Enterprise + Connected Apps	Web and desktop chat with OAuth-linked SaaS apps	Sensitive data flowing into prompts and outputs	Audit logs, Compliance API, data exports
Claude Projects	Persistent workspaces with shared files and instructions	Long-lived sensitive context, sharing sprawl	Audit log events, project-level data exports
MCP Servers	Tool servers granting Claude authenticated access	Privileged tool use, prompt injection, supply chain	OTel telemetry, MCP gateway, source app logs

SURFACE	WHAT IT IS	PRIMARY RISK	AUDIT SOURCE
Claude Code	CLI and IDE coding agent with shell, network, and repo access	Repository exfiltration, command execution, secret exposure	OTel, managed-settings.json, Compliance API
Managed Agents	Scheduled, autonomous Claude workflows with tools	Standing access, runaway actions, identity sprawl	Compliance API, IAM logs, SaaS app logs
Claude Platform Console	Admin console where engineering teams manage API keys, workspaces, and deployed agents	API key exposure, unauthorized access, unmanaged deployments	Console audit logs, IAM logs

Claude Enterprise and Connected Apps

The chat surface is the most visible and the most underestimated. With Connected Apps, a Claude.ai user can OAuth into Google Drive, Gmail, GitHub, Slack, Jira, and dozens of other SaaS applications, then ask Claude to act on the data inside them. The visibility gap opens immediately: audit logs capture the connection event and the user identity, but not the prompt or the data Claude reads back.

A finance analyst can pull a quarter's worth of board materials into a chat in seconds, and the audit log will show only that the conversation existed, not what was inside. Conversation content lives in a separate data export channel that only Primary Owners can access, with standard audit logs retained for up to 180 days.

Claude Projects and Persistent Context

Projects extend Claude with a workspace that holds files, instructions, and shared knowledge across sessions. They are valuable, and they are also where context decays into liability. A Project gets spun up around a single context, then keeps living with the data inside it long after the deal has closed.

Projects can also be created with workspace-wide visibility, which means an engineer's Project containing test customer records can become broadly accessible without a data classification step ever running. Custom retention rules at the Enterprise level apply to Projects, but the cleanup is only as good as the inventory. A Project nobody owns is a Project nobody is going to deprovision when the underlying data falls out of scope.

MCP Servers and Tool Governance



MCP is the connective tissue between Claude and the systems it can act on. Every MCP server is a privileged software component holding either a service-account credential or an OAuth token scoped to whatever the user authorized. In effect, MCP expands identity, access control, logging, and application security surfaces by formalizing how AI systems discover and invoke privileged capabilities.

Risks compound across three layers. The server itself can be malicious or vulnerable, with several public CVEs already documented across community-maintained implementations. The token can be over-scoped, granting the agent admin where read-only would do. Prompt injection in tool output can also hijack the agent into executing actions it was never asked to take, with the destination system seeing only a legitimate authenticated request from a trusted token.

Claude Code and Repository Exposure

Claude Code runs in a developer's terminal or IDE with shell access, network egress, and the ability to read and write across the local filesystem. It can clone, push, branch, and commit. It can read .env files, SSH keys, and AWS credentials sitting in plaintext on developer laptops. Anthropic ships managed-settings.json for organization-wide permission defaults, OpenTelemetry export for operational telemetry, and Compliance API coverage for the control plane.

But unfortunately gaps remain: OTel produces event-shaped operational data rather than schema-stable audit evidence, and on a developer laptop there is no native enforcement point for outbound network calls or file access boundaries unless your team operates one. The right pattern combines managed permissions, a sandboxed execution environment, and a network gateway that records every outbound call Claude Code makes.

Managed Agents and Autonomous Access

Managed Agents combine skills, connectors, and subagents to run bounded workflows on a schedule, often with no human in the loop. The blast radius is a function of the connectors and the cadence. An agent that runs every hour with write access to Salesforce, Slack, and a data warehouse holds standing privileges a human user never would. When something goes wrong, the failure mode is hundreds of automated actions executed before anyone reads the alert.

