

KONVU GUARDRAILS

Your security guidelines, in every coding agent.

Steered at the edit.

Enforced at the pull request.

More code, from more people who never read your security guidelines. **Konvu Guardrails** maps them into Controls from your standards and your code. Your coding agent gets the right one at the edit, and a PR check confirms it held before merge.

Not a scanner. **Your guidelines, not a generic checklist.**

— The guidelines exist. Nobody writing the code reads them.

Engineers ship far more with coding agents. PMs, designers, data people, and support engineers open pull requests too. A new engineer skims the guidelines once, a PM prompting Claude Code never sees them, and the agent does not know they exist. Review capacity is flat while change volume grows.

WHERE THE GUIDELINES LIVE TODAY

Wikis and architecture docs

Every read is scoped to the caller's organization. Written for people, read once at onboarding, never by a check.

Threat models and pentest reports

Trust boundaries, controls, and the findings that broke them. Filed after the review, not at the next change.

Past findings

The IDOR from the last pentest, the bug bounty report, the exploit Konvu reproduced. Each fix implies a rule nobody wrote down.

The AppSec team

The engineer who knows why that route is admin-only, and reviews every change to it by hand.

WHAT A GENERIC CHECK KNOWS

- Parameterize a SQL query
- Escape output before rendering it
- Validate and bound untrusted input
- Flag a known dangerous API

WHAT ONLY YOUR GUIDELINES SAY

- Every read is scoped to the caller's organization
- Role changes are admin-only
- Refunds keep their approval step
- Outbound webhooks go through the signing helper
- No new public route without the session middleware

The first list ships in every scanner and every coding model. The second list is yours. Guardrails turns it into checks.



Konvu is in production at Fortune 500 and Nasdaq-listed companies.

SOC 2 Type II certified. Code is never stored by Konvu or model providers, and never used for training.

[KONVU.COM/GUARDRAILS](https://konvu.com/guardrails)

Mapped from your code. Steered in the agent. Enforced at the PR.

Guardrails reads the repository with your standards, threat models, and pentest reports, maps the Controls behind your code, steers each edit where the code is written, and enforces before merge.

01

Read the code and context

The repository plus what you share: architecture docs, threat models, pentest reports, past findings. Guardrails maps assets, actors, trust boundaries, and the controls behind them.

02

Map the Controls

Concrete, checkable Controls: every read of a report is scoped to the session organization, only an admin changes a role. Each links to the code, document, or finding it came from.

03

Edit, add your own

Edit the ones that read wrong, reject the ones you disagree with, add the rules only your team knows, in plain language. Both kinds are enforced the same way. The rules stay yours.

04

Steered in the agent

When an edit touches a Control, a local hook stops it once and gives the agent in Claude Code, Codex, or Cursor that Control and its evidence, before the file is written.

05

Enforced at the PR, decided

A PR check runs the same Controls on every pull request, report-only or blocking, with no agent connected. Each finding is fixed, allowed, or denied, and the decision is recorded on the pull request.

The same change, with and without the rule

Prompt to the agent, from a PM in Claude Code: add an authenticated API route that returns a report by ID

Report route returns another tenant's data

GET /api/reports/:id · Next.js route · tests pass

```
const report = await db.report.findUnique({
  where: { id: reportId }, // no tenant constraint
});
```

Authenticated, handles missing records, returns the report. Any signed-in user who guesses an ID reads another tenant's report. No dangerous call, and no scanner rule for it unless someone wrote one. The guideline says reports are tenant-owned. It never reached the person, or the agent, writing this route.

Authentication	present
Functional test	passes
Tenant scope on the read	missing ← decides it

cross-tenant IDOR · guideline never reached the change

Reaches review as valid code

WITHOUT THE RULE

Same route, scoped to the session tenant

control tenant-scoped-reads · mapped · active

Control. Every read of a tenant-owned report must be scoped to the tenant in the authenticated session.

```
const report = await db.report.findFirst({
  where: { id: reportId, tenantId: session.user.tenantId },
});
```

The hook recognised that the edit touches tenant isolation, stopped it once, and handed the agent this Control with its evidence. The agent scoped the query before the change reached review. The PM never had to know the rule. The PR check confirmed it on the pull request.

