Security Tooling — Bloch-SIS-PoW

How the code is scanned, what each tool catches, what it cannot, and how to run everything locally. As of 2026-07-22. Unaudited by a third party — automated scanning + an internal adversarial audit reduce but do not remove risk; consensus logic and cryptographic parameter choices require human + external review.

Scope — two codebases, different threat models

LAYER	WHAT	WHERE
L1 core (Rust)	Post-quantum pure-PoW BlockDAG node: **SHA-256d** proof-of-work (Genesis-2), hybrid **ML-DSA-65 ‖ Falcon-1024** signatures, SHAKE-256 commitments, **GhostDAG** (blue_score) ordering, libp2p networking, the **bloch-euvm** eUTXO contract VM. Consensus-critical infrastructure, not smart contracts.	`crates/`, `src/` (this repo)
EVM / L2	**Not deployed Solidity.** The L2 is a Rust **revm-based scaffold** (design-only, testnet/zero-value): `l2/bloch-l2-{evm,sequencer,prover,anchor,bridge}` in the **products repo** (`bloch-protocol`). Scanned as Rust; the Solidity toolchain is configured for *when* real contracts exist.	`bloch-protocol/l2/*`

Correction to earlier design notes: the live chain is **SHA-256d** (the Module-SIS lattice PoW is the other chain-id in the code, not the live PoW), and **Casper FFG was dropped** — finality is PoW depth + checkpoint reorg-depth protection, deterministic proof-gated validation. Docs here reflect the real code.

L1 Rust scanners

TOOL	CATCHES	CANNOT CATCH	LOCAL RUN
cargo-audit	Known RustSec advisories in `Cargo.lock` (CVEs, DoS, unsound)	Logic bugs, novel vulns	`cargo audit` (see advisory posture below for the tracked `--ignore` set)
cargo-deny	Advisories + **license policy** (permissive only; AGPL allowed for the six first-party crates, copyleft denied for third-party deps) + banned/duplicate crates + untrusted registries	Same as audit for logic	`cargo deny check`
osv-scanner	Same `Cargo.lock`, but the **OSV.dev** DB (RustSec **+ GHSA**) — catches GHSA-only advisories cargo-audit's RustSec-only feed misses	Logic bugs, novel vulns	`osv-scanner --lockfile=Cargo.lock` (install: `go install github.com/google/osv-scanner/cmd/osv-scanner@latest`; needs network for the DB — supplementary, not a blocking CI gate)
cargo-geiger	`unsafe` usage across the dependency tree; flags increases	Whether the unsafe is *correct*	`cargo geiger`

TOOL	CATCHES	CANNOT CATCH	LOCAL RUN
Clippy (hardened)	Panics in consensus paths (`unwrap` / `expect`), arithmetic side-effects, pedantic smells	Semantic/consensus correctness	`cargo clippy -p bloch-crypto -p bloch-euvm -- -W clippy::pedantic -W clippy::arithmetic_side_effects -D clippy::unwrap_used -D clippy::expect_used`
Miri	Undefined behaviour in the consensus + serialization test suites	Anything not exercised by a test	`cargo +nightly miri test` (consensus/serialization crates)
cargo-fuzz / libFuzzer	Panics/UB/DoS on adversarial input to the P2P wire, PoW verifier, DAG ordering, signature parsing	Deep logic bugs, consensus splits	`cargo +nightly fuzz run <target>` (targets below)
proptest	Property invariants (deterministic ordering, blue-score monotonicity, emission conservation)	Invariants nobody wrote	`cargo test` (property tests run in-suite)

Fuzz targets (`fuzz/fuzz_targets/`): `block_parse`, `netmsg_decode`, `handshake_decode`, `pow_decode`, `pow_verify`, `sha256d_pow`, `merkle_path`, `ghostdag_order`, `mempool_ops`, `sig_verify` — priority order: P2P deserialization (primary remote surface) → SHA-256d PoW verify → GhostDAG ordering → signature/attestation parsing. An **OSS-Fuzz** application (`fuzz/oss-fuzz/{project.yaml,build.sh,Dockerfile}`) reuses these so Google runs them continuously for free.

EVM / L2 scanners (configured for future Solidity)

No Solidity is deployed yet (the L2 is a Rust revm scaffold). The Solidity toolchain — **Slither**, **Aderyn**, **Mythril**, **Echidna/Medusa**, **Foundry** (`forge test` / `coverage`), **Halmos**, **Semgrep**, **solhint** — is documented + config-scaffolded for when real contracts land. The L2 Rust crates are scanned with the same L1 Rust suite. See the bridge threat model for the highest-risk component.

