Aeva Security

Threat model, protocol controls, operational safeguards, assurance and disclosure

AevaChain LLC · aevachain.com · Version 0.1, September 2026 · applies to testnet-0

Scope. This document describes how the Aeva network protects the integrity of the ownership records it holds, the finality of the settlements it executes, and the availability of the network itself. It states what is enforced by consensus, what is enforced by operations, what is verified by independent reconciliation, and what is not guaranteed at all. It is written for issuers, custodians, validators and their risk and compliance functions. Every control marked *implemented* is present in the released software and covered by the tests named in Section 9; every control marked *planned* is a stated intention with no date. Where the current network falls short of the target posture, this document says so.

1. Security posture at a glance

Property	Mechanism	Status
Ownership records cannot be inconsistent	Conservation and lock invariants executed by validators at every block; halt on violation	implemented, tested
Settlement is final and atomic	BFT consensus with single-block finality; two-phase DvP committed in one state transition	implemented, tested
Transfer restrictions cannot be bypassed	Closed policy grammar evaluated on the post-state of every transition, including EVM calls	implemented, tested
No personal data on chain	Attestations carry only subject, type, attester, expiry, revocation and a hash	implemented, linted
Contract code cannot mint or move rights	EVM is a facade; precompiles call the native path; no issuer selectors exposed	implemented, tested
Bridge cannot mint stake	Native AEVA is canonical; routes lock and release, never mint; per-route caps and pause	implemented (RH-2), tested locally
Independent verification	Event-sourced indexer replays from genesis and reconciles against the chain every block	implemented, tested
Validator key isolation	Validators unreachable from the public network; sentries take all traffic	implemented in deployment
Hardware-backed consensus signing	Remote signer (tmkms / Horcrux)	planned; required before mainnet
External code audit	Independent review of the native modules and the EVM integration	planned; required before mainnet
Decentralised validator set	Validators the operator does not control hold a majority of stake	not yet; testnet-0 is single-operator

2. What Aeva guarantees, and what it does not

The network's promise is deliberately narrow so that it can be kept. Aeva guarantees that the ledger is internally consistent: no right kind ever has more units in positions than its supply, no position is ever encumbered beyond its balance, no settlement is ever half executed, and no transfer ever reaches an account that the asset's policy excludes. It guarantees that what it records is final in one block and never reorganised. It guarantees that anyone can rebuild the entire record from the chain's events and check it against the chain's own accounting.

Aeva does not guarantee that an issued right corresponds to anything in the world. Issuance is the one act the protocol takes on trust; it is attributed to a named issuer account and recorded with the hash of the issuer's legal documentation. Whether a claim binds anyone off chain is a property of that documentation and of the law under which the issuer operates. Aeva likewise does not guarantee the honesty of an attester: an asset that names an attester in its policy trusts that attester, and the protocol enforces the policy as written. And the network does not guarantee the value of any bridged representation on another chain beyond what that chain's own security provides.

3. Threat model

We consider the following adversaries and the controls that address each. Controls are cross-referenced to the sections that describe them.

Adversary or failure	Objective	Primary controls
Malicious or compromised validator minority (below one third of stake)	Halt or reorganise the chain, censor transactions	BFT safety with any minority faulty; slashing and jailing for double-signing and downtime; sentry isolation (4)
Coalition above one third of stake	Halt the chain	Liveness requires two-thirds honest; halting is detectable and recoverable by governance; no reorganisation is possible without two-thirds (4)
Coalition above two thirds of stake	Rewrite history, steal funds	Not defended by the protocol; defended by validator diversity and economic cost, which testnet-0 does not yet provide (4, 12)
Bug in the state machine	Create units, move locked units, bypass policy	Invariants I1–I8 executed at every block with halt on violation; fuzzed oracles; deterministic replay (5, 9)
Malicious Solidity contract	Drain or mint rights through the EVM	No rights state in the EVM; precompiles route through the native path with policy; no issuer selectors; reentrancy and partial-revert tests (7)
Compromised issuer key	Mint units, change policy, freeze assets	Powers are explicit and attributable; issuance lock is irreversible; explorers display 'issuer can mint'; freeze never blocks recovery of escrow or income (6)
Compromised attester key	Grant eligibility to ineligible accounts	Trust is per asset; revocation by the attester; expiry; the asset can change its trusted attesters (6)
Bridge validator compromise	Mint native AEVA, drain escrow	Native canonical: routes cannot mint AEVA; per-route caps bound loss per window; governance pause; two-thirds ISM threshold (8)