Agent identity governance is the new identity governance, and most security teams have no inventory yet. The agent inventory question goes beyond which agents have been formally deployed. It includes which agents are running across your SaaS environment right now, including the ones a line of business stood up without telling IT.

Claude Platform Console and API Key Exposure

The Claude Platform Console is where engineering teams manage API keys, workspaces, and the agents they build and deploy. It sits outside the chat surface entirely, which is why it gets missed: most security tooling watches Claude Enterprise, while the Console is where API key exposure and unauthorized access typically start. A key created six months ago and never rotated can sit in a script connected to three other systems, holding live access nobody remembers granting.

Deployment is the other half of the problem. Agents built through the Console inherit whatever scope their keys carry, and neither the chat surface nor the Console itself captures what those agents do once they are running. Visibility here means an inventory of workspaces, keys, and deployed agents, tied to the identities that created them.

6 Claude Risk Mitigation Strategies for CISOs

Each surface above has a mitigation that closes its riskiest path. The six strategies below map one to one, in the same order, and together they give enterprises the controls to enable Claude at scale.

1 Wire Up the Compliance API on Day One

Given Claude Enterprise's distributed audit surface, the Compliance API needs to be wired up on day one. It exposes audit log events programmatically, supports selective deletion as its own logged event, and integrates with SIEM, SOAR, and DLP pipelines. Any compliance program treating Claude as a single SaaS application with one log source will under-cover the surface area dramatically.

Treat the OAuth-mediated admin keys like any other service-account credential: vault them, rotate them, alert on use. In practice, pair the Compliance API control plane with OpenTelemetry from Claude Code for the agent operational plane, and a network gateway or LLM proxy for the tool plane. That way you get three layers of telemetry and one audit story.

2 Govern Claude Projects Like Shared Drives

Projects are durable assets, and the cleanest way to govern a surface growing weekly is to make Project creation an attested step that captures owner, classification, and lifecycle up front. Maintain an inventory of every Project in the workspace, including its owner, its data classification, its retention setting, and its sharing scope.

Run periodic content reviews on long-lived Projects the way you run access reviews on shared drives. Default to least-context: if a Project does not need a file, the file should not be in the Project. This prevents context decay from becoming a slow-growing data liability across the workspace.

3 Allowlist and Monitor MCP Servers

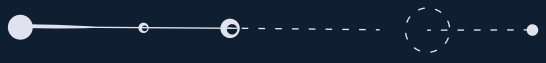
MCP servers should be reviewed with the same rigor you apply to any third-party SaaS application. They hold privileged credentials and act on Claude's behalf, which puts them in the same risk class as any integration touching corporate data. Many security teams use the OWASP MCP Top 10 as a baseline, layering their industry's regulatory controls on top.

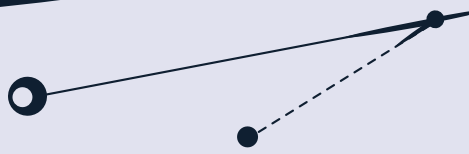
Maintaining an approved server registry tied to a procurement and security review workflow ensures new servers are vetted before they are installed. Enforcing the registry at the network layer through an MCP gateway or egress proxy is also wise: unapproved servers fail closed rather than relying on developer self-discipline.

4 Govern Claude Code Sessions

Claude Code's permission model lives in managed-settings.json, which can be deployed via MDM to developer endpoints. The repository holding your customer PII should not be the repository where you discover, mid-incident, that Claude Code had unrestricted outbound HTTPS.

Start with a default-deny posture and explicitly allow only the safe, frequent commands. Risky operations like shell execution, outbound network calls, and writes to sensitive files should require confirmation rather than run automatically. You can also run Claude Code inside a sandboxed environment where possible, so an over-permissioned session cannot reach the rest of the developer's machine or the wider network.





