TWIX CHAIN

SOVEREIGN EXECUTION • PRIVATE SECURITY INTELLIGENCE • AUTONOMOUS DIGITAL INFRASTRUCTURE

TWIX Lock Flagship Whitepaper and Ecosystem Roadmap

From implementation back to intent — then forward into verified execution.

WHITEPAPER v1.1.0 • SEPTEMBER 2026

twixchain.com • Mainnet: twix-1 • EVM: 1415006552 / 0x54574958

Document status and interpretation

This paper is the master product and technical vision for TWIX Chain as of September 2026. It deliberately separates live infrastructure from demonstrated frontend workflows, planned production systems and longer-horizon research. The roadmap is a target sequence, not a promise of dates, feature completion, token price, revenue or security outcomes.

LIVE TWIX Mainnet, canonical public network manifests, Cosmos/EVM endpoints, mainnet website, validator directory, governance surface, Explorer, Data Attestation application, Trading Tools, documentation, and the completed 15/15 hardened Cosmos EVM remediation qualification.

DEMONSTRATED TWIX Lock frontend lifecycle: repository registration, challenge-file generation, public GitHub verification path, wallet authorization UX, private software-twin console, simulated Watch/Shield, private finding/bounty flows and TWIX compute concepts.

PLANNED TWIX Lock production control plane and workers, Lock Rail, bounty network, TWIX Twin, TWIX Gateway, TWIX Exchange expansion, GameVerse, broader Xeteria integration and related APIs/SDKs.

RESEARCH Earlier Cosmos Hub Meta Account, EIP-191 binding and native-authz work; FENS-inspired implementation-to-specification modeling; future formal and autonomous execution research.

SECURITY BOUNDARY

TWIX Lock is designed for repositories and systems whose owners explicitly authorize analysis. Attack simulation belongs inside isolated authorized sandboxes. This paper does not publish or prescribe exploit techniques against third-party production systems.

Contents

1. Executive Summary

TWIX Chain is an independent Cosmos SDK network with native Ethereum-compatible execution. It is designed to let Cosmos-native and EVM-native users, wallets, applications and developers operate against one sovereign network rather than treating interoperability as a bolt-on product. Its production network identity is twix-1, with EVM chain ID 1415006552 (0x54574958), native TWIX denominated as atwix with 18 decimals, and standard public Cosmos and EVM endpoints. Before positioning TWIX as a security platform, the chain stack itself was upgraded against a 15-point Cosmos EVM module review and completed the defined post-remediation qualification at 15/15 PASS.

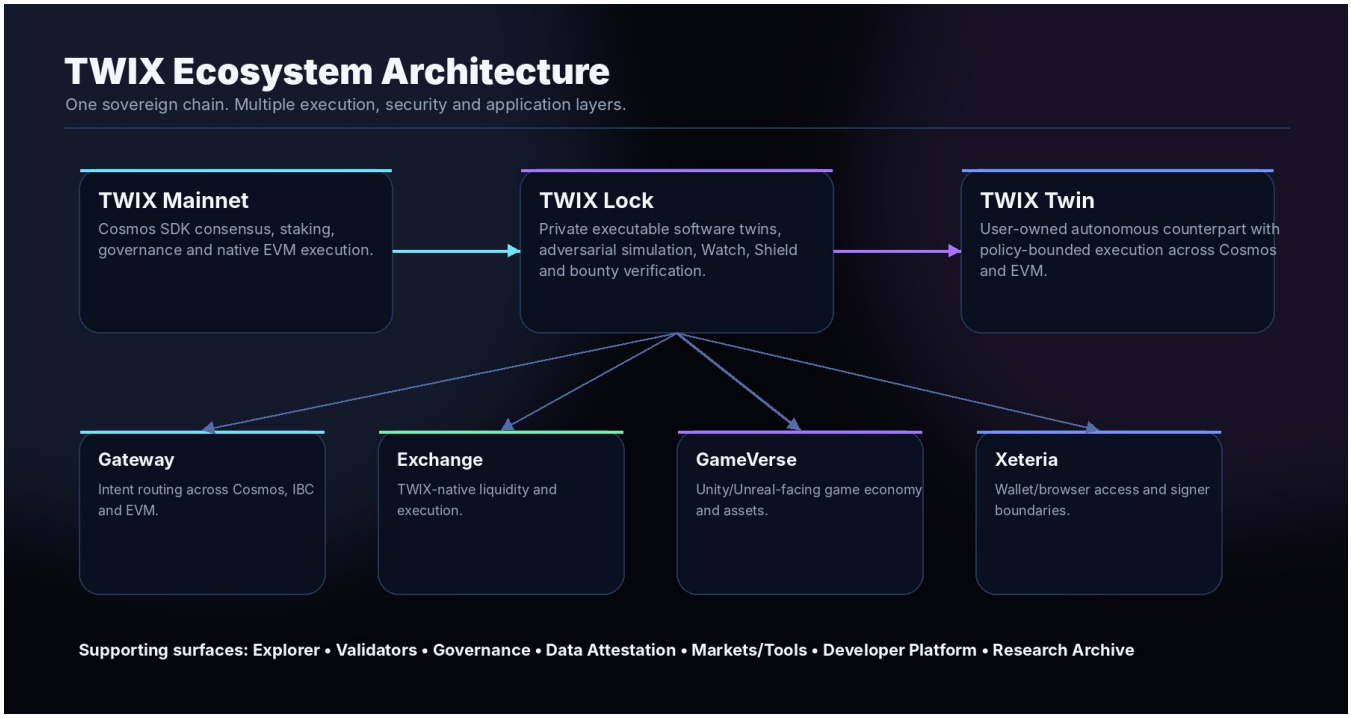
The chain is infrastructure. The flagship product is TWIX Lock: a private software security platform that creates an executable twin of an authorized repository or deployed system, reconstructs what the software actually does, searches for invariant failures in isolated sandboxes, verifies serious findings through deterministic reproduction, monitors behavioral drift continuously, and supports pre-authorized defensive response policies without allowing probabilistic AI to directly control production systems.

TWIX Lock is built around a central consent rule: no proof of authority, no adversarial simulation. A target must prove repository control by committing a Lock challenge file and then prove cryptographic authority by signing a TWIX transaction from the exact wallet named in that file. This Dual-Proof Authorization binds repository state, wallet authority, scope and expiration into a reproducible authorization record.

Beyond Lock, the TWIX ecosystem expands toward TWIX Twin, TWIX Gateway, TWIX Exchange, Markets/Trading Tools, GameVerse, Xeteria wallet/browser integration, Data Attestation, Explorer, validators, governance and a developer platform. Each product has a distinct role; together they create recurring native utility for TWIX through gas, security compute, simulation, monitoring, bounty escrow, reporting custody, routing, exchange execution, automation and application economics.

CORE THESIS

TWIX is not positioned as “another EVM chain.” Its differentiation comes from using sovereign Cosmos + EVM execution as a base for products that require cross-environment authority, verifiable security intelligence, autonomous policy and application-level economic activity.



