Zain Dana Harper

Curriculum Vitae | Systems and Verification Engineer

Seattle, Washington | Open to remote, hybrid, onsite, contract, part-time, and project-based work
zaindharper@gmail.com | https://harperz9.github.io | https://github.com/HarperZ9 ORCID: https://orcid.org/0009-0001-7175-5393

PROFILE

Seattle-based systems engineer whose current work is epistemic verification infrastructure: engines that make AI-agent work provable, compilers that make capability use part of the type system, and evidence surfaces a stranger can re-check offline. Public work also spans real-time graphics and color science, technical documentation, and operations.

The trajectory is unusual and documented: eleven years operating tree crews from the ground (rigging systems, clearance and swing judgment, second set of eyes for the person in the air), self-taught into software, then an independent engineering practice from 2023. The throughline in his own words is "finding integration through abstraction": enter ambiguity, map the moving parts, build the working surface, preserve evidence, and leave an artifact another person can inspect or use. Overall programming experience includes roughly thirteen years of hobby, self-directed, and exploratory development; the serious public and release-driven engineering arc begins in 2023.

ENGINEERING DOMAINS

Verification and accountability infrastructure. Capability gates, sealed and hash-chained action receipts, offline re-verification, witness graphs, provenance, replayable ledgers, evidence intake, preregistered protocols, and public proof surfaces.

AI-agent infrastructure and evaluation. Multi-provider and local-endpoint routing, tool-use loops, MCP tools, task decomposition, multi-agent orchestration, worker/verifier separation, and evaluation workflows.

Compilers and language tooling. Lexing, parsing, AST and intermediate representations, type checking, typed capability effects, ownership and lifetime analysis, code generation, native C output, shader output, and editor tooling.

Graphics, color, and native systems. D3D11/HLSL rendering, proxy-DLL architecture, post-processing, shared-memory IPC, color spaces and appearance models, HDR and tone mapping, ICC/VCGT and 3D-LUT workflows, and display characterization.

Documentation, compliance, and operations. API and developer documentation, implementation guides, proposals, release notes, operational playbooks, structured authoring, site assessment, scheduling, safety judgment, and vendor coordination.

THE 2026 EPISTEMIC VERIFICATION ENGINE

Flywheel | accountability engine and native client | Python (stdlib-only engine), Flutter

Shipped on the public default branch as of 2026-08-03:

- A verified-inference loop in which a model acts only through a default-deny capability gate it cannot talk past. Writes, exec, and external tools are off until a human grants them; the model cannot self-authorize.

- A receipt discipline: every tool invocation emits a sealed record binding the capability class, the admission decision, and witnessed argument and output digests (never raw content). Receipts are hash-chained, signed, and offline-verifiable; verification recomputes every seal from the record alone.
- A transitive witness graph in which a drifted action degrades exactly its downstream dependents, plus an append-only organizational learning loop that turns witnessed divergences into hash-bound, re-checkable lessons.
- An infrastructure control package: network egress receipts, credential scanning, isolation acceptance tests, a dual-confirmation kill switch (off by default; cloud credential revocation is a stub pending live IAM wiring), and cross-layer event correlation (heuristic today; statistical kernels exist but are not yet bound).
- A published, dated assessment (2026-08-01) mapping the engine against the failure classes of the July 2026 agentic-security incidents, with coverage and gaps stated per class.

Also merged as of 2026-08-03:

- Capability-typed shell admission with a Unicode-spoof neutralizer.
- A domain oracle registry that routes any claim to its verifier, with three live domains (code via pytest, mathematics via the Lean kernel, and an ml measurement gate that bounds a tested effect against a negative control) and an honest UNVERIFIABLE verdict for domains with no verifier yet.
- A model-neutral router with quota failover across providers, a subscription-auth adapter that consumes an authorized token rather than minting one, and session tooling to browse, resume, and export past verified runs.

The trained local model is the replaceable half of the loop; the engine is the durable half. No capability uplift is claimed for the model. The engine's first preregistered confirmatory run completed 2026-08-04: 2,646 certificate bodies submitted through the accept path in nine model contexts produced zero verdict disagreements, so acceptance is provably a function of the certificate and never of the model; oracle selection beat random selection at every model size on the solvable task family, scored by an independently written held-out checker with random and placebo controls (paired delta up to +0.32, exact p to 3.8e-06). The harder family defeated every model on the ladder; that null is kept and labeled uninformative, and the self-scored comparison is refused by design. Run and analysis are anchored in a signed, offline-verifiable transparency ledger.

SELECTED PUBLIC SYSTEMS

BuildLang | Rust compiler and systems language | crates.io surface

A systems language in which ambient access is part of a function's type. Typed capability effects (filesystem, network, foreign calls, gated compile-time macros) tracked through function values, closures, struct fields, control flow, and async blocks. Hindley-Milner inference, sum types, an opt-in experimental linear-types attribute with honestly scoped soundness, two-way C FFI, a production C backend, HLSL and GLSL shader output, and experimental SPIR-V, LLVM IR, WebAssembly, Rust, x86-64, and ARM64 backends. The `buildc` CLI bundles build, run, test, repl, fmt, pkg, watch, doctor, and an LSP server. Every checked build can write a receipt that `buildc receipt verify` re-derives later.

Project Telos tool family | versioned public packages

Index (repository maps, dependency graphs, context packs), Gather (provenance-aware research intake), Forum (model-agnostic orchestration with a replayable ledger), Crucible (worker/verifier evaluation with MATCH / DRIFT / UNVERIFIABLE verdicts), Learn (accountable learning), EMET (byte-level witnessing, frozen 1.0.0 specification), and public source packages for routing (Relay), toolchain auto-wiring (Plexus), and

provenance-carrying agent memory (Mneme). Registry publication is not claimed where a release has not been verified.

