



One maintainer. One open-source repo. Aligned with **23** security & privacy standards.

We did *not* write 23 checklists.

How good architecture — not a pile of paperwork — keeps a small open-source project audit-ready across the world's privacy and security rules, with no compliance team and no audit budget.

↳ each technical claim names the file that backs it • a CI drift gate keeps the tool docs honest against the code

The standards wall

23 frameworks · 6 domains · across the US, EU, UK & global bodies.

Domain	Where it applies	Frameworks aligned to
Information security	Global · US · UK	ISO 27001 · SOC 2 Type II · NIST CSF 2.0 · FedRAMP · BSIMM · UK Cyber Essentials · NCSC CAF
Privacy & data rights	EU · UK · US · APAC	GDPR / UK GDPR · Global CBPR
Healthcare data	US	HIPAA · HITRUST CSF
AI & agentic systems	EU · US · Global	EU AI Act · NIST AI RMF · OWASP LLM Top 10 · OWASP Agentic Top 10 (2026 draft)
AI-tool (MCP) security	Global · US	OWASP MCP · CSA MCP · NSA MCP Guidance
Supply chain & protocol	EU · Global	EU CRA · NIS2 · FIRST PSIRT · MITRE ATT&CK · RFC 9700/9449

No compliance team. No budget for 23 audits. So how?



The trick: 23 standards → 8 properties

Strip the labels, and they demand **the same handful of things**:

Access control

Encryption rest + transit

Audit trail

Data minimization

Tenant isolation

Supply-chain integrity

Vulnerability handling

Transparency

Compliance teams call this *control mapping* — and it's usually a 1,400-row spreadsheet and a consultant. **We built the 8 properties into the code instead.**

The payoff: add the 23rd standard and you write zero new code — you already satisfy it. Compliance cost stops scaling with the number of regimes.



The architecture that satisfies them

The 8 properties are baked into *how the code is built* — so they hold everywhere, not just where someone remembered to add them:

- **One way in for every dependency** — nothing reaches in through a back door, so there's one place to audit
- **Swappable parts behind clean seams** — the cache, search, and audit layers can change without rewriting callers
- **One encryption routine** — a single AES-256-GCM helper for all at-rest encryption (opt-in via a key), never re-implemented per feature
- **One safe web-fetch client** — *every* outbound request goes through the same SSRF-checked path
- **An audit log on every tool call** — structured, secrets stripped out, and it never slows the request

"Compliance through architecture, not bolt-on checklists."



The safest config is the *default* config

Run it the normal way — locally, inside your AI app — and there's **no network port open, no login to misconfigure, nothing exposed to the internet**. The app that launched it is the only thing that can reach it. The typical user *can't* hold it wrong, because there's nothing to set.

The heavyweight controls for shared, multi-user deployments — sign-in, per-customer isolation, rate limits, encryption, audit logs — only switch on when an operator deliberately runs it as a network server.

Secure by default, permissive by configuration. Power is unlocked by an *explicit* choice, never the reverse.



Controls, not certificates: the honest line

"Compliance as architecture" ships the *controls*. It can't ship the *operating organization* a certificate requires — so here's the split:

Layer	Who owns it
The project ships the controls	SSRF-safe fetch, AES-256-GCM at rest + TLS, secrets-masked audit logs, tenant isolation, OAuth 2.1, consent + erasure <i>primitives</i> , SBOM + signed releases + a PSIRT process
The operator owns the process	They're the data controller — sign BAAs/DPAs, set the retention <i>schedule</i> (code gives TTL knobs, not policy), run access reviews + incident response, choose lawful basis, run DPIAs
A hosted SaaS adds the program	Trained staff, controls <i>audited over time</i> , 24/7 IR, signed customer DPAs, and the actual SOC 2 / HITRUST / ISO 27001 audit — none of which a repo can contain

So "aligned with 23 standards" means *we provide the technical controls each one requires* — not that an organization has been audited against them. A binary can't clear a hospital's HIPAA bar; it hands that review its evidence.



Agency sharpens one old threat — and adds a new one

Give an AI a tool that fetches any page, and it now *chooses the URLs*. That **amplifies an old vuln** and **surfaces a new one**:

Old vuln, now automated — SSRF (OWASP Web A10). A hijacked link can steer an *autonomous* fetch at your internal network. So before any fetch the server rejects private/reserved IPs and cloud-metadata hosts, connects only to the exact resolved address (DNS-rebind defense), and re-checks on every redirect.