2. Why TWIX Exists

Crypto infrastructure often forces users and developers to choose between execution environments instead of choosing outcomes. Cosmos offers sovereign application chains, staking, governance and IBC-oriented interoperability. Ethereum-compatible tooling offers Solidity, EIP-1193 wallets, mature contract infrastructure and broad developer familiarity. TWIX is designed to expose both surfaces on one independent network while preserving clear signer, consensus and application boundaries.

The original TWIX research line focused on wallet interoperability: demonstrating that MetaMask-originated intent could participate in Cosmos execution without exporting private keys or pretending an EVM wallet is a native Cosmos signer. The production mainnet broadens that idea. TWIX now becomes the chain on which Cosmos and EVM interfaces are native parts of the same system, while the earlier Meta Account and binding work remains preserved as research evidence rather than being confused with current mainnet configuration.

The product strategy follows from the same principle: abstract complexity without hiding authority. Wallets should know what they sign. Autonomous agents should operate inside explicit policy. Security systems should prove findings without publishing exploit material. Cross-chain routing should simplify execution while showing the route and cost. Gaming should make assets portable without turning gameplay into a wallet prompt simulator.

3. TWIX Mainnet: Sovereign Cosmos + EVM Infrastructure

Property	Current production value
Network	TWIX Mainnet
Cosmos chain ID	twix-1
EVM chain ID	1415006552 / 0x54574958
Native asset	TWIX
Base denomination	atwix
Decimals	18
Account prefix	twix
Validator prefix	twixvaloper
BIP-44 coin type	60
HD path	m/44'/60'/0'/0/0
Daemon	twixd v0.1.0
Cosmos RPC	https://rpc.twixchain.com
Cosmos REST	https://api.twixchain.com
EVM JSON-RPC	https://evm.twixchain.com
EVM WebSocket	wss://ws.twixchain.com

Genesis identity is published through the canonical network manifest, including the genesis SHA-256 and block-one identity. TWIX publishes machine-readable EVM, Keplr and unified wallet objects from the primary domain so applications do not need to maintain disconnected network metadata.

The public launch surface intentionally exposes wallet and developer interfaces through hardened TLS boundaries while raw backend ports and high-risk administrative/debug namespaces remain outside the public firewall surface. Consensus validator metadata is public; consensus signing material is not. This perimeter posture is only one layer: the underlying Cosmos EVM implementation was also hardened against the 15 audit targets documented in the next chapter.

OPERATIONAL PRINCIPLE

Public where useful. Closed where dangerous. TWIX favors standard read/transaction interfaces at the edge and keeps privileged node functions private.

4. Hardened Cosmos EVM Stack: 15/15 Remediation Qualification

TWIX security begins below the product layer. The August 22, 2026 Cosmos EVM module review identified fifteen concrete security and reliability targets. The highest-risk concentration was the boundary where EVM execution interacts with Cosmos SDK state, gas accounting, authorization, token conversion, events, callbacks and IBC behavior. TWIX upgraded the complete chain build against those targets and completed the defined post-remediation qualification with 15 of 15 targets passing.

The qualification result is a checkpoint, not a permanent-security claim. “15/15 PASS” means every remediation target defined from that review passed the TWIX post-upgrade qualification criteria. Future code, dependencies, governance changes and operational configuration still require continued review and regression testing.

HARDENED QUALIFICATION · 15 / 15 PASS

Source-level remediation complements TWIX’s restricted public RPC/firewall posture. The review and qualification focus on the implementation boundaries that can create cross-runtime state, accounting, authorization, availability or IBC failure modes.

4.1 Remediation matrix

The table below records the public engineering treatment for all fifteen audit targets. It intentionally describes the remediation and qualification domain without publishing exploit recipes, private proof-of-concept material or weaponized reproduction sequences.

#	Audit target	TWIX hardened treatment	Qualification
01	Phantom Vesting Precompile (HIGH)	Remove phantom-active behavior; require advertised/active precompile addresses to resolve to real registered implementations and fail deterministically.	PASS - startup/registration validation and deterministic error behavior.
02	Trace API trusts caller-supplied block data (HIGH if public)	Constrain public tracing to canonical chain context and operator-defined replay/workload limits; separate synthetic/internal tracing from public historical claims.	PASS - canonical-state and replay-boundary qualification.
03	Caller-selected excessive trace timeout (HIGH if public)	Use server-controlled trace budgets and independent output/resource ceilings rather than caller-selected long durations.	PASS - timeout clamp and resource-boundary qualification.
04	Disabled token-pair semantics inconsistent (HIGH/MEDIUM)	Enforce one clear enable/disable policy across relevant ERC20 transfer, approval, conversion and ICS20-facing behavior.	PASS - disabled-pair semantic matrix.
05	Gas meter can corrupt accounting on overflow (HIGH/MEDIUM)	Check overflow before mutation so previously consumed gas is preserved; qualify boundary and recovery semantics explicitly.	PASS - overflow boundary/recovery regression.

#	Audit target	TWIX hardened treatment	Qualification
06	Failed EVM transactions can persist native hook state (HIGH with writable hooks)	Treat failed EVM execution as a cross-runtime atomicity boundary; prevent unintended/unmetered native application-state side effects outside explicit policy.	PASS - failed-execution and native-hook failure injection.
07	WebSocket server missing important timeouts (MEDIUM/HIGH public)	Bound connection lifetime, request context, message/connection resources and internal JSON-RPC client behavior.	PASS - connection and resource-exhaustion qualification.
08	Debug profiling can write caller-chosen paths (HIGH if exposed)	Keep dangerous debug/profiling outside public interfaces and constrain filesystem paths, durations and profiling resource use.	PASS - public exposure and profiling-boundary qualification.
09	ERC20 balanceOf spendable vs owned balance (MEDIUM)	Harden ERC20 accounting semantics so integrations do not confuse total ownership with transfer spendability for locked/vesting state.	PASS - locked/vesting accounting regression.
10	Virtual fee collection can panic on bad metadata (MEDIUM/HIGH)	Validate gas-token metadata requirements before fee processing and return explicit configuration/error results instead of ordinary-transaction panic paths.	PASS - invalid-metadata fee-path qualification.
11	IBC destination callback writes before final safety check (MEDIUM)	Perform final safety/invariant checks before durable cache write so callback behavior remains locally atomic beyond outer-cache assumptions.	PASS - direct callback rollback regression.
12	ICS20 precompile missing adversarial suite (HIGH)	Build regression coverage around the ERC20/native/IBC/callback/gas/revert boundary including failure, timeout, malformed-input and replay-oriented cases.	PASS - adversarial ICS20 qualification.
13	ERC20/WERC20 gas may underprice native Cosmos work (HIGH/MEDIUM)	Treat native work as part of EVM gas economics; fixed compatibility costs must remain bounded by measurement and worst-case regression.	PASS - gas-budget and worst-case native-work qualification.
14	Example chain enables every stateful precompile (MEDIUM/HIGH)	Use explicit production allowlisting/least privilege instead of inheriting a broad demo/example stateful-precompile profile.	PASS - production precompile allowlist qualification.
15	Txpool reset can panic inside long-lived worker (MEDIUM)	Contain reset/recheck faults as recoverable operational errors or controlled supervisor conditions rather than arbitrary worker panics.	PASS - reset/rechecker fault-path regression.