5 Inventory and Score Managed Agents

Agents are non-human identities with standing access. The governance you would apply to a service account, applied to an agent that speaks natural language and writes to Slack, is the right starting point. Every Managed Agent needs an entry in your AI agent inventory: what it does, who owns it, what connectors it holds, what data it touches, what schedule it runs on, and what its compromise blast radius would look like.

Score agents by privilege scope and data sensitivity, then enforce least privilege at the connector level (for instance, an agent triaging Salesforce tickets with write access across the entire org should score higher than an agent posting weekly campaign summaries to Slack). Apply identity threat detection to agent activity the same way you apply it to human identity. When an agent's behavior deviates from baseline, the alert needs to land in a queue someone reads in business hours.

6 Govern Platform Console Keys and Roles

Treat every API key issued through the Platform Console as a service-account credential: inventory it, scope it to least privilege, rotate it on a schedule, and alert on anomalous use. Key creation should be tied to an owner and a documented purpose, so a key found in a script a year later traces back to a person and a decision.

Apply the same discipline to Console access itself. Role governance, SSO enforcement, and IP restrictions limit who can mint keys and deploy agents in the first place, and workspace-level separation keeps a compromised development key away from production data. Review deployed agents alongside the Managed Agent inventory from strategy five, since an agent built through the Console carries the same standing access with less native oversight.

How to Operationalize Claude Governance With Reco



The six mitigations above all assume an underlying capability: continuous, accurate visibility into where Claude is operating, what it is touching, and which identities are involved. Most security teams have parts of this picture in different tools, with no way to correlate them. Reco's agent and SaaS security platform was built for exactly this kind of problem. The platform discovers, prioritizes, and remediates across the full Claude footprint inside your third-party ecosystem.

Build the Claude Inventory

Reco discovers every Claude surface across your environment automatically, drawing on seven independent discovery sensors. It reads endpoints through CrowdStrike, Defender, or SentinelOne, network traffic through SASE, and browser activity through Reco BrowserGuard, then adds SSO logs, social logins, connected plugins, and shadow email.

Upon deployment, it quickly uncovers all Claude.ai accounts, Claude Enterprise workspaces, Connected Apps, Projects, Claude Code installations, and the Claude Platform Console through their OAuth and integration footprints, then classifies every account it finds as sanctioned, shadow, or AI agent.

This means you get a live source of truth for where Claude is operating across your environment, refreshed continuously as users connect new applications or stand up new Projects. By shining a light on shadow Claude usage, Reco helps you close inventory gaps that used to take quarters to resolve manually.

