The Six Pillars of Boing Network

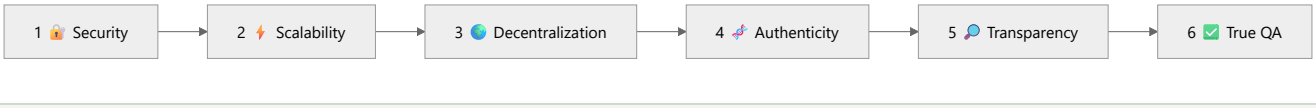
Authentic. Decentralized. Optimal. Quality-Assured.

August 2026 · boing.network · boing.observer

> 👋 ***Everyday users:*** *this is the why. Use it when you want to know what Boing will not trade away.*
> 🛠️ ***Developers:*** *each pillar maps to shipped behavior — VM, RPC, QA, consensus.*
> 🏗️ ***Operators:*** *when two good ideas fight, apply this order as an engineering rule, not a slogan.*

This is the written form of the six pillars. It is meant to be readable: technical enough that an engineer can map each pillar to shipped behavior, casual enough that you do not need the full spec in your lap.

For stack, crates, and doc pointers, see BOING-NETWORK-ESSENTIALS.md. For the QA pipeline itself, see QUALITY-ASSURANCE-NETWORK.md.



Why pillars, and why an order?

Every chain says it wants security, speed, and openness. The interesting part is what you do when those goals collide.

Boing is an independent L1. We did not inherit another chain's trade-offs, so we wrote ours down. When two good ideas fight, we apply this order — not as marketing, as an engineering rule:

Security → Scalability → Decentralization → Authenticity → Transparency → True quality assurance

A short example: a faster block interval that weakens BFT safety loses to Security. A throughput trick that only a handful of operators can run loses to Decentralization. A "just ship it and moderate later" deploy path loses to True QA — but QA never outranks a real safety bug.

The last pillar is last because it is a *product* of the others working. It is not optional. It is just not allowed to punch a hole in consensus to look strict.

1. 🔐 Security

Plain terms. Correctness first. We would rather be a little slower than be wrong, slash the wrong person, or let a malformed tx take down a block.

On chain. Consensus is proof-of-stake HotStuff BFT. Signatures are Ed25519. Hashes are BLAKE3. Equivocation is detected and slashable. Execution is metered: every opcode and tx type has gas, and native fees are `ceil(gas_used / 21_000)` BOING so a simple transfer costs one token, not tens of thousands. Public RPC is rate-limited. Failed or underfunded transactions are isolated so they cannot stall block production.

What that means for you. If it is a safety property, it belongs in the protocol, not in a slide. Advisories and incident process live in SECURITY-STANDARDS.md. If you are integrating a wallet or dApp, treat the node as hostile until proofs and receipts check out.

---

2. ⚡ Scalability

Plain terms. The chain should feel snappy without quietly giving up the other pillars.

On chain. Target block time is about two seconds. Transfers that do not touch the same accounts can execute in parallel; access-list batching feeds that scheduler. The VM is stack-based and purpose-built (Boing opcodes only — no foreign bytecode engine in consensus). State is a sparse Merkle tree today, with Verkle as the stated target. Networking is libp2p (TCP, Noise, gossipsub, request-response).

What that means for you. Scale is "more useful work per second under the same security assumptions," not "turn off checks." If a shortcut needs a permissioned sequencer or a trusted VM, it is the wrong shortcut.

---

3. 🌎 Decentralization

Plain terms. Anyone who meets the protocol rules can participate. Nobody gets a special door.

On chain. There is no validator whitelist: stake in, validate. Governance is phased (proposal → cooling → execution) and time-locked. Protocol QA is the same idea in another layer: automation first, and when the node is unsure, a **community QA pool** decides — not a single operator with a kill switch.

Hosted testnet bootnodes (libp2p multiaddrs):

- `/ip4/169.155.48.188/tcp/4001`

- `/ip4/109.105.220.118/tcp/4001`

Public JSON-RPC is `https://testnet-rpc.boing.network/`. Those are conveniences for joining, not a chokepoint for consensus.