4.2 Qualification philosophy

The remediation program emphasized failure-injection and regression rather than a narrow happy-path test. Cross-runtime rollback, callback failure, IBC/ICS20 behavior, gas boundaries, disabled capability state, invalid metadata and long-lived worker faults were treated as security-relevant consequences. This is the same philosophy TWIX Lock later formalizes for customer software: discover an assumption, reconstruct the behavior, challenge the failure condition, reproduce it, remediate it and re-test the exact condition.

4.3 Defense in depth

The source-level remediation is reinforced by the production deployment boundary: unsafe Comet methods, EVM admin/personal/debug/trace/txpool namespaces and raw backend services are not part of the normal public interface; unprotected transactions and insecure unlock behavior are disabled; application keys remain separated from consensus signing authority; and validator health is monitored continuously. Source hardening and perimeter hardening are complementary controls, not substitutes for each other.

5. Product Architecture and Ecosystem Map

The ecosystem is organized into layers so the chain, applications and research history remain unambiguous.

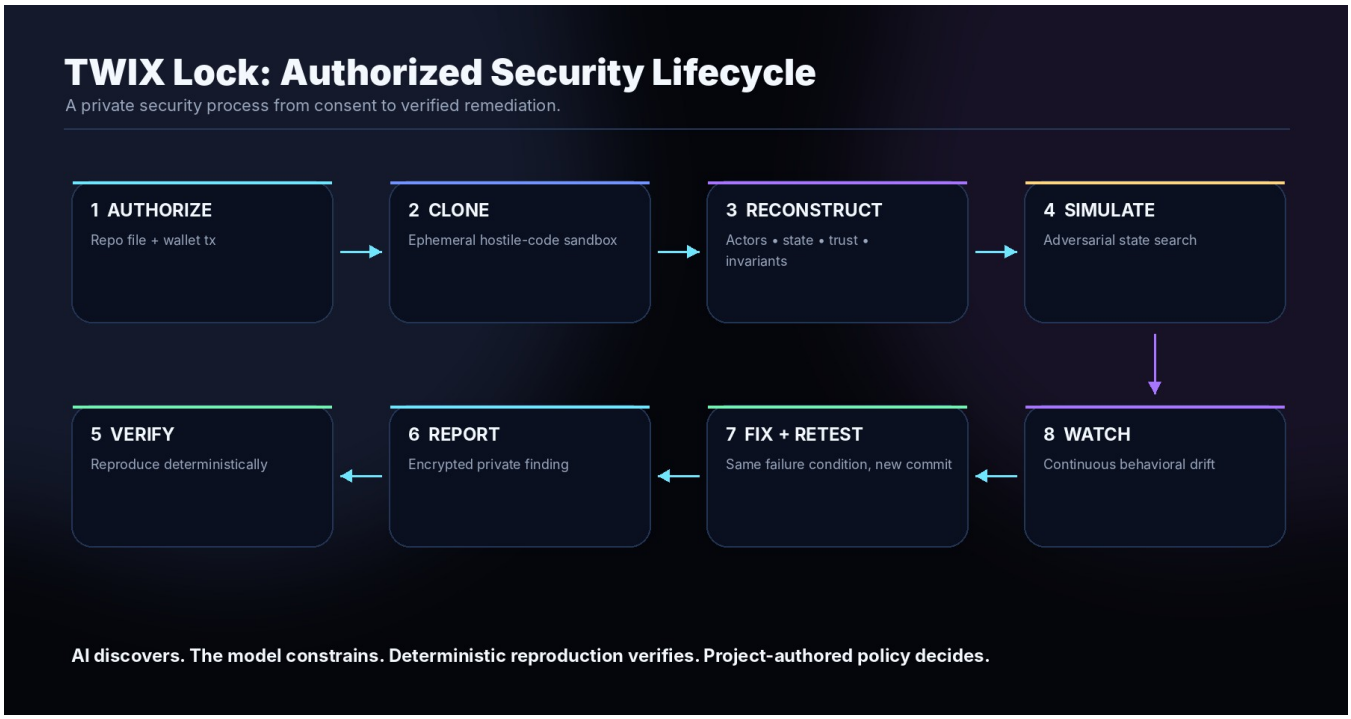
Layer	Products / surfaces	Purpose
Consensus & execution	TWIX Mainnet, validators, staking, governance, native EVM	Block production, economic security and sovereign state.
Visibility & trust	Explorer, docs, manifests, network/security pages	Make network state and integration data inspectable.
Flagship security	TWIX Lock	Private executable software twins, simulation, Watch, Shield and bounty verification.
Autonomous execution	TWIX Twin	Policy-bounded digital counterparts acting across Cosmos and EVM.
Interoperability	TWIX Gateway	Intent-based asset/action routing across Cosmos, IBC and EVM.
Liquidity & intelligence	TWIX Exchange, Markets, Trading Tools	Execution, liquidity, risk modeling and market analysis.
Gaming	TWIX GameVerse + SDKs	Game economies, player-owned assets, Unity/Unreal integration and marketplaces.
Wallet / user layer	Xeteria	TWIX-aware wallet/browser access with preserved signer boundaries.
Evidence infrastructure	Data Attestation, future Lock Rail	Signed physical-data commitments and confidential security evidence.
Developer platform	RPC/REST/EVM, SDKs, APIs, manifests	Build, integrate and automate against a documented surface.
Research archive	Meta Account, binding, peer review, FENS-adjacent research	Preserve evidence and experimental lineage without confusing current mainnet state.

6. TWIX Lock: The Flagship

TWIX Lock is the primary differentiating product proposed for TWIX. It is not a static code scanner and not an “AI audit” wrapper. Every authorized target becomes a private, reconstructable software

twin whose implementation, dependencies, authority paths, state transitions and expected invariants can be modeled and repeatedly challenged.

