TruVerifAI

PANEL REVIEW · SECURITY & DATA FLOW

Your code stays in your tenancy

For an enterprise security / AppSec reviewer. Every claim is verifiable against the public client source, and this document is the contract.

> **In one line:** your code stays in your tenancy. Only the specific **diff you submit for review** leaves your machine; the local gates send **hashes and labels, never your source**; and you can turn even the diff-storage off, or run the whole review on **your own models in your own cloud** so it never reaches us at all.

And you don't have to take our word for it. Everything this command installs is open source: MIT-licensed, zero runtime dependencies. Every claim in this document is verifiable against the public client source at github.com/TruVerifAI/init, and this document is the contract: if something here is wrong, tell us and we fix it.

Part 1: How the product handles your code and data

This is the security posture that lets you deploy Panel Review anywhere: on your developers' machines, in your CI, in your own cloud. It is true whether or not you use the evaluation sandbox in Part 2.

What leaves the machine, exactly

Event	What is sent to us	What is NOT sent
Gate check (on a risky write/commit)	a **hashed** repo fingerprint (SHA-256 of the git remote URL or repo path), **content hashes** of the changed hunks, and the local classifier's **category labels + scores**	your source code; your file paths (only coarse class tags like "test/docs"); the raw repo URL/path
Panel review (`audit_coding` etc.)	only the **diff / context your agent explicitly submits** for that review. That is the product: the change you chose to have reviewed	anything you didn't submit
Update check	the client's own version string	anything else
Diagnostics (`TVAI_PAYLOAD_LOG`)	nothing (local-only, opt-in)	

What is stored

- **Persistence OFF** (one toggle, your choice, settings → MCP → "Decide on data persistence"): review **content is not stored**; only usage/billing **metadata** (counts, timestamps, model consensus, verdicts, hashed fingerprints) is recorded. "Nothing stored" is precise here as *no content*, not *no metadata*.
- **Persistence ON** (the account default): the submitted review **content** is stored (this is what powers your dashboard audit trail) plus usage/billing **metadata**. It stays until you delete it; there is no automatic expiry, and you can clear it or turn persistence off at any time.
- Gate-check payloads (hashes + labels) back the admin dashboard row: repo fingerprint + category labels + hunk count + a pushed flag. **Never a diff, path, command text, or commit SHA.**

The data-control ladder: you choose how much reaches us

1. **Default:** hosted panel (our models), persistence on.
2. **Persistence off:** no review content stored our side.
3. **BYOK (bring your own keys):** the review runs on your own model-provider keys.
4. **BYOM (bring your own models):** the panel runs on **your own models in your own AWS Bedrock / GCP Vertex tenancy**. Review content is processed by the models **you already operate** and **never reaches our model accounts**. For enterprises that already run their own inference, this is the strongest posture and needs no new infrastructure on your side.

All four are self-serve on truverif.ai, keyed to your own login.

The client is open source

The entire client is public and dependency-free: `npm pack @truverifai/init`, extract, and read `bin/tvai.js`, `lib/*.js`, and `vendor/gates/*.py`; that is the whole runtime surface. Published with build provenance attestations. Nothing runs as a service, daemon, or startup item; the gates run only when your agent host invokes its hooks.

Authentication & secrets

- Sign-in is a browser **device-flow**: the CLI never sees a password; it mints a per-user API key you approve in your own browser. The key is written only to `~/.truverifai/config.json` on the machine you run it on.
- The key is **revocable**: `npx @truverifai/init uninstall` revokes it server-side (`POST /v1/keys/self/revoke`) before deleting it locally, so a leaked copy is dead. You can also revoke any key from your API-keys page.

Failure behaviour

> The gates **fail open by design**: if our server is unreachable or anything errors, your commit/write proceeds and a visible notice says the change was not gated. The tool never blocks your work on its own failure, and never blocks it on *our* outage.

Part 2: The proposed test setup (a disposable sandbox)

Why we propose a sandbox

To evaluate a tool like this properly, a security team normally has to install it somewhere and point it at real code, which means provisioning a real machine, real credentials, and a real repository before the review has even started. We remove all of that. Instead of asking you to install anything on your own laptops or run it against your own repositories during the evaluation, we hand you a **template repo you launch as a disposable machine in your own tenancy**. You get to exercise the full product (the gates, the panel, the data controls in Part 1) with **zero footprint on your real environment**, and throw the whole thing away when you are done. The sandbox is an evaluation vehicle, not the deployment model: once you are satisfied, you deploy the product itself using the controls in Part 1.

What the sandbox is

A disposable machine **in your own tenancy** (a GitHub Codespace in your org, or your own AWS EC2 / any Docker host) with Panel Review's client-side gates installed by `npx @truverifai/init`. It ships with throwaway demo code, so no real repository of yours is ever involved.

What it gives you

- **Nothing touches your real environment.** The gates, the key, and the demo code all live on the disposable machine, never on a developer laptop or a production repo.
- **No shared credential.** The sandbox uses **your** freshly minted, self-revocable device-flow key, not a secret baked into the template.
- **Fully disposable.** Deleting the machine removes everything on it, including the local key file. Uninstall is the machine ceasing to exist.
- **The full product.** Every control in Part 1 (persistence off, BYOK, BYOM, the open-source client) is available in the sandbox exactly as it is in a real deployment, keyed to the login you use inside it.

What remains after you delete the sandbox

- The disposable machine and everything on it (including the local key file): **gone**.
- Server-side: the API key (revoke it in one command, or from the API-keys page) and, if persistence was on, the stored review content + usage metadata until you clear it. With persistence off, only usage metadata.