What that means for you. Run a node if you want. Disagree in governance if you want. Unsure deploys go to the pool, not to whoever happens to run the explorer.

---

4. 🧬 Authenticity

Plain terms. Boing is its own chain. Not a fork with the logos swapped, not a framework wrapped around someone else's VM.

On chain. Custom stack-based VM, HotStuff BFT, BLAKE3, Ed25519, native DEX discovery on L1, protocol-level deploy QA. The identity is one network, one account model, one execution story. We will steal a *good idea* when it is demonstrably better. We will not rent another L1's consensus, bytecode, or culture to look familiar.

What that means for you. Tooling should feel straightforward — deploy, sign, call — without pretending this is EVM. The SDK and RPC catalog (`boing_getRpcMethodCatalog`, `boing_getNetworkInfo`) are the map. If a tutorial says "just use Solidity unchanged," it is the wrong tutorial.

---

5. 🔎 Transparency

Plain terms. Trust comes from being able to check, not from being asked to believe.

On chain. The protocol is open source. Specs, runbooks, and QA registries are public. Signing payloads are meant to be human-readable. QA rejections carry a `rule_id` and a `message` so a deployer can fix the thing that actually failed. Anyone can read the live rule set (`boing_getQaRegistry`) and pool config (`boing_qaPoolConfig`) on the RPC they trust.

The explorer publishes a live QA transparency dashboard from that same public RPC: queue, parameters, registry JSON.

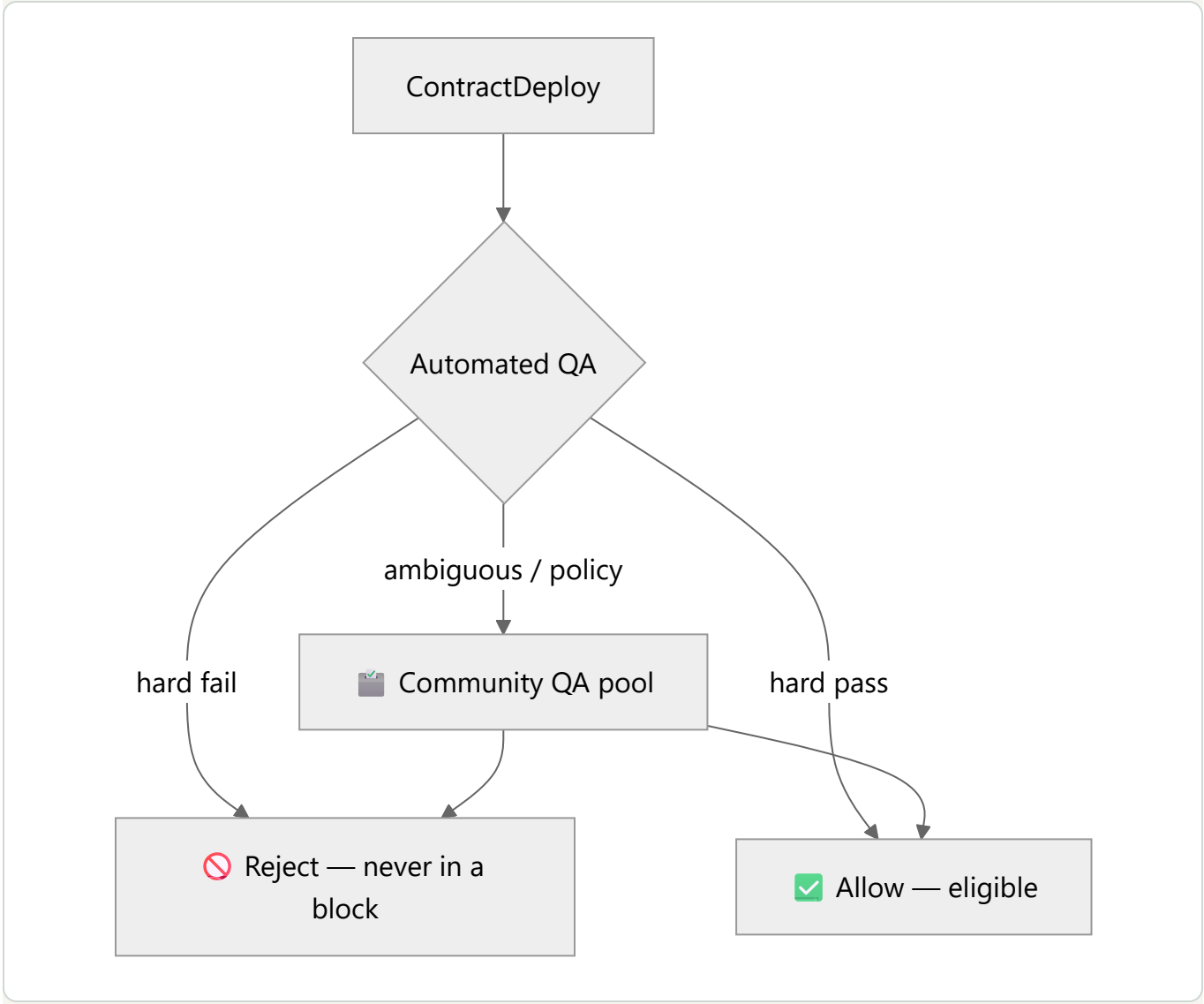
What that means for you. If a rule cannot be pointed at, it is not a rule yet. Prefer `boing_getQaRegistry` on the endpoint you use over a screenshot of a markdown file — operators can change policy; the node tells you what it is enforcing *now*.