Adversary or failure	Objective	Primary controls
Indexer or explorer compromise	Show a false picture of the ledger	The indexer never writes to the chain; reconciliation drift is displayed, never corrected; anyone can run their own replica (10)
Operator error during upgrade	Halt or fork the network	Governance-scheduled upgrades; supervised binary switch; rehearsed on a full replica before every real upgrade (11)
Denial of service on public endpoints	Make the network unusable	Validators are not public; sentries and RPC nodes are disposable and rate-limited; state-sync restores a node in minutes (4)

4. Consensus, validators and network topology

Aeva is a sovereign proof-of-stake network using Byzantine-fault-tolerant consensus. A block is final when validators holding more than two thirds of bonded stake have signed it. There is no probabilistic confirmation window and no dependence on another chain for finality or data availability. Safety (no two conflicting blocks are ever finalised) holds as long as less than one third of stake is faulty; liveness (blocks keep being produced) requires more than two thirds to be online and honest. Double-signing is detected from the evidence in the chain and punished by slashing bonded stake and permanent removal; prolonged downtime is punished by a smaller slash and temporary jailing.

Validators do not accept connections from the public network. Each validator connects only to its own sentry nodes, with peer exchange disabled and its peer identity marked private, so its address is never gossiped. Sentries and dedicated RPC nodes absorb peer-to-peer traffic and serve RPC, gRPC, REST and the EVM JSON-RPC behind a reverse proxy with rate limits and TLS. They are stateless with respect to consensus and can be rebuilt from state-sync snapshots in minutes. A private archive node with full history exists solely to feed reconciliation (Section 10) and is not exposed.

Consensus keys on testnet-0 are file-based with restrictive permissions and are generated on the validator host during the genesis ceremony; they never leave that host and are never transmitted. The target posture for a production network is a remote signer with hardware-backed keys and double-sign protection (tmkms or Horcrux), which is a prerequisite for mainnet rather than an option.

5. Protocol invariants executed in consensus

The controls that matter most are not policies about the code; they are properties the code checks about itself, in consensus, before it will produce the next block. At the end of every block each validator verifies:

```
I1 sum of positions = supply, for every asset and right kind
I2 sum of locks = encumbered <= units, for every position
I2b every active encumbrance has exactly its lock, and every lock has its
encumbrance
I4 payout module balance = distributed - paid, per denomination
I5 waterfall pool balance = deposited - paid, per denomination
I6 redemption pool balance = funded - paid - withdrawn, per pool
I7 every routed income checkpoint is backed by a lock of at least its units
I8 bridge escrow = outbound - inbound, per native route
```

The check runs on running sums maintained at every mutation, so it costs time proportional to the number of asset kinds rather than holders; a full scan over every position runs at a configurable period (every

thousand blocks on testnet-0) and on demand. A violation halts the network. This is intentional: a ledger that has detected an inconsistency in its own accounting must not produce another block until the cause is understood. The halt is a designed outcome with a runbook, not an accident.

6. Asset-level controls

6.1 Roles and attribution

Every power over an asset belongs to a named account: the issuer (schema, issuance, redemption, policy, freeze, waterfall, roles, liquidation), the operator (liens on assets that permit them, liquidation), the transfer agent and the custodian (records today, approval flows later). Every exercise of a power is a signed transaction attributed to that account and visible in the explorer. There is no administrative back door in the protocol: the network's governance can upgrade the software, but cannot move a unit of anyone's rights.

6.2 Policy enforcement

Transfer restrictions are expressed in a closed grammar (open; all-of; any-of; credential from named attesters; allowlist; maximum holders) with bounded size and depth. Evaluation is deterministic and runs against the state that would result from the whole transition, so it cannot be gamed by ordering. The same evaluation is applied whether the transfer arrives as a native message, a settlement leg, a foreclosure, or an ERC-20 call from a contract; contract accounts are subjects like any other. A denial names the rule that failed. Policy changes apply prospectively only; no change by an issuer can invalidate a holding that was valid when acquired.

6.3 Freeze, liens, issuance lock

An issuer can freeze an asset, which stops issuance, transfers and new settlements but never prevents cancellation of a settlement, release of an encumbrance, claiming of income or redemption: a freeze can immobilise an asset but cannot trap value owed to a holder. Liens, the one encumbrance created without the holder's signature, are possible only on assets whose issuer opted in, only by the operator, and only with a thirty-two-byte reference to the legal instrument; they are visible to the holder immediately. An issuer can lock issuance irreversibly, after which the supply of every kind can only fall.