Mapped from	code · threat model
AppSec review	optional · edit or reject
CI check on the PR	holds
konvu · tenant-scoped-reads	Merged

WITH GUARDRAILS

Every check ends in a decision:

HOLDS

BREAKS A RULE

NEW ACCESS · ALLOW OR DENY

Fix a break. Allow new access to record it as a rule, or deny it. Recorded on the pull request.

The same rules, in either place or both

In the coding agent

Claude Code, Codex, Cursor, GitHub Copilot and other coding tools. A local hook matches each edit, with no model on the path, stops the first edit to covered code once with the guidance, and the agent corrects course before review. Whoever prompted the change.

In CI

The same rules run on every pull request with no coding agent involved, which covers code written by hand. Start in report-only mode, then block violations for the repositories and rules where a violation must not merge.

Hooks and CI are independent. Run CI with no agent connected, run the hooks with no CI step, or both. Guardrails runs alongside your coding agents. Your scanner stays.

Where the current tools stop.

Guardrails runs alongside the tools your team already uses. What each one covers, and the gap it leaves on a pull request.

01

Code review and CODEOWNERS

CODEOWNERS routes the pull request to the people who know the guideline. It cannot tell them what to look for, and their capacity is flat. Guardrails writes the guideline down with its evidence and checks every pull request the same way, whoever reviews it.

02

SECURITY.md, AGENTS.md and rules files

Advice the agent may ignore. Written by hand, loaded into every session whether relevant or not, stale as soon as the application moves, and nothing checks it was followed. Guardrails maps the Controls from the application, supplies only the one a change touches, and checks them in CI whether or not the agent read anything.

03

SAST custom rules

Semgrep taint mode and CodeQL can express a missing tenant check, if someone writes the rule for each service and keeps it current. Guardrails maps that rule from the application and keeps it current. It complements the scanner and does not replace it.

04

AI code reviewers

Reviews each diff cold, without your threat model or the Controls your code relies on. The Guardrails PR check reviews against your mapped Controls, names the evidence, and records what was decided.

Add the rules only your team knows, in plain language.

Custom rules are steered and enforced exactly like mapped Controls, in the coding agent and in CI. For example:

- No route is reachable without the session middleware
- Refunds above the threshold keep the approval step
- Outbound webhooks go through the signing helper
- Role changes are admin-only and written to the audit log

Evidence and control for AppSec

Fixed once, stays fixed

A finding from a pentest, or an exploit Konvu reproduced, becomes a Control with the fix commit as its evidence. The same bug cannot come back in a new endpoint.

Yours to edit

Checks start from the mapped Controls. Review, edit, or reject any of them when you want. Add your own. The rules stay yours.

Report-only or blocking

Start by observing. Turn on blocking, per repository, for the rules where a violation must not merge. Branch protection stays yours.

Every decision recorded

Which rule fired, what it found, who allowed or denied, and what changed. SOC 2 Type II certified.

● WHERE THE CODE IS WRITTEN

Claude Code Codex Cursor GitHub Copilot
konvu CLI MCP other coding agents

● BEFORE MERGE

GitHub pull requests GitLab merge requests
Any CI via konvu CLI Report-only Blocking

WHO IT IS FOR

- **AppSec and Product Security leaders** who own the guidelines and cannot review every change
- **CISOs** who need the guidelines stated, enforced, and recorded
- **Anyone prompting a coding agent**, engineer or PM, who gets the rule at the line they are changing without having read the wiki

BUILT FOR

Any application with rules that are not in a generic checklist. Tenant boundaries, roles, permissions, entitlements, approval flows, money movement, the internal helpers every class of code must go through. Teams that build with coding agents, check changes in CI, or both.

GET STARTED

Start with one repository.

Connect a repository and Guardrails maps its Controls with their evidence. Add your own, and every change is checked from then on. Free trial. SOC 2 Type II. Code is never stored.

Start free trial

Book a walkthrough · sales@konvu.com