---

6. ✅ True quality assurance

Plain terms. Assets do not land on-chain and then get “cleaned up.” They pass the bar first. Memes are allowed. Malice is not.

On chain. Every `ContractDeploy` is classified **allow**, **reject**, or **unsure** before inclusion.



Outcome	What happens	Typical reasons
Reject	Never enters a block	Empty or oversized bytecode, invalid opcode, malformed stream, blocklisted hash, known scam pattern, invalid purpose category
Allow	Eligible for inclusion	Hard rules pass; purpose is valid or omitted; no policy that forces review
Unsure	Referred to the community QA pool	Soft / ambiguous cases (for example purpose `"other"` with almost no description), or a category that governance always wants reviewed

Checks include the opcode whitelist, well-formedness, bytecode blocklist, optional deploy metadata content policy, scam patterns, and purpose declaration. Purpose categories such as meme, community, and entertainment are first-class — we do not treat culture as a scam signal.

Assigned reviewers can Allow / Reject Unsure items (`boing_qaPoolVote`). That is a protocol vote, not a vibe check in a private chat.

What that means for you. Pre-flight with `boing_qaCheck` (the explorer's QA check is a thin UI on that RPC). Declare a real purpose. If you are rejected, read the `rule_id` . Canonical malice and governance-mutable lists are in QUALITY-ASSURANCE-NETWORK.md; live status is on QA transparency.

How they fit together

Security keeps the ledger honest. Scalability keeps it usable. Decentralization keeps it from becoming a club. Authenticity keeps the architecture ours. Transparency lets outsiders verify the first four. True QA is how we refuse to dump low-quality or hostile bytecode onto a ledger we just spent five pillars trying to keep clean.

If you remember one sentence: **we automate the boring judgments, we decentralize the hard ones, and we do not trade safety for theater.**

Further reading

Document	Why it is next
BOING-NETWORK-ESSENTIALS.md	Stack, crates, design philosophy
QUALITY-ASSURANCE-NETWORK.md	Allow / Reject / Unsure, malice definition, governance-mutable rules
TECHNICAL-SPECIFICATION.md	Crypto, VM, gas, RPC shapes
RPC-API-SPEC.md	JSON-RPC, including QA and DEX discovery
SECURITY-STANDARDS.md	Advisories and incident expectations
TESTNET.md	How to join the public testnet

Published copies of this write-up: boing.network/about and boing.observer/about (PDF).