AI AGENT SECURITY FOR CLAUDE

Govern what Claude agents can access and do

Claude Code and Cowork act with the access of the person behind them. Obsidian scopes that access down, then checks each tool call against your policy before it runs, using available identity, HR, and application context and your own data labels.

01

Find and scope Claude agents

See the MCP servers, skills, and plugins each agent uses, and cut access its work doesn't need.

02

Decide each tool call

A sidecar checks the call against your policy before it reaches the tool or app.

03

Prove what policy stopped

Each decision is logged with the agent, user, tool, arguments, and matching policy.

THE PROBLEM

Claude agents act in seconds, with everything their user can reach

Claude Code and Cowork work through the user's credentials, with every permission that account holds in apps such as Salesforce and Snowflake. Users connect them to company apps and MCP tools, often without a security review. A mistaken or misled agent can take any action its user could.

- **Destructive actions run first.** A finding reviewed the next morning arrives after the table is gone.
- **Secrets come into reach.** Claude Code can read SSH keys on a laptop into its context.
- **Shadow agents skip review.** Developers connect Claude to apps and MCP servers on their own.
- **Data crosses apps.** Cowork can query a Snowflake table and post the results to Slack.

THE SECURITY GAP

A valid login doesn’t settle whether an action should run

Identity providers decide who can sign in, and app permissions decide what an account can do. Neither knows that the user behind a Cowork session left the company this morning, or that the file the agent wants carries your Highly Sensitive label.

YOUR CONTROLS STAY IN PLACE. HERE’S WHAT OBSIDIAN ADDS.

CONTROL YOU RUN	WHAT IT ESTABLISHES	WHAT OBSIDIAN ADDS
Identity provider and SSO	Authenticates the user and enforces sign-in policy.	Checks each Claude tool call with identity and employment context, such as a Workday termination.
Permissions in third-party apps	Decide which operations the user’s account can perform.	An action-level check, so a DROP TABLE the account may run can still be blocked for a contractor.
Anthropic’s built-in controls	Apply controls to Claude agents and their actions within Claude.	HR and cross-app context, under the same policies you set for Codex and Microsoft Copilot.
MCP and API gateways	Intercept and authenticate agent traffic, and enforce their own rules.	The MCP Security Proxy checks MCP calls to supported apps. The Policy Decision API returns decisions to your gateways.

- Obsidian adds a decision made with the person, agent, credential, tool, app, and data in view.

TWO CONTROLS THAT WORK TOGETHER

GOVERNANCE

Narrow the access first

- ✓ Excessive-access findings, with risk factors scored by criticality
- ✓ Agents that can read cloud credentials or private keys, and skills with embedded secrets
- ✓ Inventory of each agent’s models, skills, hooks, plugins, and MCP servers