Advisory posture — the honest disposition

`cargo-audit` / `cargo-deny` are green with a **documented ignore set** (full rationale in `deny.toml` + `audit.toml`). Two classes:

1. **Unmaintained-notice** (no runtime vuln): the vendored `pqcrypto-*` PQ crates (PQClean archived — frozen under Cargo.lock by design), and SP1/zk host-side toolchain crates (backoff, ansi_term, instant, derivative, lru, …) that never touch the consensus/P2P runtime.

1. ⚠️ **OPEN real vulnerabilities — tracked, not dismissed:**

- **RUSTSEC-2026-0118** — hickory-proto NSEC3 closest-encloser proof: **unbounded loop (DoS)**.
- **RUSTSEC-2026-0119** — hickory-proto **O(n²) name-compression CPU exhaustion (DoS)**.
- **Why not fixed:** the fix is in hickory 0.26.x (DNSSEC code moved to `hickory-net`); **libp2p 0.56.0 pins hickory-proto 0.25.2** via `libp2p-dns` + `libp2p-mdns`. We cannot bump hickory without a libp2p release that repins it.
- **Reachability / mitigation:** the DNS surface is `/dns4/` multiaddr resolution and LAN-only mDNS discovery. The node's canonical peering uses `/ip4/` addresses (no DNS resolution), and DNSSEC validation is not enabled — practical exposure is a self-inflicted malicious resolver or a hostile LAN.
- **Action:** remove both `--ignore` entries the moment a libp2p release pins hickory ≥ 0.26.
- ⚠️ **GHSA-vxx9-2994-q338 — yamux 0.12.1 (CVSS 8.7):** stream-multiplexer DoS. Surfaced by **osv-scanner** (GHSA-only — cargo-audit's RustSec feed does not carry it). Same upstream block as hickory: `libp2p-yamux 0.47.0` (libp2p 0.56.0) pins yamux 0.12.1; the fix (yamux 0.13/0.14) is unreachable until libp2p repins.

Reachability is any connected peer over the yamux muxer — **the most exposed of the open residuals**; noise/`/ip4/` peering does not mitigate it. **Action:** bump the moment libp2p repins yamux; until then this is an accepted-but-tracked P2P DoS risk.

- **GHSA-vj64-rjf3-w3v7 — p3-challenger 0.2.2-succinct (CVSS 8.9)** and **GHSA-3g92-f9ch-qjcm — p3-symmetric (2.9):** plonky3 crates pinned by the SP1 4.2.1 prover stack — **host-side proof generation only, never the node consensus/P2P runtime.** Not bumpable without moving off pinned SP1. Tracked; low practical exposure.

Fixes applied this pass (real bumps, in-semver, staged in `Cargo.lock`):

- `spin` **0.9.8 → 0.9.9** — 0.9.8 was **yanked** (would fail `cargo audit --deny warnings` and `deny yanked="deny"`).
- `rpassword` **7.4.0 → 7.5.4** — clears **GHSA-2p6r-x3vv-xqm2**; `rpassword` is a **direct** node dependency (CLI passphrase entry), so this is a first-party surface, not transitive.

bloch-euvm — internally audited + remediated

The native eUTXO contract VM had a dedicated internal adversarial audit (`crates/bloch-euvm/audit/INTERNAL-AUDIT-2026-07.md`). Verdict: proceed to consensus-wiring engineering, **hard block on activation** until the blockers close. **0 critical** (the crate is inert/feature-off), **2 HIGH** integration blockers + **1 MEDIUM** — **all remediated**, each with a passing regression test:

- **F1** — the block commitment now binds `SparseMerkleTree::root()` over the eUTXO set (was a 36-byte scalar summary).
- **F2** — gas now scales with operand byte length + hard memory/size ceilings (was flat: an 8 MB hash cost the same as 1 byte).
- **F3** — checked arithmetic + `overflow-checks = true` mandated in `[profile.release]` (closes the release-wraps-vs-debug-panics validator split).

330 tests pass (debug **and** release parity). Kirpich (the Ustav charter-audit gate) adds 23 fail-closed KRP rules on top.

What automated tooling CANNOT catch (needs human + third-party review)

- **Consensus logic** — GhostDAG ordering correctness, reorg-depth safety, the height-gated fork mechanics.
- **Cryptographic parameter choices** — the ML-DSA-65/Falcon-1024 hybrid construction, SHAKE-256 domain separation, the PoW difficulty regime.
- **The Rust↔EVM bridge** — the highest-risk component (message replay, cross-side signature verification, finality inheritance). See the bridge threat model.
- **Economic/game-theoretic** properties at low hashrate (the chain is 51%-attackable today).

A third-party audit is required before any consensus activation of the native-contract features. Automated scanning is a floor, not a ceiling.