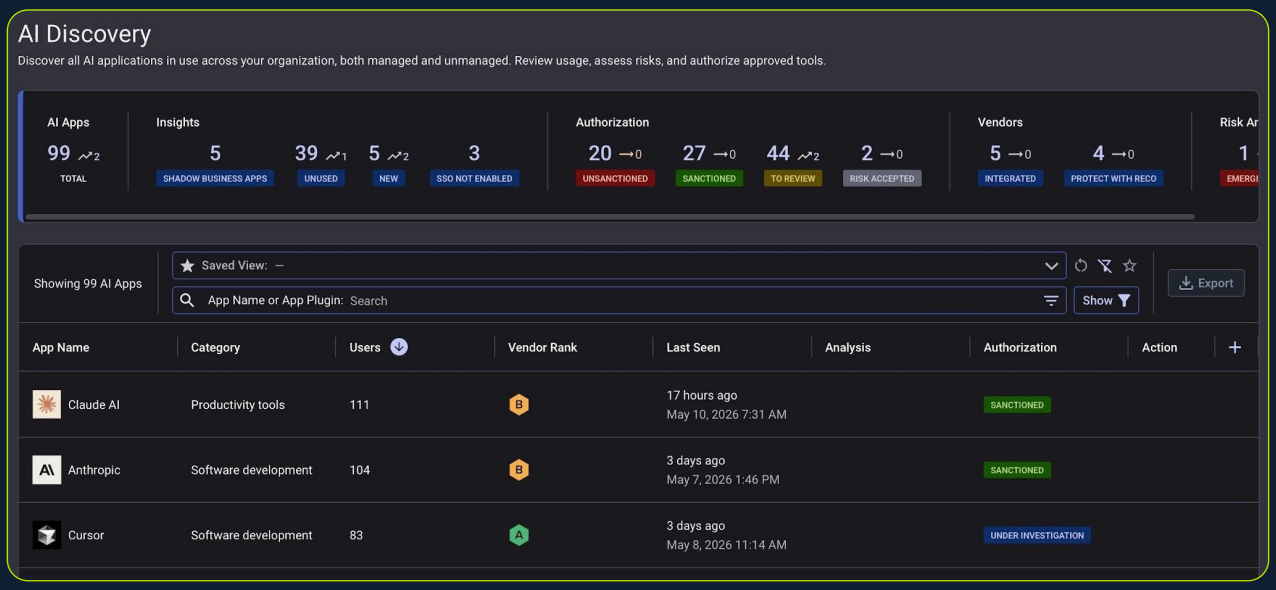


Figure 1: Reco's AI Discovery Dashboard, showing Claude AI and Anthropic among sanctioned applications.

Map Identities and Permissions

For every Claude identity, Reco maps which agents and applications it connects to, what permissions it holds, and what data it can reach. Powered by Reco Graph, it automatically surfaces toxic combinations where a single identity bridges sensitive systems through Claude, OAuth, MCP, or API integrations.

It illuminates the hidden web of Claude connections in a way no manual audit could. With this context, security teams can instantly understand the scope of access any Claude identity has and how it ties to users and data.