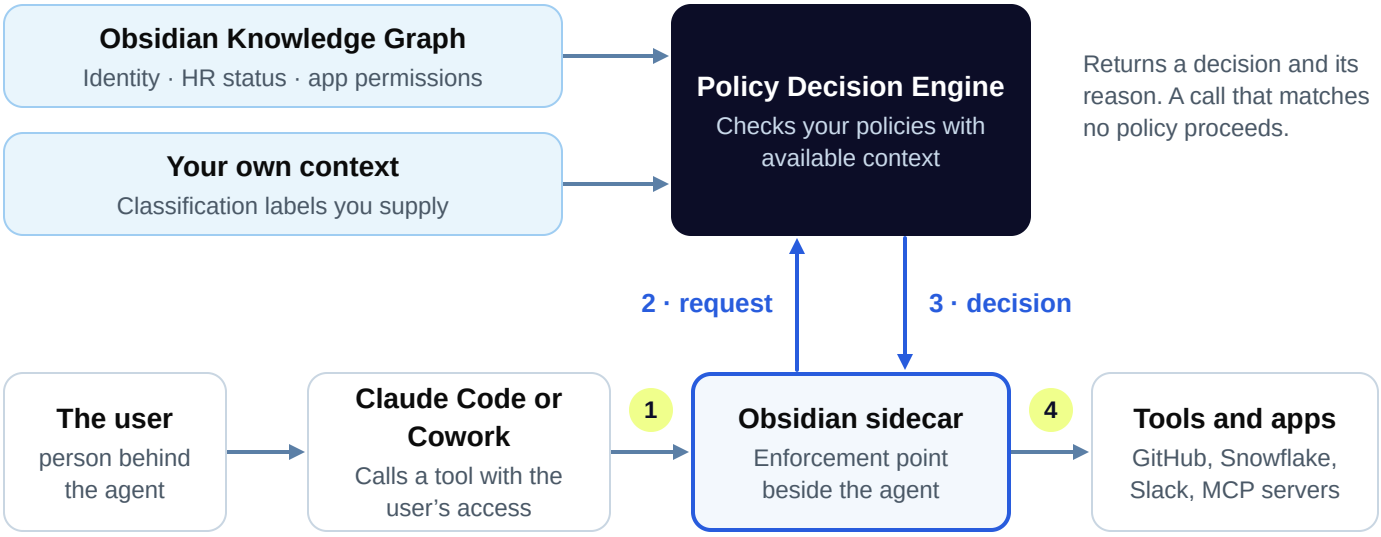
RUNTIME POLICY

Decide the actions that remain

- ✓ A sidecar checks each tool call before it runs, tested first with Flag
- ✓ New: your own data labels, such as Highly Sensitive, in each decision
- ✓ New: the same policies now reach OpenAI Codex, from one console

HOW IT WORKS

A sidecar asks the Policy Decision Engine before each Claude tool call runs



1. Claude calls a tool, and the sidecar pauses the call with the agent, user, tool, and arguments.

2. The sidecar sends the call to the Policy Decision Engine.

3. The engine checks your policies with available context and returns a decision and its reason.

4. The call proceeds or stops, or waits for approval in Claude Code, and the decision is logged.

CLAUDE CODE AND COWORK

One sidecar model and one set of policies, with different outcomes

	CLAUDE CODE	COWORK
Discovery and governance	Available since August	Available since August
Runtime enforcement	Sidecar checks each tool call before it runs	Sidecar checks each tool call before it runs
Policy outcomes	Flag, block, or require approval for policies that run before an operation	Flag or block
MCP servers	Inventoried, with calls to unsanctioned systems blockable	Inventoried, with calls to unsanctioned systems blockable

In Claude Code, the user sees a message the policy author wrote, so a developer knows why an action stopped. A call that matches no policy proceeds.

CONTEXT IN ACTION

A contractor’s agent tries to drop a critical table. The policy stops it.

THE POLICY

Contractors’ agents can’t drop tables labeled critical

1	A contractor asks Claude Code to tidy up a Snowflake environment.	Claude Code
2	The agent calls DROP TABLE on a table your team labeled critical.	DROP TABLE
3	The sidecar asks Obsidian, which sees what Snowflake can’t.	Contractor, from HR Critical, your label
4	The table stays, and the contractor sees the policy author’s message.	Blocked

Illustrative example. Snowflake sees a valid account, with no view of HR status or your labels.

EVALUATE

Four questions to ask any vendor

01 If a user’s employment ends in Workday this morning, does the next agent action reflect it?

02 Which outcomes can each coding agent return: flag, block, or require approval?

03 Does one policy cover Claude Code, Cowork, and Codex, or is each one rebuilt?

04 What does the log record when an agent action is flagged or blocked?

A question a vendor can’t answer clearly is a gap your team covers by hand.

See a Claude agent action decided in context

Request a demo to see how Obsidian decides what a Claude Code or Cowork agent does before the action reaches your apps.

Request a Demo obsidiansecurity.com/get-a-demo

Scan to book a demo