6.4 Escrow and encumbrance safety

Locked units never leave a holder's position except through the two paths the protocol defines: release back to the holder, or transfer to the beneficiary through the same policy-checked route as any transfer. Escrow for a settlement is released by cancellation, by the counterparty's decline, or automatically at expiry by the network itself; a holder never depends on another party's cooperation to recover escrowed units. Income keeps accruing to the record holder on locked units unless the holder explicitly routed it.

7. Execution environment containment

Aeva offers an Ethereum Virtual Machine so that wallets and Solidity can be used; the design prevents it from becoming an attack surface against the asset model. No right, lock, settlement or payout is stored in EVM state. Rights are exposed to contracts as precompiles at deterministic addresses whose transfer functions execute the native transfer path, including policy evaluation; a denied transfer reverts with the failing rule as its reason. The precompiles expose no minting, burning or issuer functionality, and unknown selectors revert. Precompile actions never open a nested state cache, a rule established after testing showed that doing so breaks the EVM's revert semantics. Reentrant calls, partial reverts and out-of-gas inside a precompile are tested to leave state unchanged. Allowances saturate at the protocol's amount bound rather than rejecting the unbounded approvals that routers send.

The EVM integration is a third-party component (Cosmos EVM) that is itself pre-release and under audit by its maintainers. Aeva wires only its core, fee-market and ERC-20 modules and enables only the bank precompile alongside its own; staking, governance and interchain precompiles are disabled to minimise the surface. This dependency is one reason an independent audit is a mainnet prerequisite.

8. Bridge security

Interoperability with Robinhood Chain uses a permissionless message-passing bridge with warp routes. The rule that governs every route is that *native AEVA is canonical*: bridging out locks native units in escrow and mints a representation on the other chain; bridging in burns the representation and releases from escrow. No route can mint native AEVA, and a test asserts that no minting permission for the staking token exists outside the mint and staking modules. This is the single most important bridge property: a bridge that could mint the staking token would make the bridge, not the validator set, the security of the chain.

Each route carries rolling-window caps on outbound and inbound volume, set by governance, that bound the loss from any compromise to one window's allowance. Governance can pause a route, which rejects both directions. Messages are verified by an interchain security module whose signers are the network's validators with a two-thirds threshold; the signer set rotates by governance. The escrow invariant I8 is executed with the others. Bridged assets from other chains arrive as synthetic denominations whose backing is the other chain's escrow; their security is that of the other chain and the bridge, and they are labelled as bridged everywhere they appear.

9. Assurance: how the software is tested

Every module is developed against acceptance tests written before the implementation, and the release process is a fixed pipeline rather than a judgement call. For each release:

Fuzzing with oracles. Each module has a fuzz suite that executes ten thousand random operations, including deliberately invalid ones, against the chain and against an independent model of the intended semantics written in arbitrary-precision arithmetic. After every operation the model's accept-or-reject decision, every balance, every lock, every claimable amount and every invariant must match. Divergences to date have been found in the model, not the chain, and each is recorded.

Determinism. The indexer replays the chain's entire event history from genesis; two replays must produce byte-identical databases. Genesis export and import must reproduce the state exactly.

End-to-end. A single-node network and a multi-node rehearsal network run the full lifecycle through the command line and through browser automation: asset creation, issuance, splits, credentialed transfers, settlements, pledges, income, waterfalls, redemption, the EVM and the bridge, with accessibility checks on every explorer page.

Upgrade rehearsal. Before any software upgrade reaches a live network, the upgrade is executed on a full replica built from the previous release, with the indexer and explorer running through it and reconciliation required to show zero drift on the other side. The operational failure modes found in rehearsal (a late binary, a configuration that silently disables an API) are numbered incidents in the runbook with their recoveries.

Performance under load. A load generator submits a mixed workload and reports block time, latency percentiles, settlement-within-one-block rates and failures by error code; the same harness caught a consensus-halting regression in a pre-release build before any human reviewer did.

Hygiene. No floating-point arithmetic anywhere in the protocol, indexer or clients, enforced by lint. No secrets in the repository, enforced by a scanner on every commit. No binaries in the repository. Address parsing and configuration access on hot paths are forbidden by lint.