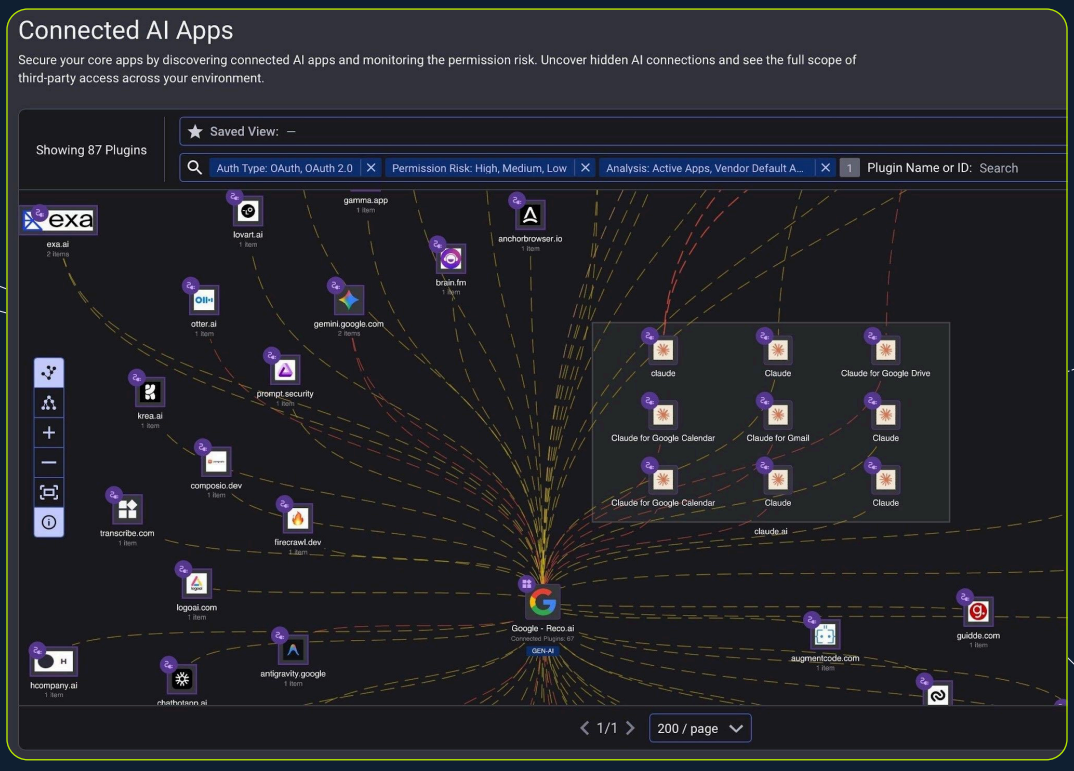


Figure 2: Reco Graph mapping connected AI applications, including Claude variants.

Govern Tools, Agents, and Context

Reco's AI agents inventory captures every Claude agent, MCP server, and embedded AI feature operating across your third-party ecosystem. This means you get visibility not just into sanctioned agents, but also into the ones that are stood up without IT involvement.

For each agent built on Claude, Reco maps the model it runs, every MCP server it connects to, and every tool it can invoke, then evaluates the risk in context: the agent, its owner, the owner's status, the applications it touches, and the blast radius if something goes wrong.

From there, Reco helps right-size access through identity and access governance, directly limiting what an attacker can do if an agent or its session is compromised. Project sprawl, agent sprawl, and shadow MCP all flow into the same governance loop with the same primitives.

