bloch-euvm — Internal Security Audit

Crate: `bloch-euvm` (Ustav / PSTRN-1 execution layer) **Scope:** `lib.rs` (VM core), `state.rs` (SHAKE-256 sparse Merkle tree), `minting.rs` (native mint/burn), `modules.rs` (charter→validator compiler), `batcher.rs` (settlement/AMM reference), `harness.rs` (activation + `accept_block_model`) **Audit type:** Internal pre-wiring audit (a third-party audit is to follow) **Date:** 2026-07-22 **Auditor:** Lead auditor, internal **Commit context:** branch `euvm/integrate`, activation gated at `EUVM_ACTIVATION_HEIGHT = u64::MAX` (inert)

REMEDIATION STATUS — ALL FINDINGS CLOSED (2026-07-22). Every finding below (F1–F6) has been **fixed** and pinned by a passing regression test; the two HIGH integration blockers (F1 commitment binding, F2 gas metering) and the cross-cutting `overflow-checks` mandate (F3) are resolved. **330 tests pass in debug *and* release. **The body below is preserved as the original audit record; see** §6 Remediation — closed **for the per-finding disposition. The crate remains isolated / feature-off / not consensus-wired; activation stays gated on the forthcoming** third-party audit** + a coordinated height-gated hard fork.

1. Executive summary

Verdict — PROCEED TO STEP 5 ENGINEERING, HARD BLOCK ON ACTIVATION

`bloch-euvm` is **architecturally sound and, in its live surface, safe today**: the entire module is behind an inert activation sentinel (`EUVM_ACTIVATION_HEIGHT = u64::MAX`, `harness.rs:45`), nothing routes through `accept_block`, and the VM interpreter `run()` is **verifiably panic-clean** on adversarial input. There is **no live consensus or DoS risk in the current tree.**

However, the crate is **not ready to be wired and activated.** Two findings are hard blockers that must be closed as part of Step 5, *before* the activation height is ever lowered on any network:

1. **The state commitment does not bind eUTXO state (D1 / HIGH).** `accept_block_model` commits a 36-byte scalar summary (`n_txs ‖ gas_used ‖ burned ‖ to_miner`), not a root over the resulting UTXO set. Two blocks with entirely different token movements produce byte-identical commitments. This is a self-labeled placeholder, but it is the one thing that makes the layer consensus-meaningful.
2. **Gas is flat and does not meter work or memory (Gas-DoS / HIGH-when-wired).** The gas schedule is operand-length-independent; a 1-byte and an 8-MB hash both cost 60 gas, and `Dup` amplifies ~50 MiB of memory for ~1000 gas. Once wired, per-node memory limits make block acceptance machine-dependent — a liveness/consensus split.

The remaining findings (overflow / build-profile divergence, mint ordering, batcher saturation) are real and must be fixed before the fork, but are lower severity and largely confined to reference/foundation code that is not on the validation path.

A **critical cross-cutting fact** underlies the overflow findings: the workspace `[profile.release]` (`Cargo.toml:209`) sets `opt-level` / `lto` / `codegen-units` only — it does **not** enable `overflow-checks`. So the shipped release build **wraps silently** on overflow while debug/test builds **panic**. A validator set running mixed profiles can therefore diverge (crash vs. wrapped value) on the same input. Any consensus build MUST mandate `overflow-checks = true`.

Severity tally

SEVERITY	COUNT
Critical	0

SEVERITY	COUNT
High	2
Medium	1
Low	2
Info	1

No critical (live-exploitable) issue exists because the module is inert. The two Highs are **integration blockers**, not live exploits.

2. Findings (ranked by severity)

#	SEV	FINDING	LOCATION	INVARIANT VIOLATED	TRIGGER	FIX
F1	**HIGH**	Commitment binds a scalar summary, not eUTXO state	`harness.rs:59–64`, `encode_eu_section :144–152`, `accept_block_model`	State commitment must bind the full post-state (inputs, outputs, datums, values)	Block A moves 100 BLCH, block B moves 999 → byte-identical 36-byte commitment; distinct output datums → same commitment	Commit `SparseMerkleTree::root()` (already in `state.rs`) over the resulting UTXO set; carry gas/fee as bound side-data
F2	**HIGH** (when wired; MEDIUM now)	Flat gas schedule → unmetered CPU & memory	`lib.rs:175–182` (`gas_cost`), `MAX_STACK=1024` `lib.rs:186`	Gas must upper-bound CPU and memory per op	1 B vs 8 MB `Sha256d` / `Shake256` both cost 60 gas; `Dup` yields ~50 MiB resident for ~1000 gas (>40 KB/gas)	Charge gas ∝ operand byte length for `PushBytes` / `Dup` / `Pick` / `Sha256d` / `Shake256` / `Size`; add hard per-operand, per-program, and total-allocated-bytes ceilings
F3	**MEDIUM**	Unchecked arithmetic → build-profile divergence (panic vs. wrap)	`minting.rs:313` (`cap+1`), `batcher.rs:124/128` (`old_k` / `new_k` `i128` mul); `lib.rs:484` theoretical-only	Consensus arithmetic must be deterministic and fail-closed across build profiles	`cap == i128::MAX` overflows; reserves near `u64::MAX` overflow `i128` (`u64::MAX² > i128::MAX`)	`cap.checked_add(1)`, `checked_mul` in `old_k` / `new_k`, fail-closed on `None`; **mandate `overflow-checks = true` for all consensus builds**
F4	**LOW**	`settle()` uses `saturating_add` on reserves	`batcher.rs:247/250`	Reserve accounting must reject, not silently cap, on overflow	`give_amount` pushes a reserve past `u64::MAX` → silently saturates, breaking value conservation	Use `checked_add` → reject the settlement on `None`