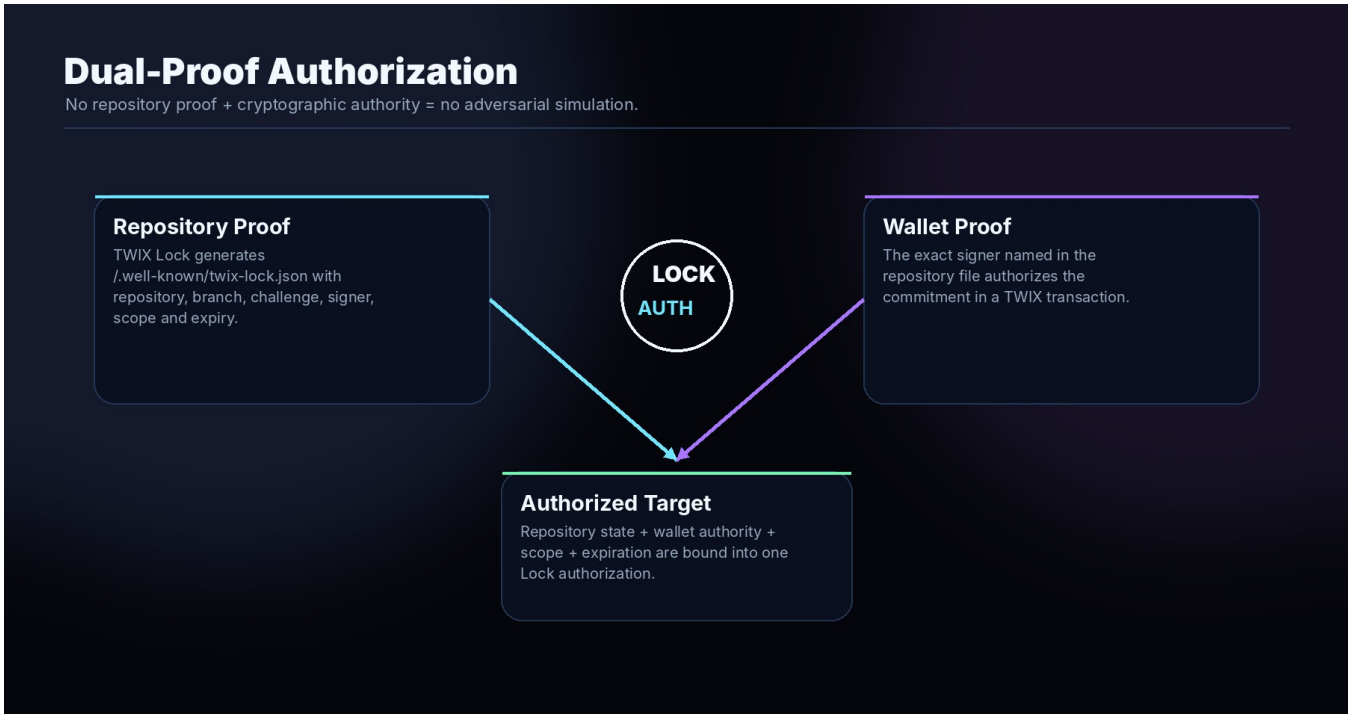
The Lock lifecycle is: authorize, clone, reconstruct, analyze, simulate, verify, report, remediate, re-test and watch. The same model can later support Shield policies, bounty verification and continuous behavioral-drift detection.



The business objective is to move software security from episodic audits toward continuous coverage. A project should eventually think in terms of “Lock coverage” in the same way it thinks about uptime monitoring, incident response or insurance: the system is continuously understood, re-tested when it changes and able to prove whether a previously demonstrated failure condition has been removed.

7. Dual-Proof Authorization

A public repository URL does not constitute authorization for adversarial simulation. Lock therefore requires two independent proofs before high-risk analysis capabilities unlock.



7.1 Repository proof

Lock generates a one-time challenge document to be committed at /.well-known/twix-lock.json on the repository and branch being authorized. The file binds the repository, branch, challenge, authorized wallet, capability scope and expiration. Production verification should pin the exact commit containing the authorization file.

7.2 Wallet proof

The wallet named by the repository file then authorizes the same commitment on TWIX. The platform verifies the cryptographically authenticated transaction sender rather than trusting an address typed into a form. The sender must match the repository-declared authority. Scope upgrades, renewals and revocation can use fresh commitments. Production can support either Cosmos-side or EVM-side authorizers, but each address must be explicitly named and verified; Lock should never infer that two account formats are controlled by the same key unless that relationship is separately proven.

7.3 Capability scope

- clone — acquire the authorized source state
- build — execute controlled build processes in isolated infrastructure
- analyze — reconstruct architecture, authority and behavior
- simulate — run adversarial state exploration inside the sandbox
- monitor — continuously re-analyze approved repository/deployment changes
- bounty — accept private researcher submissions and verified payouts
- shield — enable separately governed production-defense adapters

RULE

No authorization file + matching TWIX wallet authority = no adversarial simulation.

8. Private Software Twins and Sandbox Isolation

Submitted repositories must be treated as hostile. Dependency scripts, build hooks, test binaries and supply-chain components can all execute arbitrary code. Lock therefore separates the control plane from execution workers and assumes analysis targets can attempt to escape or abuse their environment.

- Control plane: project metadata, authorization, billing, scheduling and policy. It does not execute target code.
- Acquisition workers: fetch exact source/commit state, manifests, submodules and approved artifacts.
- Build workers: disposable, resource-capped environments used to compile or assemble the target.
- Simulation workers: stronger isolation for running the target, local chains/forks and adversarial state exploration.
- Network policy: no unrestricted outbound access by default; allow only explicitly required dependency or test endpoints through controlled mechanisms.
- Persistence: preserve source hashes, environment definitions, model state, test corpus and evidence; do not keep expensive VMs alive unnecessarily.

The product therefore maintains a persistent logical twin, not necessarily a permanently running virtual machine. Clean workers can be rehydrated from the stored environment definition whenever Watch, simulation or remediation verification requires them.

9. Implementation Reconstruction and FENS-Inspired Verification

The conceptual origin of Lock is an inversion of a core FENS idea. FENS is oriented around explicit rules, deterministic execution and consequences. TWIX Lock works from the other direction: given an implementation and observed behavior, infer the system model that best explains what can happen, then challenge that model.

Direction	Question	Output
FENS-style specification → execution	Given explicit rules, what happens?	Deterministic consequence.
TWIX Lock implementation → model	Given code and behavior, what are the effective rules?	Actors, authority, state, dependencies, invariants and failure conditions.
Long-term convergence	Does implementation actually match intended specification?	Behavioral equivalence/difference evidence.

The AI layer may propose relationships, assumptions or invariants. Those proposals do not become facts merely because a model generated them. Lock should compile hypotheses into machine-testable structures and accept a security claim only when the relevant behavior can be reproduced under controlled conditions.

DECISION BOUNDARY

AI discovers. Determinism decides.

10. Adversarial Simulation and Deterministic Proof

Lock simulation is designed to search the authorized twin for sequences that invalidate modeled assumptions. The important distinction is that the platform attacks the sandbox model, not an unconsenting production target.

Candidate simulation families include state-transition ordering, callback/failure behavior, privilege changes, upgrade paths, accounting consistency, dependency failures, stale external state, cross-environment synchronization and other system-specific invariant challenges. Production implementation should be capability-aware and constrained by the target authorization scope.

A suspected issue should pass a validation ladder: hypothesis → controlled reproduction → sandbox reset → independent reproduction → impact classification → private evidence package. Unreproducible hypotheses remain hypotheses.

FINDING STANDARD

A Lock finding is not “the AI thinks this is vulnerable.” A verified finding is a repeatable violation against an authorized target state with a preserved environment and evidence commitment.

11. Lock Watch and Shield

11.1 Watch: behavioral drift

Watch monitors approved repositories, releases, dependencies, deployments and governance/configuration changes. The key question is not merely “what lines changed?” but “what changed in the executable model?” A small source diff can materially alter authority, state reachability or an invariant dependency.