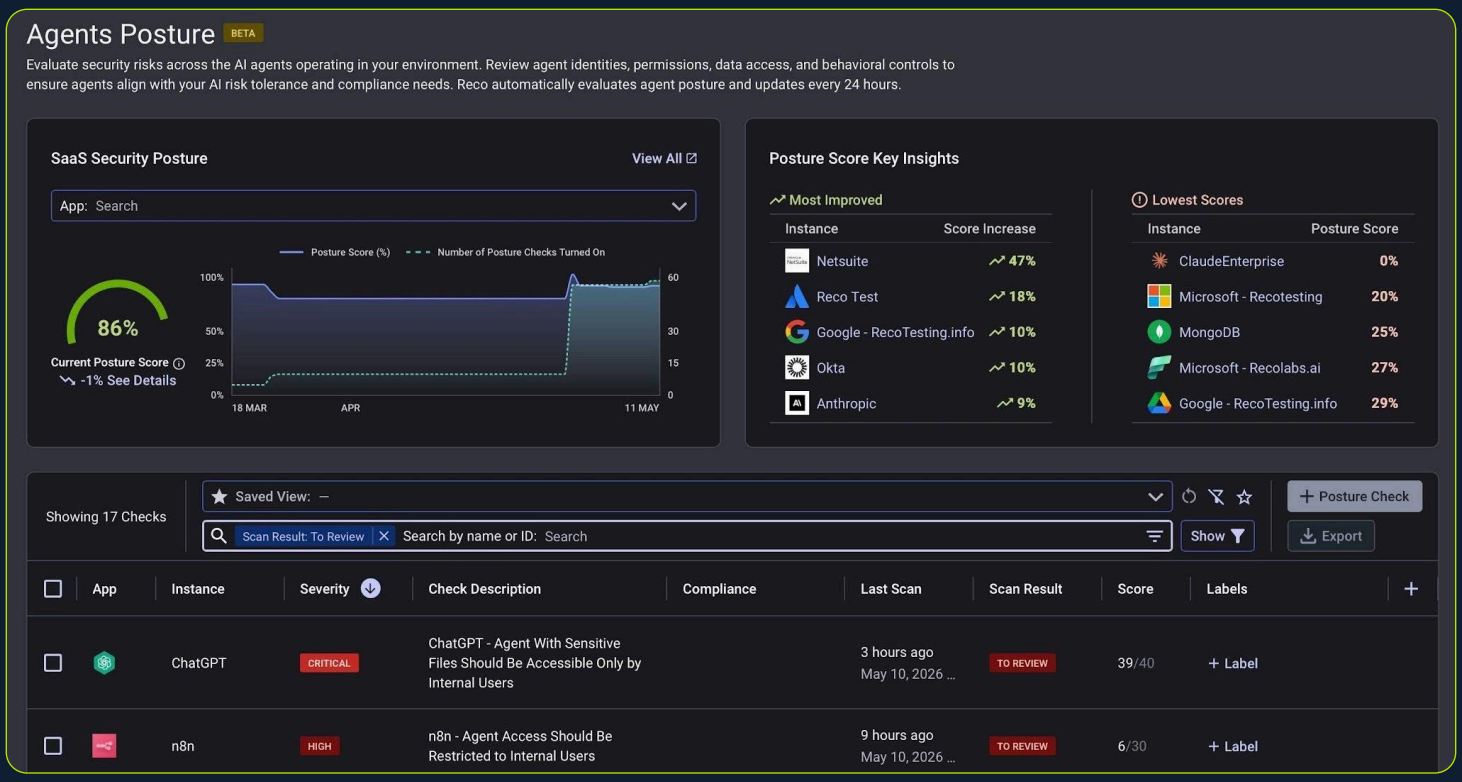


Figure 3: Reco's Agents Posture Dashboard surface scoring agent governance.

Classify Data and Identity Risk

Reco classifies the data and identities involved in each connection by sensitivity and risk. Sensitive data flowing into Claude through Connected Apps, Projects, or MCP servers is tagged against your organization's data classification scheme, while the identities driving those flows are scored on privilege level and behavior history.

This means risk findings come tied to specific data classifications and identity contexts, rather than surfaced as undifferentiated alerts. With this classification layer in place, security teams can prioritize the connections that actually matter and treat the rest as expected business activity.

