AI AGENT SECURITY FOR OPENAI CODEX

Keep Codex agents inside your security policy

Codex runs with your developers' access to code, data, and production systems. Obsidian finds Codex agents and what they can reach, then checks each tool call against the same policies you apply to Claude Code, Cowork, and Microsoft Copilot.

01

See Codex agents and their reach

Inventory the models, tools, and MCP servers Codex uses, and flag excessive access.

02

Stop risky tool calls before they run

A sidecar beside Codex allows or blocks each call under your policy.

03

One policy set across coding agents

Codex follows the policies you already set, from one Obsidian console.

THE PROBLEM

Codex works with the developer's access, and the risky moments happen mid-session

Codex runs on the credentials of the developer or account behind it, whether or not anyone approved that scope for an agent. Most of a session is routine. The few actions that aren't can each be fully authorized.

- **A destructive command.** Codex drops a table or deletes records the account is allowed to delete.
- **A read far outside the project.** Codex opens a file nowhere near the repository it was asked to work on.
- **A secret in context.** A credential or private key gets pasted into the agent's context.
- **Adoption without review.** Developers connect Codex to apps and MCP tools on their own, often alongside other coding agents.

THE SECURITY GAP

Account permissions can’t tell a routine command from a risky one

An identity provider authenticates the developer, and app permissions decide which operations the account can perform. Neither decides whether this action, taken by an agent right now, should run. That depends on who’s behind the agent, what it can reach, and what the data is worth.

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 developer and enforces sign-in policy.	Checks each Codex tool call with identity and employment context, such as a Workday termination.
Permissions in apps and databases	Decide which operations the developer’s account can perform.	An action-level check, so a DROP TABLE the account may run can still be blocked when policy says so.
OpenAI’s built-in Codex controls	Apply controls to Codex and its actions within Codex.	HR and cross-app context, under the same policies you set for Claude Code, Cowork, and Copilot.
Rules copied into each tool	Let each platform enforce its own version of a rule.	One policy set in one console, decided for each supported enforcement point.

- Obsidian works alongside Codex’s built-in controls. It doesn’t replace them or the sandbox Codex runs in.

FOUR SEPARATE JOBS

Discover

Which Codex agents exist, and the models, tools, and MCP servers they use.

Analyze risk

Excessive access, risk factors scored by criticality, and unsanctioned connections.

Decide and enforce

Whether a tool call runs. The sidecar applies allow or block first.

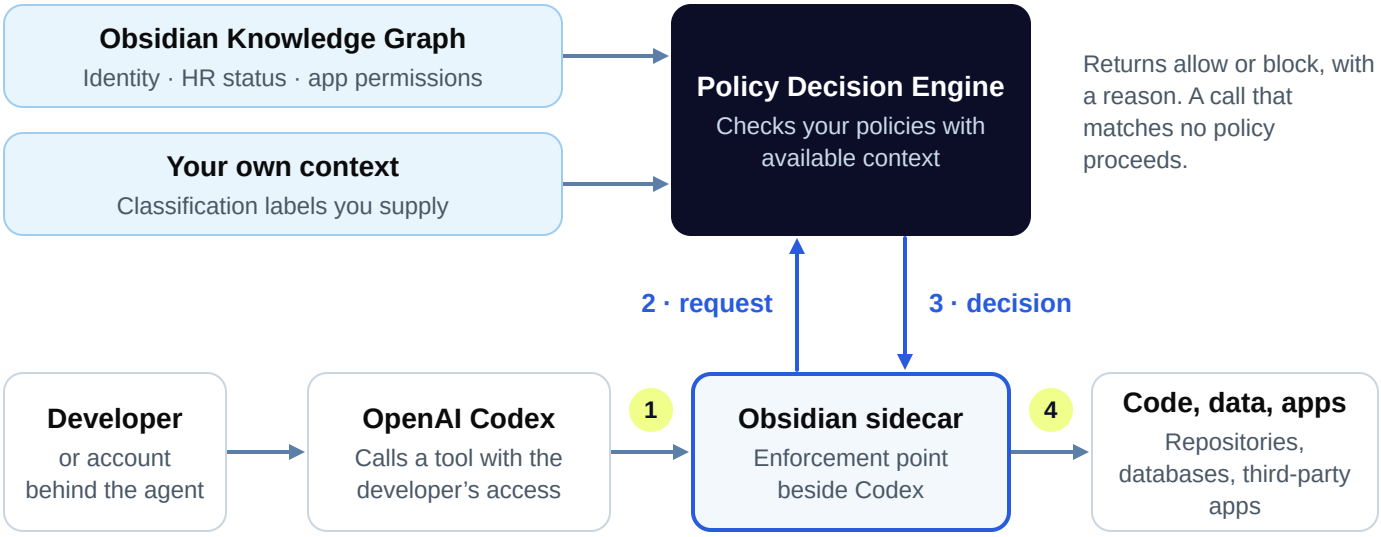
Record

Each decision, with the agent, developer, tool, arguments, and policy.

Discovery and risk analysis don’t stop an action on their own. Enforcement applies to Codex sessions where the Obsidian sidecar runs.

HOW IT WORKS

For Codex, the enforcement point is the Obsidian sidecar running beside it



1. Codex calls a tool, and the sidecar pauses the call with the agent, developer, tool, and arguments.

2. The sidecar asks the Policy Decision Engine for a decision.

3. The engine checks your policies with available identity, HR, and app context and your labels.

4. The call proceeds or stops, and the decision is logged.

POLICY OUTCOMES FOR CODEX

OUTCOME	WHAT HAPPENS TO THE TOOL CALL
Flag	Recorded on real Codex traffic without being stopped, so a team can test a policy before enforcing it.
Block	Stopped before it runs. The decision and its reason are logged.
No match	Proceeds, so routine work continues.
Require approval	Not available for Codex today. It's available where the enforcement point supports it, which today means Claude Code policies that run before an operation.

Two other enforcement points apply to agents generally, not to Codex specifically: the MCP Security Proxy for MCP tool calls to supported apps, and the Policy Decision API for gateways and harnesses you run. Both use the same policies.

CONTEXT IN ACTION

Each decision turns on a fact the coding agent can’t see

WHAT OBSIDIAN KNOWS	DECISION	WHY THE ENFORCING TOOL ALONE MISSES IT
The developer was terminated in Workday at 9am, and a Codex session on their credentials is still running	Block the next tool calls	The session is valid, and the agent runtime has no HR data
HR lists the developer as a contractor, and the table Codex tries to drop carries your critical label	Block	Snowflake sees a valid account, with no view of HR status or your labels
Your label marks a file Highly Sensitive, and the developer is outside the Executive team	Block the read	The app enforces account permissions, with no view of your labels

Illustrative examples.

EVALUATE

Four questions to ask any vendor

01 If a developer’s employment ends in Workday this morning, does the next Codex tool call reflect it?

02 Which Codex sessions does enforcement cover, and what happens when a policy matches?

03 Can one policy apply to Codex, Claude Code, Cowork, and Copilot without being rebuilt?

04 What does the log record for each flagged or blocked Codex tool call?

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

Decide what Codex does before it acts

Request a demo to see how Obsidian decides a Codex agent’s action before it reaches your code, data, and third-party apps.

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

Scan to book a demo