11.2 Shield: pre-authorized response

Shield is the proposed defensive runtime layer. It must never allow an LLM to arbitrarily pause a protocol, move funds or execute privileged actions. A production response requires a deterministically verified condition and a project-authored policy that already permits a specific action through a defined adapter.

State	Meaning	Example policy response
Green	Expected behavior	Observe and record.
Yellow	Unexpected model drift	Alert maintainers and increase simulation depth.
Orange	Verified degradation of a protected assumption	Apply pre-authorized limits or disable a narrow risky route.

State	Meaning	Example policy response
Red	Reproducible critical invariant violation	Invoke an explicitly pre-approved emergency adapter while preserving safe recovery operations where possible.

The governance sequence is intentionally separated: AI hypothesis → model condition → deterministic reproduction → policy evaluation → human/organization authorization boundary → adapter → evidence/attestation.

12. Lock Rail: Confidential Findings and Remediation

Security findings should never become public vulnerability payloads. Lock Rail is the proposed confidential reporting layer derived from the same architectural principle used by Securail: public proof does not require public data. Securail's existing architecture demonstrates local payload encryption with AES-256-GCM and recipient-bound X25519 key wrapping; Lock Rail can adapt that pattern while minimizing even more public metadata because vulnerability timing and access activity can themselves be sensitive.

A verified finding package can contain the affected source state, model failure, deterministic reproduction evidence, impact analysis, remediation guidance, environment definition and engine/version metadata. The package is encrypted for authorized recipients and stored privately. Public TWIX state, if used, contains only commitments or authorization events—not exploit calldata, PoCs, vulnerable code excerpts or reproduction instructions.

- Recipient-bound access: security authorities receive encrypted finding packages rather than emailed plaintext reports.
- Re-keying / delegated access: authorized project personnel or external auditors can receive access without publishing the payload.
- Remediation linkage: a fix verification package references the prior finding commitment, vulnerable commit, remediation commit and repeat-test outcome.
- Metadata minimization: high-value customers may choose blinded repository identifiers or fully private anchoring policies to avoid leaking incident timing/activity.

NON-NEGOTIABLE

Prove the finding. Never expose the vulnerability.

13. Lock Bounty: Private Bug-Bounty Verification

Lock can become the verification layer between authorized projects and security researchers. The project can opt into private researcher submissions and fund a TWIX bounty pool. A submitted claim is not accepted simply because a researcher labels it critical; Lock attempts to reproduce it against the authorized target state.

1. Researcher submits affected component/commit, claim, severity rationale and encrypted evidence.
2. Lock validates authorization and rebuilds the relevant source state.

3. The platform executes a private reproduction plan in isolated infrastructure.
4. If reproducible, the project receives a sealed verified finding.
5. The project patches the issue and pushes an authorized remediation state.
6. Lock reruns the exact demonstrated failure condition.
7. If remediation is verified, the bounty workflow can release TWIX according to project policy.

Optional researcher TWIX staking can be used as a confidence/reputation mechanism, but it should not be mandatory for legitimate vulnerability reporting. False positives and uncertain research are part of security work; the system should reward reproducibility and history rather than automatically punish good-faith mistakes.

Project reputation can also become meaningful. A public project profile may safely show coverage state, model freshness, resolved finding count or proof commitments without disclosing active vulnerabilities. Unresolved critical findings should remain private unless the project has explicitly defined another disclosure policy; the product philosophy is remediation, not public shaming.

14. TWIX Economics and Monetization

TWIX is intended to be a functional settlement and utility asset across the ecosystem. Lock is particularly important because it can generate recurring, non-speculative token demand tied to expensive security work.

Utility / revenue source	TWIX role	Economic character
Network gas	Transaction fees for Cosmos/EVM activity	Base protocol demand
Lock authorization	Low-cost authorization/revocation transactions	Consent/control-plane utility
Lock Inspect / Deep Analysis	Pay for source/modeling compute	Usage-based
Lock Sim	Pay for sandbox compute, worker parallelism and simulation depth	Compute-intensive usage
Lock Watch	Continuous monitoring subscription settled in TWIX	Recurring
Lock Shield	Enterprise policy/adapters/coverage	Recurring + service
Lock Rail	Private report custody, re-keying, retention and proof events	Usage/retention
Lock Bounty	Bounty escrow, verified payouts and optional platform fee	Marketplace
TWIX Gateway	Routing/aggregation/execution fees where applicable	Transaction flow
TWIX Exchange	Trading/liquidity/market infrastructure fees	Transaction flow
TWIX Twin	Automation, policy and advanced execution services	Recurring/usage
GameVerse	Game economy, marketplace and developer services	Application economy

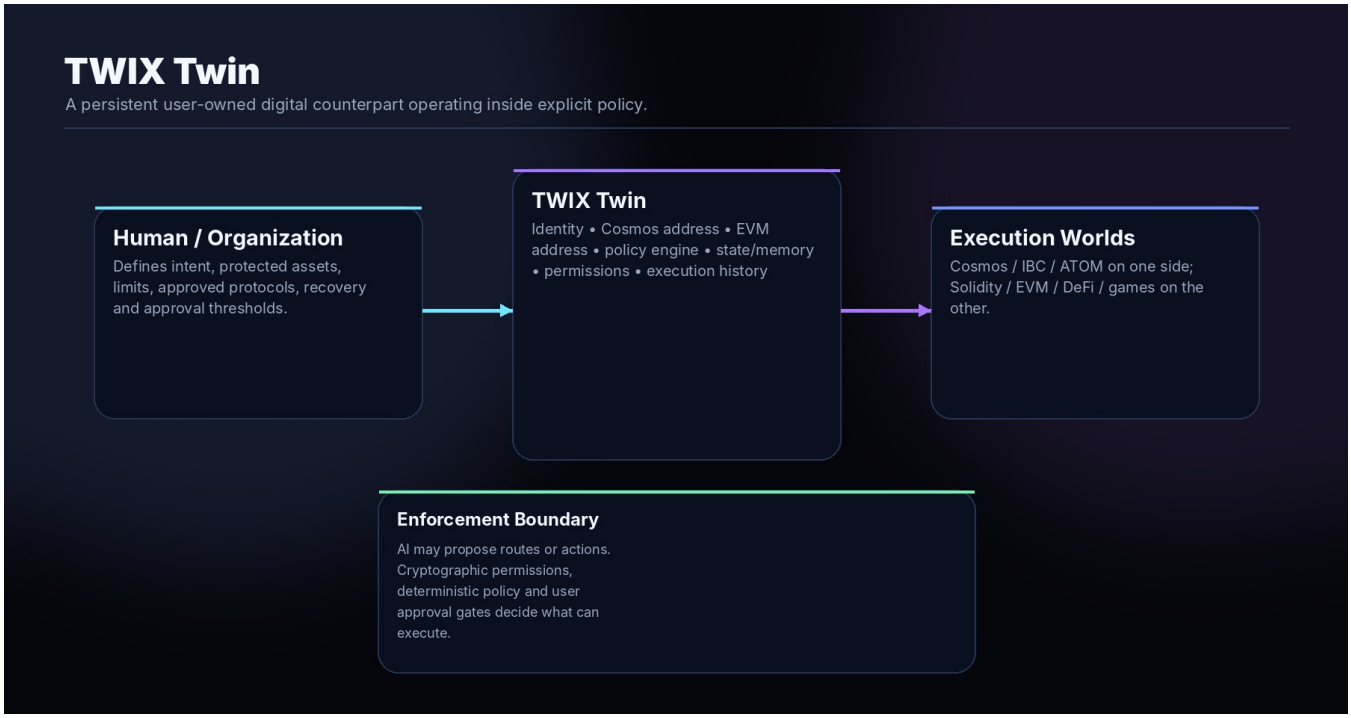
Utility / revenue source	TWIX role	Economic character
Developer APIs	Premium simulation, routing, monitoring or data endpoints	Usage/subscription

Authorization should remain intentionally inexpensive. The customer should pay for actual analysis depth, compute, monitoring and optional advanced services—not for the privilege of proving consent. Pricing should be transparent before a job is submitted and can be expressed as configurable TWIX budgets rather than fixed marketing tiers.