#	SEV	FINDING	LOCATION	INVARIANT VIOLATED	TRIGGER	FIX
F5	**LOW**	`fixed_supply_cap_policy(cap)` construction hazard	`minting.rs:313`	Policy constructors must not panic / emit dead policy on in-range input	`cap == i128::MAX` → debug panic; release wraps to `PushInt(i128::MIN)` → fail-closed dead policy	`cap.checked_add(1)` returning an error, or emit `new <= cap` without the `+1`
F6	**INFO**	`validate_tx_with_mint` error/gas outcome depends on `mints` Vec order	`minting.rs:181–217`	Same logical mint-set must yield the same accept/error regardless of encoding order	Reordering the `mints` slice changes *which* rejection error surfaces and where the shared gas budget is exhausted (conservation itself is order-independent)	Require `mints` in canonical `asset_id` order; reject otherwise (mirror the existing `DuplicatePolicy` guard)

Positive result — `run()` is panic-clean (CONFIRMED). Every interpreter path was independently traced: `pop!` → `StackUnderflow`; `Pick` → `checked_sub`; `Add/Sub/Mul` → `checked_*` → `Overflow`; `CtxField/TxOut*` → `.get().ok_or(...)`; `as_asset` → `try_into`; gas → `checked_sub` → `OutOfGas`. No panicking program could be constructed. F3/F5 panics live only in *reference constructors and helpers*, never in the interpreter.

---

3. Per-invariant assessment

Determinism — PASS (with F1 gap). Same input → same output bytes holds throughout; encoding, hashing, and per-asset folds are all deterministic (`BTreeMap` ordering). The single gap is *binding*, not determinism: the committed bytes are stable but do not bind the eUTXO effects (F1).

Panics / memory-safety — PASS (interpreter). `run()` is panic-clean under adversarial input. Residual panics (F3, F5) are in non-wired reference constructors/helpers and only fire under `overflow-checks` (debug/test); the current release profile wraps instead.

Gas — FAIL (must fix before wiring). The schedule (`lib.rs:175–182`) is flat and operand-length-independent, with no program-byte or operand-byte ceiling. `MAX_STACK=1024` bounds stack *depth*, not operand *size*. This is F2 and is the second hard blocker.

Conservation — PASS (strong), one LOW. Per-asset conservation is enforced via order-independent `BTreeMap` folds; mint/burn relaxation is a checked `i128` net-delta with a BLCH-unmintable guard, policy-hash-identity check, and `burn >= 0` floor. The only conservation defect is F4 (`saturating_add` in the reference batcher).

State proofs — PASS (not yet load-bearing). The SHAKE-256 sparse Merkle tree (fixed depth 256, domain-tagged KEY/LEAF/NODE, empty-subtree ladder) provides working membership and non-membership proofs with `verify()`. Sound in isolation — but not yet the thing the block commits to (see F1). Wiring `root()` into the commitment is the fix that makes both F1 and this component load-bearing.

Modules — PASS, one construction hazard. The charter→validator compiler emits deterministic `Vec<Op>` per `ModuleKind`. The one issue is the `cap+1` construction hazard (F5) in the reference supply policy.

Batcher — reference only, two issues. The settlement/AMM helpers are explicitly reference/not-wired. `old_k` / `new_k` overflow on large reserves (F3) and `settle()` saturates (F4). The production settlement path (`build_settlement_tx` → `validate_tx`) computes `k` via the VM's `checked_mul`; the helper panic is reachable only by a consumer calling `old_k` / `new_k` directly.

Activation — PASS (good discipline). `EUVM_ACTIVATION_HEIGHT = u64::MAX` is an inert sentinel; `is_feature_active(height)` is a deterministic `height >= H` comparison every node computes identically; below activation the committed bytes are byte-for-byte the legacy path. Gating discipline is correct — this is what keeps the crate safe today.

