Read-only as a design principle

Guide · Watchdog **Version 1.0 - 2026-08-04** Author: Canine Development

Who this is for: security reviewers, technical buyers and engineering leaders assessing how Watchdog handles source code. It sets out what it accesses, where the code is processed, how long each class of data is kept, what leaves the analysis environment, what is retained afterwards, and where the scope ends.

Most tools that read your source code hold on to it: an index, a warm copy, a searchable dashboard. Watchdog reads a repository to produce a signed result, and then disposes of the working copy. What you keep afterwards is a document, not a seat in a system that holds your code.

---

1. What Watchdog receives

The Watchdog GitHub App requests **read-only Contents and Metadata**. It takes no actions and receives no webhook payloads containing code. Sign-in goes through a self-hosted identity provider brokering GitHub, from which Watchdog receives your login, display name and email address.

There is exactly one optional write, and it is off until you enable it. The **audit-marker issue** maintains a single GitHub issue tracking open findings, and requires the App to hold `issues:write` on that repository. It creates, updates or closes that one issue and touches nothing else. If you want no write path at all, do not grant it; the rest of the product is unaffected.

Public GitLab projects can be added by URL, cloned anonymously with no token and no installation. For code hosted elsewhere, a ZIP upload per scan is available.

What no permission and no path allows: Watchdog does not modify your source. There are no commits, no branches, no pushes, no pull requests and no file edits.

---

2. Read-only is an architecture, not a setting

The scan pipeline runs one direction - clone, analyse, sign, publish - and what leaves it is an artifact.

The integration surface follows the same rule. Watchdog exposes an MCP endpoint so that an AI agent, or a person, can ask what the last scan found and what to fix next. Its thirteen tools cover reading findings and reports, claiming and releasing work, provisional resolution, dispute, advisory flags, detector-gap reports and re-scan requests. **No tool writes to your repository, and no tool lets an agent improve its own score.** A

resolution recorded through MCP is provisional and is arbitrated by the next scan; a dispute goes to human triage; agent keys are capability-scoped; re-scan is unavailable unless the repository owner enables it. Your agent reads the target, your agent does the fixing, and the next scan measures whether it worked.

Watchdog measures. It does not fix, and it never fails a build or gates a pull-request merge - there is no CI integration through which it could. It needs nothing installed in your repository or your pipeline to run, and where you do run a linter, a SAST tool or a CI gate, it runs alongside them.

One place where the product does hold something back is worth stating plainly, because "never a gate" is otherwise easy to over-read: in the compliance workflow, a control that an automated check caught failing is pre-set to Fail, and passing it requires a written justification that is reproduced in the artifact. That is a deliberate integrity control on a document you sign, not a gate on your development workflow.

---

3. Where the code is processed, and how long each thing is kept

Every run works from a copy that exists for that scan. Two questions decide whether that sentence means anything: where it lives, and when it goes.

Where. The scan checkout and the engine's scratch space are RAM-backed `tmpfs` mounts, asserted on every deploy: a host whose mounts, cleanup timer or encrypted volume have drifted fails the release rather than serving from a state the data-processing agreement does not describe. Your source tree is never written to disk. Analysis runs in a container with Linux capabilities dropped and no new privileges, on a deny-by-default egress network. Cloning runs in the scan worker outside that container, writing into the RAM-backed path.

How long. By data class:

What	Where it lives	How long
Cloned repository source	RAM only	Deleted when the scan finishes - completed, failed or cancelled alike. A recurring sweep removes anything a crashed run leaves behind, and nothing survives a restart of the host.
Uploaded ZIP source	RAM only	Not deleted at the end of the scan, because unlike a repository it cannot be fetched again - dropping it would mean asking you to upload a second time. Removed after seven days without use, and gone on restart either way.
Per-scan working data	RAM only	Deleted when the scan finishes, including on failure.
Reports, findings, scores, trend points, run metadata	Encrypted volume (LUKS2, AES-256-XTS)	Retained while the repository is connected.
Account data	Database	You can close the account yourself. There is a 14-day reversible period, after which it is deleted. Invoices are kept where bookkeeping law requires.