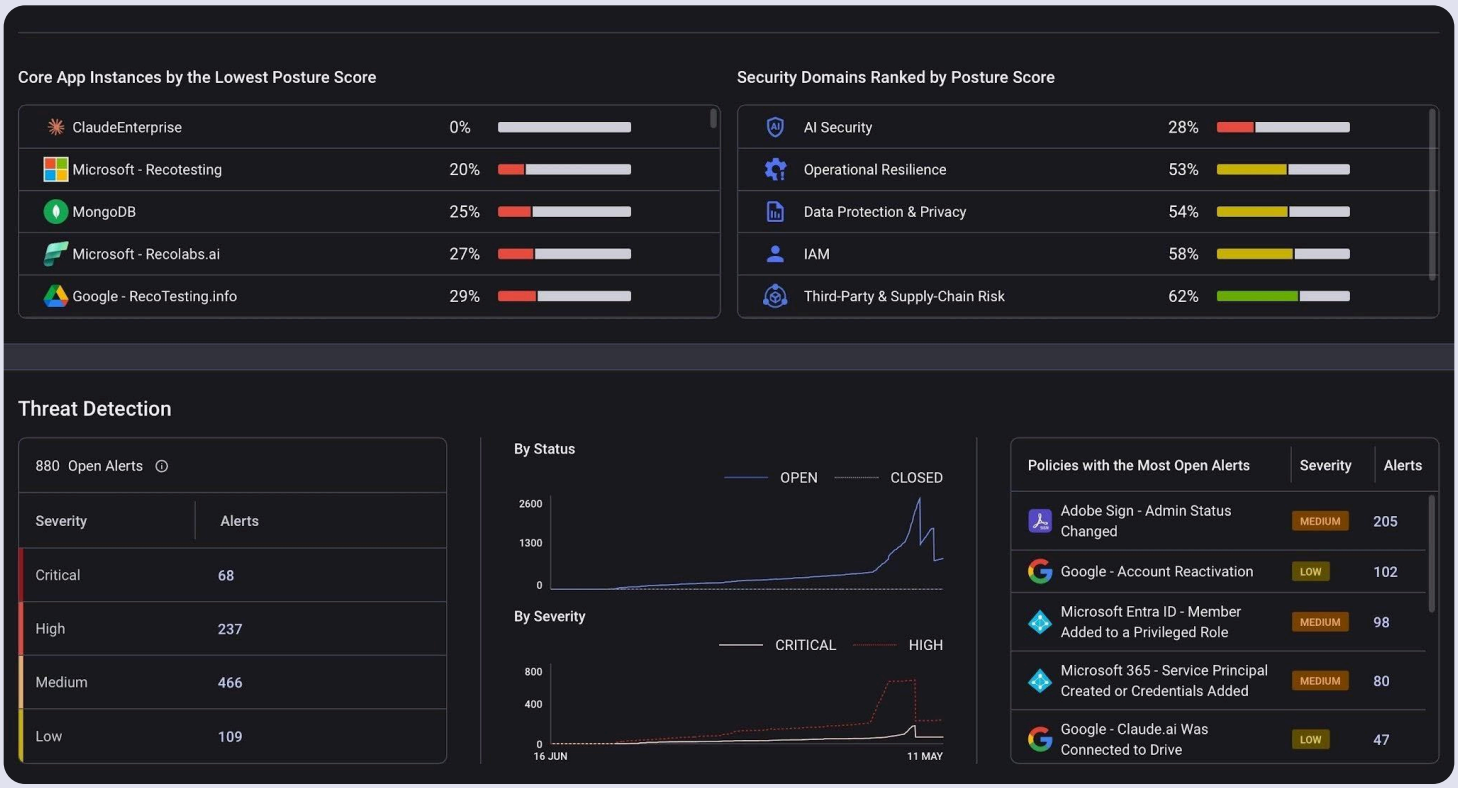


Figure 4: Reco ranking app instances by posture score and classifying alerts by severity.

Detect Exposure and Prove Control

Reco's threat detection engine applies identity-centric behavioral analysis to Claude agents the same way it does to human identities, distinguishing normal Claude automation from suspicious deviations in real time.

The engine ships with 46 detection policies and a real-time feed of every action in Claude Enterprise, and each alert carries full context, remediation steps, and MITRE ATT&CK mapping.

When the inevitable audit lands on your desk, the evidence is already organized: Project lifecycle, agent inventory, MCP allowlist enforcement, and identity exposure all rendered against the same Graph view.

The integration also runs in the other direction: Reco exposes its own MCP server, so Claude can query the identity graph, posture findings, and threat alerts in natural language, and an analyst can pull offboarded users still holding access, open findings by severity, or the last 30 days of MCP server changes without leaving the chat.