TWIX also creates a natural bounty settlement asset: projects fund pools, validated researchers receive rewards, and the platform can meter verification work in the same currency used by the network. The goal is native utility tied to productive activity rather than artificial burn mechanics.

15. TWIX Twin

TWIX Twin is the proposed autonomous entity layer: a persistent user-owned or organization-owned digital counterpart that can hold identity, Cosmos/EVM addresses, permissions, policy, state and execution history. It is not a chatbot with a private key. The central design requirement is constrained authority.



15.1 Twin types

- Personal Twin — portfolio, recurring payments, staking/routing and approved DeFi execution.
- Business Twin — treasury operations, invoices, supplier/payment rules and operational automation.
- DAO Twin — governance-constrained execution under approved policies.
- Game Twin — persistent autonomous character/entity capable of owning and using assets.
- Marketplace Twin — rules-based buyer/seller/agent within approved markets.
- Developer Twin — infrastructure monitoring, service funding, deployment or operational automation.

- Asset Twin — an agent associated with a real-world or digital asset and its permitted workflows.

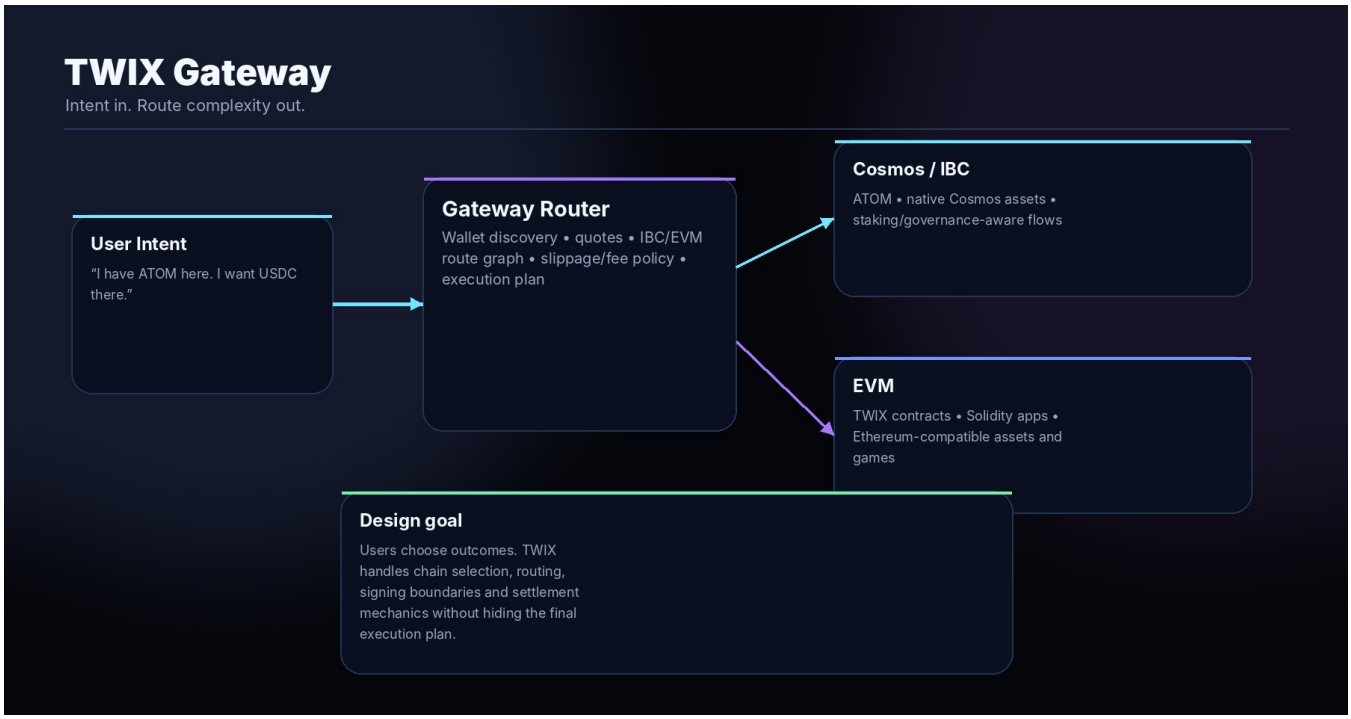
15.2 Policy model

A Twin policy should distinguish actions that are automatically allowed, actions requiring user/organization approval and actions that are permanently forbidden. Examples include daily spending limits, protected assets, approved contracts, bridge destinations, transaction-size thresholds, chain allowlists and recovery controls.

Lock can become a security companion to Twin: model agent policies, test execution boundaries, monitor behavioral drift and attest to approved automation. Gateway becomes Twin's routing/execution layer, while the chain provides settlement and authorization.

16. TWIX Gateway

TWIX Gateway is the proposed consumer and developer access layer that hides unnecessary Cosmos/EVM plumbing while preserving explicit transaction plans. The user chooses an outcome; Gateway discovers available routes, costs and signer requirements across Cosmos, IBC and EVM.



A canonical user story is simple: "I have ATOM here; I want USDC there." Gateway should determine whether the best path is an IBC transfer, a TWIX-native swap, an EVM contract call, a bridge/solver route or a multi-step combination, then present the final route, fee, slippage and approval boundary before execution.

Gateway also becomes an API/SDK product. External wallets, dApps and games should be able to request a route or intent plan without implementing every chain-specific primitive themselves. Over time, Gateway can expose policy-aware routing for Twin and institutional applications.

17. TWIX Exchange, Markets and Trading Tools

17.1 TWIX Exchange

TWIX Exchange is the planned native liquidity/execution layer. Rather than positioning a standalone AMM as the flagship, Exchange serves Gateway, Twin, users and applications. Solidity/OpenZeppelin-compatible contracts can provide familiar EVM development patterns while the broader chain retains Cosmos-native staking/governance and IBC connectivity.

The project may reuse lessons or code from prior exchange builds, but TWIX-native deployment should have TWIX identity, independent contracts, transparent pool economics and clear security boundaries.