A genuinely new class — indirect prompt injection. A booby-trapped page can try to hijack the AI reading it. The server strips active markup, caps size, and stamps every result that carries external text with an **untrusted-external-content** marker — in the JSON *envelope*, never inside the content where a page could forge it, and **enforced by a cross-tool drift test** so a new tool can't ship unmarked. It does **not** enforce the prompt boundary — that's the *host's* job, where the model and agent loop live.

Prompt injection is #1 on OWASP's LLM list, and the agentic rules are still being drafted. This tool sits squarely in that gap.

↳ proof: internal/scrapper/ssrf.go · internal/tools/scrape.go (envelope "trust" marker) · internal/tools/metadata_test.go (cross-tool gate) · internal/content/sanitize.go · OWASP Web A10 · LLM01



An erasure registry you can't outrun

Privacy starts with collecting little: the cache is content-addressed (not keyed to a user), and the operator — not us — is the data controller. But the moment a feature *does* store personal data, "right to be forgotten" has to actually work.

So it's enforced structurally: **every store that holds personal data must register an Exporter + Eraser**. One `(tenant, user)` request fans out to all of them — and each store ships a round-trip **release-gate test**.

Add a new feature and forget to wire its eraser? CI fails — not an auditor. GDPR access / portability / erasure (Art. 15 / 17 / 20) becomes a property of the build, not a promise in a policy PDF.

shipped in v1.13



Consent as a primitive: record → verify → honor

The AI-tool standard (MCP) says *asking* the user for consent is the **client app's** job — so most servers do nothing. But whoever *stores* the data is, in law, the **data controller**, and GDPR / Quebec Law 25 make *them* prove consent was given and honor a withdrawal — a duty a login token can't discharge.

So the server treats consent as three things it actually does:

- **Records** it — encrypted, logged, with a typed purpose ("memory," "analytics," "workspace").
- **Verifies** it on every access — no record, no processing (it defaults to *off*).
- **Honors** it — a withdrawal automatically erases the data it covered.

shipped in v1.13



The docs are tested, not trusted

"Keep the docs accurate" isn't a good intention here — a stack of small mechanisms makes drift hard to write and impossible to merge quietly:

Layer	What keeps it honest
Mechanical facts are machine-checked	Tool lists, output shapes, read/write flags are never hardcoded — they point at the file that defines them
Rules the AI writes by	<code>CLAUDE.md</code> makes "every claim links to its file" a mechanical rule for Claude / Copilot / Cursor
A test reads the docs	At build time it parses <code>docs/TOOLS.md</code> , starts a real server, and fails if the documented tools or shapes don't match reality
Gates that can't be skipped	The doc-drift check runs in CI on every PR — even docs-only ones

The *judgment* — threat models, standards crosswalks — lives in prose and gets human review. Where a claim is enumerable, the build enforces it.



So what is this actually worth?

The same architecture pays off differently for everyone who touches it:

If you're...	What you get
A user	The safe setup is the <i>default</i> — nothing to configure, nothing to leak
An operator / buyer	One small binary you can take to a hospital, an EU regulator, or a federal review — evidence links to code, not slideware
A developer / contributor	Add a feature and the guardrails (consent, erasure, audit, tests) come <i>with</i> it — you can't ship the unsafe version
A founder / eng leader	Compliance cost scales with <i>features you turn on</i> , not with the number of regulations or headcount

Compliance stops being a project you fund and becomes a property you inherit.



What transfers to any project

1. **Map standards to primitives, not checklists** — satisfy many at once.
2. **Make the default the safe one** — gate power behind explicit config.
3. **Let compliance scale with features** (tiers) — not a big-bang program.
4. **Encode the rules as constraints, enforce them in CI** — including for AI coding agents.
5. **If a doc can be wrong without a test failing, it will be** — so test it.

Honest boundaries: "aligned with," not "certified." The local and server threat models differ. By default it stores little; the features that do store personal data are opt-in, consent-gated, and erasable — not absent.



Read the code, not the marketing.

Each technical claim here names the file that backs it — open any one and check.
And a CI drift gate keeps the tool docs honest against the code.

The receipts: **v1.10** tenant isolation → **v1.11** zero-config fallback →
v1.12 HTTP hardening → **v1.13** consent + GDPR erasure →
v1.14 OWASP Agentic Top 10 hardening (audit-driven).

*Solo maintainer · MIT licensed · **contributors welcome**. Spot a claim that doesn't match the code? Open an issue or a PR — that's the whole point. Come help build it: github.com/zoharbabin/web-researcher-mcp*