Read the fourth row carefully, because it is where "we do not keep your code" is most easily over-read. Reports quote **short code excerpts**: the file path, the line number and the fragment a finding cites. That is derived material about your source and it persists. What does not persist is the tree.

The clone token is minted fresh and short-lived for each scan from the connection you authorised, is used only for the clone or fetch, and is stripped from the checkout's stored remote so it does not sit on any medium afterwards. Standing access comes from the App installation and you can revoke it at any time.

If your posture requires that source never leave your own boundary, a self-hosted deployment does the clone and the analysis on your infrastructure and sends only results outward. Its ingestion path is token-authenticated per repository.

---

4. The language model runs on our own hardware

Some findings carry an advisory read from a language model, used for prioritisation and explanation.

In the managed service, that model is **self-hosted on hardware Canine Development owns and operates**. No source code is sent to OpenAI, Anthropic, Google or any other AI provider, and your code is not used to train any model, ours or anyone else's. This is structural rather than a setting: the analysis container's only route out is an allowlisting proxy that permits package registries and scanner databases, plus a host-side endpoint whose upstream is the local model. No AI-provider domain is reachable from the environment your code is analysed in.

A customer running Watchdog on their own infrastructure configures their own inference endpoint, and the choice is theirs.

Two consequences. On residency: all processing and storage happens on servers owned and operated by Canine Development in Denmark, so your data stays in the EU. Note that your existing code host remains your own relationship and is not relocated by this - it is a supplier you chose, governed by your agreement with it. On the score: because the model is advisory, its output is marked as advisory in the artifact and can change what a report says but never the computed number. The measurement is reproducible independently of any model.

---

5. Who can see what is kept

Report access mirrors your access at the code host, re-synchronised daily: access lost there is access lost here within a day. Public reports exist only for public repositories that are published, and they pass through a sanitiser that strips account names and security-sensitive specifics. Production access inside Canine Development is restricted to named operations personnel and is logged. There are no contractors and no third parties in that path.

The whole sub-processor list is one entry: a payment processor, which activates only when billing is taken into use. If you do not use billing there are no sub-processors at all.

---

6. The deliverable is a signed package you hold

Every scan produces a **CAI delivery package**: a content-addressed manifest with an Ed25519 signature, pinned to the exact commit and the rubric version, with the supporting evidence embedded. The reports travel inside it - Executive, Engineering, Security and Audit views as HTML and PDF, alongside CSV, SARIF and CycloneDX SBOM outputs.

Two checks confirm one of these, and they are separate:

1. **The signature.** Ed25519-verify the package against the issuer's public key. A valid signature proves the package was produced by the issuer rather than assembled by whoever shared it.
2. **The content hash.** Recompute the hash over the manifest and compare. One edited byte breaks it.

Both are documented at cai.canine.dev/verify, and real signed packages are browsable at watchdog.canine.dev/publicreports. Neither check requires an account with us. Separately, the score itself is reproducible: run the open reference scorer over the published evidence, pinned to the same rubric version, and you should get the same number.

Reproducibility is a property of the record, not a promise about it. Every report carries the audit trail behind its deep-scan dimensions - each external tool, the exact command, the number of findings it returned, and the retained raw output:

Appendix B — Reproduction & audit trail

Every external tool invocation behind a deep-scan dimension — the tool, its captured version, the exact command, how many findings it yielded, and a link to the retained raw output. **To reproduce any finding:** check out the same commit and run the command shown (repo-relative — never an absolute scratch path). The complete raw scanner output is retained verbatim under `artifacts/raw/` (indexed in `artifacts/raw/index.json`); per-invocation exit codes and wall-clock durations are in `sidecar.json` — kept out of this table so the rendered report stays byte-identical across runs of the same commit.