Real-time graphics work

Public HLSL/C++ graphics work includes a Skyrim post-processing project whose current public career materials report more than 900,000 downloads, plus D3D11/HLSL systems involving tone mapping, TAA, SSR, SSGI, GTAO, volumetrics, GPU traces, and read-only shared-memory bridges.

RESEARCH AND PUBLIC WRITING

- "No Receipt, No Accept" (2026): the flagship essay on evidence discipline for machine-generated work. https://harperz9.github.io/no-receipt-no-accept.html
- "Models Propose, Oracles Dispose" (July 2026): the propose/dispose rule and the receipt-emitting compiler. https://harperz9.github.io (writing index)
- "Pick the Lock for Everyone" (2026): capability distributed by construction. https://harperz9.github.io/pick-the-lock-for-everyone.html
- Agentic security assessment (2026-08-01): Flywheel mapped against the July 2026 incident classes, in the Flywheel repository.
- Preregistered protocols with confirmatory addenda (July 2026) covering size-invariant verification, control replication, isomorphic perturbation, and trainer diagnostics, in the Flywheel repository. The preregistered confirmatory run completed 2026-08-04 with its endpoint met and the analysis anchored in the signed preregistration ledger.

SPEAKING AND COMMUNITY

- "Pick the Lock for Everyone: Building Verifiable AI Workflows in Python" (the "Proof, Not a Portfolio" talk), invited, Puget Sound Programming Python (PuPPy), 25 minutes, scheduled 2026-08-19.
- Coalition for Secure AI (CoSAI): individual contributor to Workstream 4 (Secure Design for Agentic Systems), contributing secure-design patterns and a reference implementation to the workstream as of August 2026.

UPSTREAM OPEN-SOURCE CONTRIBUTIONS

Merged | 22 pull requests across 19 repositories maintained by others

Defect repair, test coverage, and documentation accepted upstream in projects including Datasette, tomlkit, pydantic-ai, DeepEval, pydash, and grimp. Representative merged work: HTTP status handling for malformed composite-key row URLs, a TOML round-trip bug where an array of tables swallowed the next sibling key, a metric crash on unhashable tool outputs, missing module and layer validation errors, and seeded determinism and mid-task recovery fixtures for agent projects.

Open and awaiting maintainer review | 11 pull requests across 11 repositories

Including the Model Context Protocol Python SDK, Drizzle ORM, Datasette, and LLM. Open is not accepted.

PROFESSIONAL EXPERIENCE

Independent Systems Engineer / Founder-Builder | Zentropy Labs (Project Telos, HarperZ9) Seattle / remote | 2023-present

- Build and maintain the public systems portfolio across Python, Rust, C++, HLSL, TypeScript/JavaScript, and web-native surfaces.
- Own architecture, implementation, test strategy, documentation, packaging, release notes, demos, public claims, and maintenance planning.
- Use coordinated coding agents for parallel discovery, implementation, and review while keeping first-party architecture, integration, verification, and release decisions explicit.

Freelance Technical Writer / GRC Documentation / Product Operations Remote | 2017-present

- Produce API and implementation guides, security and compliance documentation, proposals, RFP and grant material, release notes, and operational playbooks.
- Work with NIST 800-171, CMMC readiness, SOC 2, ISO 27001, DFARS, incident response, and audit-support concepts in a technical-writing capacity.
- Keep client names, private deliverables, compliance evidence, and contract terms outside the public portfolio.

Operations and Commercial Arboriculture Lead | Family business Seattle area | 2015-present

- Operated tree crews from the ground for eleven years: ran the rigging systems, judged clearances and how far limbs would swing relative to structures and people, and served as the second set of eyes for the person in the air.
- Led client intake, estimates, site assessment, scheduling, safety judgment, crew and vendor coordination, proposals, and customer communication.
- Carry field-tested operating discipline into software and research systems.

Technical Networking Support, Xbox Division | Microsoft Redmond, Washington | 2013-2014

- Diagnosed TCP/IP, DNS, NAT, firewall, router, and account-adjacent console networking issues across phone and chat support.
- Documented repeatable resolutions and translated technical fixes into clear customer guidance.

TECHNICAL SKILLS

Languages: Python, Rust, C++, TypeScript/JavaScript, Lua, HLSL, C#, PowerShell, Bash. **Verification:** capability gates, sealed receipts, hash chains, offline re-verification, witness graphs, provenance, preregistration discipline, evidence contracts. **Development and delivery:** Git, GitHub, pytest, Node test tooling, GitHub Actions, Linux, CMake, vcpkg, package metadata, CLI design, MCP/tool surfaces, static sites, Canvas/WebGL. **Documentation:** developer and API documentation, implementation and support guides, proposals, release documentation, compliance and audit-support writing.

WORK PREFERENCES

Remote preferred; open to onsite or hybrid work, contract, full-time, part-time, and project-based engagements. Strong fits include AI/agent infrastructure, verification and security tooling, compiler and language tooling, graphics and color, research operations, technical writing, and domain-agnostic systems work.

PUBLIC LINKS

Main site: https://harperz9.github.io Portfolio: https://harperz9.github.io/portfolio.html Research: https://harperz9.github.io/research.html Repositories: https://github.com/HarperZ9 ORCID:

https://orcid.org/0009-0001-7175-5393