---

4. Test baseline & audit-added coverage

- **Baseline:** 147 pre-audit tests passing under `cargo test -p bloch-euvm`.
- **Audit added: 39 targeted adversarial repro tests** across 10 `tests/audit_*.rs` harness files (plus in-source pinning tests for the construction hazards), all passing:

HARNESS	TESTS	COVERS
`audit_determinism.rs`	3	same-input→same-bytes
`audit_determinism_commitment.rs`	4	F1 — commitment does not bind eUTXO state
`audit_gas.rs`	5	F2 — unmetered work/memory
`audit_panics.rs`	5	`run()` panic-clean + F3/F5 overflow sites
`audit_batcher.rs`	4	F3 `old_k` / `new_k`, F4 `saturating_add`
`audit_conservation.rs`	7	per-asset conservation, mint net-delta
`audit_stateproof.rs`	4	SMT membership / non-membership
`audit_modules.rs`	1	charter compiler determinism
`audit_modules_supply.rs`	2	F5 supply-cap policy
`audit_activation.rs`	4	inert sentinel + exact-height flip

Every finding in §2 has at least one passing repro test that demonstrates the exact claim.

---

5. Prioritized remediation — gating the hard fork

1. **Fix the two HIGH blockers (F1, F2).**

- F1: commit `SparseMerkleTree::root()` over the resulting UTXO set in `accept_block_model`; keep gas/fee as bound side-data.
- F2: length-proportional gas for `PushBytes` / `Dup` / `Pick` / `Sha256d` / `Shake256` / `Size`, plus hard per-operand, per-program, and total-allocated-bytes ceilings enforced before any wiring.

1. **Fix the MEDIUM/LOW arithmetic & ordering issues (F3, F4, F5, F6)** and **mandate** `overflow-checks = true` in every consensus build profile (the single most important cross-cutting fix).
2. **Re-audit internally** against the fixed tree; extend the repro suite to prove F1/F2 are closed (commitment now diverges on differing eUTXO state; gas now scales with operand length; ceilings reject).
3. **Commission the third-party audit** on the wired-but-inert crate — with F1/F2 closed and `overflow-checks` on, so the external auditor reviews the intended consensus behavior, not placeholders.
4. **Only then set the activation height** — on testnet first, with the fork gated behind a height that gives node operators time to upgrade, and never before the third-party audit signs off.

Do not lower `EUVM_ACTIVATION_HEIGHT` until steps 1–4 are complete.

---

6. Remediation — closed (2026-07-22)

All six findings are fixed and each is pinned by a passing regression test. Steps 1–3 of §5 are complete; steps 4–5 (third-party audit, then activation-height setting) remain the standing gate before any consensus wiring goes live.

#	SEV	FIX LANDED	VERIFYING TEST
F1	HIGH	`accept_block_model` now commits `SparseMerkleTree::root()` over the resulting eUTXO set; gas/fee carried as bound side-data. Differing token movements now produce **diverging** commitments.	`audit_determinism_commitment.rs` — asserts distinct eUTXO effects → distinct roots
F2	HIGH	Gas is now proportional to operand byte length (`GAS_PER_BYTE` / `HASH_GAS_PER_BYTE`) for `PushBytes` / `Dup` / `Pick` / `Sha256d` / `Shake256` / `Size`, with hard ceilings enforced before execution: per-operand 1 MiB, per-program 1 MiB, total-allocated 64 MiB, plus tx-resource limits — all **fail-closed**.	`audit_gas.rs` — length-scaled cost + ceiling rejections
F3	MED+X	`checked_add` / `checked_mul` at `minting.rs` (`cap+1`) and `batcher.rs` (`old_k` / `new_k`); **`overflow-checks = true` added to `[profile.release]`** — release and debug now panic-parity, closing the mixed-profile validator split.	`audit_panics.rs` , `audit_batcher.rs`
F4	LOW	`settle()` reserve math switched `saturating_add` → `checked_add` , rejecting the settlement on overflow instead of silently capping.	`audit_batcher.rs`
F5	LOW	`fixed_supply_cap_policy(cap)` uses `checked_add(1)` , returning an error on `cap == i128::MAX` instead of panicking / emitting a dead policy.	`audit_modules_supply.rs`
F6	INFO	`validate_tx_with_mint` requires `mints` in canonical `asset_id` order and rejects otherwise (mirrors the `DuplicatePolicy` guard) — accept/error outcome is now encoding-order-independent.	`audit_conservation.rs`

Post-remediation baseline: the suite grew from 147 → **330** tests, all passing under `cargo test -p bloch-euvm` in **both** debug and `--release` . On top of the VM, the **Kirpich** charter-audit gate (`kirpich.rs` + 4 lanes, 23 fail-closed KRP rules) refuses to compile any Ustav token charter carrying a blocking defect. The crate is still inert (`EUVM_ACTIVATION_HEIGHT = u64::MAX`); nothing routes through `accept_block` .

Internal audit — for engineering and pre-fork planning. Findings F1–F6 remediated 2026-07-22 (§6). Not a substitute for the forthcoming third-party audit, which remains mandatory before activation.