DIMENSION	TOOL	VERSION	COMMAND	FINDINGS	RAW OUTPUT
D28 · Secrets (history)	gitleaks	–	`gitleaks detect --no-banner --report-format json --report-path /dev/stdout --exit-code 0 --source .`	2	artifacts/raw/gitleaks-history.json
D29 · Static Analysis (SAST)	semgrep	–	`semgrep --config /opt/semgrep-rules/security-audit.yml --config /opt/semgrep-rules/owasp-top-ten.yml --json --quiet --timeout 0 --metrics off .`	0	artifacts/raw/semgrep.json
D30 · Dependency Vulnerabilities	none (no readable dependency manifest)	–	`none (no readable dependency manifest): not present in this environment`	0	—
D31 · IaC & Container Security	trivy	–	`trivy: not applicable — No Infrastructure-as-Code or container manifests found (Dockerfile, Terraform, Kubernetes/Helm, CloudFormation); nothing to scan.`	0	—

The audit trail from the published survey of psf/requests. Every command is repo-relative, so checking out the same commit and running the line shown reproduces the finding. A dimension with nothing to analyse records that, and why, in the same table.

This is what makes the read-only posture worth something. You are not renting continued access to a system that holds your code; you are holding a document that stands up on its own.

---

7. What Watchdog does not do

- **Source-side analysis only.** It reads source, history, configuration and manifests. It does not run your application, probe a live endpoint, perform dynamic scanning or conduct a penetration test.
- **Evidence, not certification.** It produces signed, reproducible evidence about the automatable part of code quality and security posture. It does not make you compliant and does not certify you against GDPR, SOC 2 or ISO 27001. The division of labour is that the engine measures what can be measured, you declare the rest, and a named human signs. Canine Development holds no SOC 2 attestation today.
- **The codebase is the unit of measurement.** There is no individual-developer score and no velocity or DORA-style ranking. Where dimensions read git history they identify where knowledge is concentrated, and no per-person score is computed or stored.

Coverage: as published on 4 August 2026, the production catalogue held **13 languages**, 12 analysed deeply and JavaScript analysed structurally. The catalogue is a live production record and may change, so check the current version at watchdog.canine.dev/api/public/language-support. The rubric holds 127 catalogue entries across ten lenses - five always on, and five that light up when the codebase warrants them - published at cai.canine.dev/spec.

8. The short answers

Where does our source code go, and how long is it kept? Into RAM on servers in Denmark, for the length of one scan. A cloned repository is deleted when the scan finishes; an uploaded ZIP is kept until seven days without use, because it cannot be re-fetched. Neither survives a restart. What persists is the report package and the short code excerpts its findings cite, on an encrypted volume, for as long as the repository is connected.

Do you send our code to a third-party AI? No. In the managed service the model is self-hosted, and the analysis environment's egress allowlist does not include any AI provider. A customer running their own deployment configures their own endpoint.

Do you train on our code? No - neither our own models nor anyone else's.

Can Watchdog change our code or our repo? Not your code. There are no commits, branches, pushes, pull requests or file edits, and no MCP tool writes to a repository. One optional feature, off until you enable it, maintains a single GitHub issue as an audit marker and needs `issues:write` to do it.

Is this a penetration test? No. It is source-side analysis. Dynamic and outside-in testing are a separate discipline.

Does this make us compliant or certified? No. It produces evidence. You declare the rest, and a human signs.

Learn more and verify

- **Data handling, hosting and residency:** watchdog.canine.dev/security.
- **The contractual version of §3 and §5**, on the Danish Data Protection Agency's Article 28(8) standard clauses: the DPA, whose annexes carry the retention table, the sub-processor list and the security measures.
- **The signature scheme and both checks:** cai.canine.dev/verify.
- **How the score is computed and what supported means per language:** the methodology page.
- **Where the compliance failure-gate applies:** watchdog.canine.dev/compliance.
- **Real signed packages you can open and check:** watchdog.canine.dev/publicreports.