10. Independent verification

A ledger that claims to be consistent about itself should be checked by something that does not share its code path. Aeva's reference indexer consumes block results from any node, derives the entire rights graph from typed events alone, and at every configurable interval compares its derived tables against the chain's own invariant status: supply, position sums, lock sums, encumbrance sums, module balances, bridge escrow and per-holder claimable amounts. A disagreement is recorded as *drift*, counted, alerted on and displayed on every page of the reference explorer. It is never corrected in place; the remedy is to find the cause and replay. Anyone may run a replica against a public node and reach the same conclusion. The indexer has no write path to the chain, so its compromise cannot affect the ledger, only the picture of it, and that picture is checked against the ledger continuously.

11. Operations and incident response

11.1 Monitoring and alerting

Every node exports metrics; the reconciler exports drift and lag. Pages are raised for reconciliation drift, for a stalled indexer, for the absence of a new block, for a validator missing a threshold of recent blocks, for low disk, and for faucet exhaustion. Alert rules are unit-tested against synthetic time series so that a rule cannot silently stop firing.

11.2 Upgrades

Software upgrades are proposed and voted on chain, scheduled at a height, and switched by a supervisor on every node at that height. Every upgrade is rehearsed as described in Section 9 before the proposal is submitted.

11.3 Incident classes and responses

Class	Detection	Response
Invariant violation	Chain halts by design	Preserve state; identify the transition; fix; coordinated restart or rollback by governance; post-incident report
Reconciliation drift	Drift counter and page	Do not clear; identify cause; replay the indexer; if the chain is at fault, escalate to invariant handling
Validator key compromise	Double-sign evidence or operator report	Slashing is automatic; validator rotates keys and re-bonds; ISM signer set rotated by governance
Bridge anomaly	Escrow mismatch, cap hits, unexpected message volume	Pause the route by governance; investigate; resume with adjusted caps
Public endpoint outage	Health checks	Rebuild the node from snapshot; validators unaffected

11.4 Backup and recovery

The archive node's data and the indexer database are snapshotted daily, and the restore drill is part of the release checklist: a node restored from snapshot must replay to a database byte-identical with production. State-sync snapshots published by the sentries let any node rejoin from a trusted height without replaying history.

12. Known gaps and roadmap

We consider it part of a security posture to state plainly what is missing.

Validator decentralisation. Testnet-0 is operated by a single operator with a small validator set. The network is sovereign from genesis but not decentralised; it becomes so when validators the operator does not control hold a majority of stake. Until then, the operator is a trusted party for liveness.

Consensus signing. File-based consensus keys are acceptable for a testnet and not for mainnet. A remote signer with hardware-backed keys is a mainnet prerequisite.

External audit. No independent audit has yet been performed on the native modules or on the EVM integration, and the EVM component is itself pre-release. Both are mainnet prerequisites.

Bridge on a live counterparty chain. The bridge is tested end to end against a local EVM and against the reference rehearsal network; deployment against Robinhood Chain's testnet is the next step, and mainnet routes will begin with conservative caps.

Ledger hardware signing for users. Legacy Amino signing for hardware wallets is not yet supported for native transactions.

Bug bounty. A public bounty programme is planned to coincide with the audit.

13. Responsible disclosure

Security reports are welcome and taken seriously. Researchers who find a vulnerability are asked to report it privately through the security contact published at aevachain.com, to allow a reasonable period for a fix before public disclosure, and to avoid actions that degrade the network or affect other users' data. AevaChain commits to acknowledging reports promptly, to keeping reporters informed, and to crediting them if they wish. Testnet-0 is a test network with no real value at stake; reproducing an issue on it is encouraged, and testnet funds may be freely used to do so.

14. Summary for risk and compliance functions

Aeva's security model is built on the principle that the properties which matter are enforced by consensus and verified independently, rather than promised by policy. The ledger cannot hold more units of a right than exist, cannot encumber more than a holder has, cannot settle half a trade, and cannot deliver a restricted asset to an ineligible account, whether the instruction arrives natively, through settlement, through foreclosure or through a smart contract. Its records are final in one block, never reorganised, and reconstructible by anyone from public events. Personal data is never stored. The staking token cannot be minted by any bridge. The network halts rather than continue on inconsistent accounting. What remains before a production network is operational and organisational rather than architectural: a decentralised validator set, hardware-backed signing, an independent audit and a bounty programme, each listed above with its status.