17.2 TWIX Markets and Trading Tools

The public toolkit already contains strategy/risk analysis surfaces such as Token Trade Optimiser, TWIX Intercept, Perpetual Hedge Lab and Prediction Signal Lab, plus additional calculators in the unified tools hub. These tools are non-custodial analytical applications. They can evolve into a broader TWIX Markets layer that supplies simulation and decision support to human users and, eventually, policy-limited Twins.

BOUNDARY

Analytical models should explain assumptions and uncertainty. They are not guaranteed-return engines and should not receive unrestricted authority to trade user funds.

18. TWIX GameVerse and Game Infrastructure

Gaming is a major long-horizon flagship application category. TWIX's EVM compatibility, MetaMask support and Cosmos interoperability can support Unity/Unreal-facing SDKs, player-owned assets, in-game marketplaces, guild/treasury systems and tokenized economies without forcing gameplay to become a sequence of manual wallet prompts.

The strongest demonstration would be an actual high-quality persistent game/world that proves the stack: ships, land, items, resources, businesses, guild treasuries or characters can be player-owned assets; TWIX executes the economy; MetaMask or other EVM wallets handle familiar ownership/signing; Cosmos/IBC assets such as ATOM can participate through Gateway; and standards-compatible NFTs can be portable to broader EVM marketplaces — including OpenSea where chain, asset and indexer support permits.

18.1 Developer stack

- TWIX Unity SDK and later Unreal integration.
- Wallet/account onboarding that abstracts chain configuration while preserving signer authority.
- In-game asset contracts and marketplace primitives.
- Gas/transaction UX suitable for high-frequency game interaction.
- Gateway routes for external Cosmos/EVM assets.

- Twin-compatible autonomous characters and game agents under explicit rules.
- Game telemetry/asset proof integrations with Data Attestation where physical-world links make sense.

18.2 Product identity

The TWIX binary-star identity is naturally suited to a game universe: two execution worlds represented as a binary star system, with TWIX as the shared economic and settlement layer. The product should be designed as a real game first and a blockchain demonstration second.

19. Xeteria Wallet and Browser Layer

Xeteria is the wallet/browser brand intended to provide a user-facing access layer across Cosmos and EVM. The integration direction preserves signer boundaries: Cosmos operations remain native Cosmos actions, EVM operations use EIP-1193-compatible signing, and private keys should not be handed to website code or public infrastructure.

The broader Xeteria ecosystem can expose TWIX assets, Gateway routes, Exchange, Twin authorization and application deep links in one interface. Direct wallet integrations remain preferred over mandatory paid wallet-connection middleware.

TWIX's older Meta Account and Native Authz research is valuable here as evidence that wallet-control relationships can be expressed explicitly rather than by exporting keys. Those experiments remain in the research archive and inform future wallet architecture without being presented as the current mainnet account model.

20. Data Attestation, Explorer, Validators and Governance

20.1 Data Attestation

The TWIX Data Attestation application is distinct from consensus validation. Devices or gateways can sign physical-world records, batch commitments and optionally anchor proof to TWIX EVM. This creates a generalized evidence surface for meters, machines, sensors and other signed data without giving those device keys any role in block production.

20.2 Explorer

The Explorer indexes both Cosmos and EVM-facing network activity, including blocks, transactions, contracts, accounts, validators, governance, charts and tokenomics. It is the public visibility layer for the network and should remain separate from protected operational state.

20.3 Validators

Consensus validators secure twix-1. Operator address, bonded status, stake, commission and jail state are public network data. Current validator counts and stake should be queried live rather than frozen into this whitepaper because the set is expected to evolve.

20.4 Governance

Network-sensitive parameter changes should use Cosmos governance. The launch governance surface already tracks proposal state and current parameters separately so proposal text is not misrepresented as executed configuration. This principle should extend to future upgrades, economic policy and major protocol changes.

21. ATOM and Cosmos Ecosystem Value

TWIX is designed to create concrete utility for Cosmos assets and users rather than merely claiming "Cosmos alignment." The strongest value path is to make ATOM a first-class asset in Gateway, Exchange, GameVerse and Twin workflows while using TWIX's EVM surface to expose Solidity-compatible destinations and applications.

- Gateway: simplify movement of ATOM and other Cosmos assets into approved EVM applications and back toward Cosmos routes.
- Exchange: provide native TWIX liquidity venues that can pair TWIX with bridged/IBC-connected Cosmos assets as integrations mature.
- Twin: allow policy-bounded staking, routing and portfolio operations involving ATOM without surrendering unrestricted key authority.
- Gaming: let Cosmos-native economic value participate in TWIX game economies where routing/asset standards support it.
- Developer ecosystem: create another EVM-compatible execution environment that remains rooted in Cosmos governance/staking architecture and can consume IBC-oriented infrastructure.

The earlier Cosmos Hub Meta Account and binding work also demonstrates TWIX's historical focus on connecting wallet ecosystems. The production strategy is broader: use interoperability to create useful flows and applications, not interoperability for its own sake.

22. Developer Platform and Integration Standards

TWIX should behave like a documented platform, not a chain that forces developers to reverse-engineer its configuration. The current site already publishes canonical network manifests and public endpoints. The roadmap expands this into product APIs and SDKs.

Surface	Purpose	Roadmap direction
Cosmos RPC / REST	Blocks, txs, staking, governance, module state	Maintain hardened standard access.
EVM JSON-RPC / WSS	MetaMask, ethers, Foundry, Solidity apps	Maintain standard-compatible EVM developer surface.
Network manifests	MetaMask, Keplr, unified wallet metadata	One source of truth for chain identity.
Lock API	Targets, authorization, jobs, findings, Watch	Authenticated enterprise/developer API.
Gateway SDK/API	Quotes, route plans, execution intents	Embed cross-ecosystem routing in apps/games.

Surface	Purpose	Roadmap direction
Twin SDK	Policy, approvals, intents, agent state	Embed user-owned autonomous entities.
Game SDK	Unity/Unreal wallet/assets/marketplace	Make TWIX game integration developer-friendly.
Attestation API	Signed records and proof status	Physical-data and application evidence integrations.

Private repository access for Lock should use a narrowly scoped GitHub App or equivalent provider integration. Customers should not be asked to paste broad personal access tokens into the frontend. Secret material belongs in protected backend configuration, not downloadable builds or public repositories.

23. Security, Privacy and Governance Principles

TWIX applies its security thesis to its own infrastructure first. The completed 15/15 Cosmos EVM remediation qualification is the baseline evidence behind that claim; TWIX Lock extends the same model-and-reproduce discipline to authorized external software.

Explicit authority: Signer identity, repository control, capability scope and production-response authority should be explicit and verifiable.

Least privilege: Expose only the endpoints, wallet permissions and application capabilities required for the job.

Private-by-design findings: Security intelligence remains encrypted and access-controlled; public commitments never carry exploit payloads.

AI is advisory: Probabilistic systems can propose hypotheses and routes. Deterministic checks and explicit policy gate consequential execution.

Separation of concerns: Consensus keys, application keys, device keys, wallets, report encryption identities and operational secrets are different trust domains.