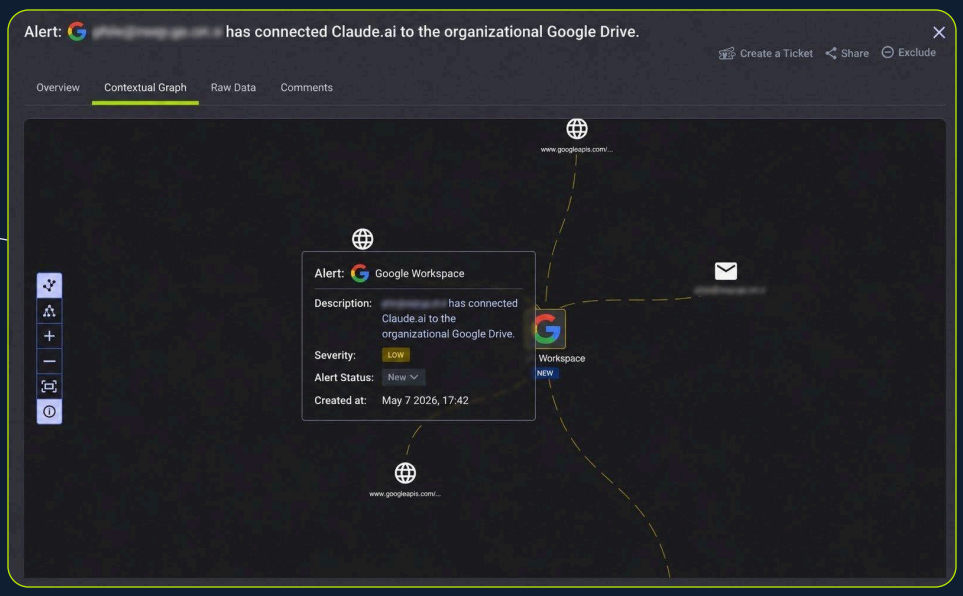


Figure 5: A Reco alert detecting a user connecting Claude.ai to the organizational Google Drive.

The Most Frequently Asked Questions

These are the questions that come up most often when CISOs and security architects start mapping their Claude footprint against their existing controls.

Do we need separate controls for Claude?

Yes. Claude's enterprise footprint extends across MCP servers, Claude Code, Projects, Managed Agents, and the Claude Platform Console, each with its own identity model and audit story. Controls designed for chat-surface DLP miss most of where Claude actually operates inside an enterprise, and the Compliance API and OTel streams unique to Claude carry telemetry your existing OpenAI controls have no equivalent for.

How is Claude Enterprise different?

Claude Enterprise covers Claude Code under a single agreement and includes the Compliance API, configurable retention, SCIM, audit logs, and SSO. The plan exists because deploying a frontier model across a workforce raises governance and identity questions that individual or team plans do not carry. Treating Claude Enterprise as just a chatbot underprices both the risk and the available controls, especially the Compliance API, which is the most useful governance primitive in the entire stack.

Why govern Claude Projects?

Projects hold persistent context, files, and instructions that often outlive their original business purpose. Without a lifecycle policy tied to ownership, retention, and data classification, Projects accumulate sensitive content over months and become a slow-moving data exposure surface no single user is monitoring. The governance overhead matches the level you would apply to a shared drive or wiki space, with explicit owners, retention policies, and periodic content reviews.

Why are MCP and Claude Code higher-risk?

Both grant Claude authenticated access to systems holding real privilege. MCP servers carry tokens scoped to SaaS data; Claude Code carries shell access on a developer's machine. A single over-scoped token or a single prompt injection in tool output can produce actions that look legitimate at the destination system. Audit evidence is fragmented across the Compliance API, OTel streams, and the source system's own logs, so reconstructing what happened during an incident takes correlation across three telemetry sources rather than one.

How should we govern Managed Agents?

Treat every agent as a non-human identity with standing access and apply the governance you would to any service account: ownership, inventory, scope review, behavioral monitoring, and lifecycle. Agents act in natural language and across multiple systems at once, so identity threat detection and connector-level scope reviews matter more than they do for traditional service accounts. The right starting point is an inventory tied to ownership, then connector-by-connector scope review, then behavioral baselining once the agent has been running for a few weeks.

Take Your Next Steps With Reco

Most enterprises will discover their Claude governance gaps the hard way: an audit finding, a leaked secret, an agent that wrote a Slack message it should not have. The teams that avoid those moments are the ones putting governance in place now, before the surface area expands again.

Reco gives you that governance across every Claude surface: a live inventory, an identity and permission map, risk classification, agent oversight, and threat detection tied to audit-ready evidence. Catching exposure, proving control, and right-sizing access stop being separate workstreams and become the same loop.

Reco is listed by Anthropic as a security integration for the Claude Compliance API.

To see Reco in action, request a **demo** today.

Request a Demo

© 2026 Reco. All rights reserved.