Reproducibility: Critical claims should reference exact source/environment state and be repeatable.

On-chain governance for network policy: Consensus-sensitive economic/parameter changes should be transparent and executed by the chain's governance model.

No security theater: A "protected" badge should reflect current model/watch state and verifiable coverage, not a permanent marketing certification.

24. Phased Roadmap

The roadmap below is a target implementation sequence. Security engineering, external integrations, audits, regulatory constraints, wallet/registry approvals and real-world usage can change priorities or timing. Later phases should not be marketed as live until they are demonstrably deployed.

Phase	Target window	Primary deliverables
Phase 0 — Foundation	Live / 2026	Mainnet twix-1; EVM chain ID and public JSON-RPC; hardened public edge; 15/15 Cosmos EVM remediation qualification; website v5; validator directory; Explorer; governance; Data Attestation; Trading Tools; canonical manifests; docs; TWIX Lock flagship frontend demo and authorization UX.
Phase 1 — Lock Core	Target Q4 2026	Production Lock control plane; accounts/organizations; GitHub App; Dual-Proof authorization registry; job scheduler; source acquisition; disposable build/simulation workers; project dashboard; private encrypted findings; TWIX metering and billing; internal dogfooding against TWIX itself.
Phase 2 — Lock Deep Analysis + Watch	Target H1 2027	Executable system model; reusable invariant library; deeper cross-module/Cosmos-EVM analysis; deterministic reproduction pipeline; continuous repository/deployment Watch; remediation regression tests; Lock Rail alpha; private bounty beta.
Phase 3 — Shield + Security Marketplace	Target H2 2027	Project-authored Shield policies and narrow defense adapters; enterprise response governance; researcher identities/reputation; TWIX bounty escrow/payout; optional researcher confidence staking; public coverage profiles with zero vulnerability disclosure.

Phase	Target window	Primary deliverables
Phase 4 — Twin + Gateway	Target 2028	TWIX Twin beta; personal/business/DAO policies; approval thresholds; Gateway route graph across Cosmos/IBC/EVM; API/SDK; initial Twin+Gateway automation; Xeteria integration; Exchange used as native liquidity/execution engine.
Phase 5 — GameVerse + Ecosystem Scale	Target 2028+	Unity SDK and game developer stack; flagship persistent game/world; NFT/asset marketplace; Gateway-enabled external asset flows; Twin-powered autonomous game entities; institutional Lock coverage; broader developer and enterprise platform.

24.1 Parallel workstreams

- Keplr and wallet registry integration as upstream approvals progress.
- MetaMask/EIP-1193 onboarding and broader wallet compatibility.
- Validator decentralization and operational hardening.
- DEX/Exchange contracts, liquidity programs and Gateway route integrations.
- Data Attestation maturity, external device/gateway integrations and evidence APIs.
- Xeteria wallet/browser production integration.
- Security audits, internal qualification and public documentation for each production component.

25. Risks, Constraints and Non-Goals

Area	Risk / constraint	Design response
Lock execution	Repositories can be actively malicious	Strong worker isolation, disposable environments, restricted networking and separation from control plane.
AI analysis	Models can hallucinate or overstate severity	Require machine-testable hypotheses and deterministic reproduction before verification.
Continuous monitoring	Coverage can create false confidence	Expose model freshness/coverage and explicitly state what is not monitored.
Shield	Automated defense can cause outages or economic damage	Only project-authored, deterministic, narrow response policies; no direct LLM authority.
Confidentiality	Metadata can reveal security activity even when payloads are encrypted	Minimize public metadata; support blinded/private commitment policies.
Gateway	Bridges/routes introduce external trust and smart-contract dependencies	Route risk scoring, allowlists/policy, transparent path disclosure and Lock coverage where feasible.

Area	Risk / constraint	Design response
Twin	Autonomous agents can amplify mistakes	Explicit permissions, protected assets, limits, approvals, recovery and audit history.
Gaming	Blockchain friction can damage gameplay	Game-first UX, batched/abstracted transactions and optional on-chain depth rather than constant prompts.
Token economics	Utility demand is not token price support	Focus on productive settlement/compute demand; make no price or return claims.
Regulatory / enterprise	Security services, autonomous execution and markets may face jurisdictional requirements	Design modular policies, access controls and legal/compliance review by deployment context.

25.1 Non-goals

- TWIX Lock is not an authorization to test systems whose owners did not consent.
- TWIX Twin is not intended to be an unrestricted AI-controlled hot wallet.
- Gateway is not intended to hide route risk or signer approval from users.
- A Lock coverage badge is not a guarantee that software contains no vulnerabilities.
- TWIX Trading Tools do not promise profit or replace risk management.
- Historical Cosmos Hub research is not current TWIX Mainnet configuration.

26. Conclusion

TWIX begins as sovereign Cosmos + EVM infrastructure with a completed 15/15 hardened-stack remediation qualification, but the product thesis is larger. TWIX Lock turns implementation into an executable security model and continuously challenges it. TWIX Twin turns user intent into policy-bounded autonomous execution. TWIX Gateway turns cross-ecosystem complexity into explicit route plans. Exchange, Markets, GameVerse, Xeteria, Data Attestation, Explorer, validators and governance complete a platform in which the chain's native asset is used for real work rather than decorative utility.

The most important unifying design principle is authority. Repositories prove who may authorize analysis. Wallets prove cryptographic control. AI proposes but does not silently seize authority. Twins operate inside policy. Gateway shows execution paths. Shield acts only under prior consent. Vulnerability intelligence stays private while proofs remain verifiable.

TWIX VISION

One sovereign network. Two execution worlds. Private security intelligence. Policy-bounded autonomous entities. Verifiable execution.

Appendix A. Canonical Network Constants

Key	Value
Network	TWIX Mainnet
Cosmos chain ID	twix-1
EVM chain ID decimal	1415006552
EVM chain ID hex	0x54574958
Native currency	TWIX
Base denom	atwix
Decimals	18
Bech32 account prefix	twix
BIP-44 coin type	60
HD path	m/44'/60'/0'/0/0
Cosmos RPC	https://rpc.twixchain.com
REST	https://api.twixchain.com
EVM JSON-RPC	https://evm.twixchain.com
EVM WebSocket	wss://ws.twixchain.com
Website	https://twixchain.com
Explorer	https://explorer.twixchain.com
Data Attestation	https://validator.twixchain.com

Canonical network metadata should always be read from the current machine-readable manifests on twixchain.com for integration. This appendix is a September 2026 snapshot.

Appendix B. TWIX Lock Authorization Schema

Illustrative shape for the repository authorization artifact. Exact production schema, signing serialization and registry contract/message format should be versioned before mainnet use.

```
{
  "version": "1",
  "service": "twix-lock",
  "repository": "https://github.com/example/protocol",
  "branch": "main",
  "challenge": "tlk_<one-time-random-challenge>",
  "authorized_signers": [
    {"type": "evm", "address": "0x..."}
  ],
  "permissions": ["clone", "build", "analyze", "simulate", "monitor"],
  "expires_at": "<ISO-8601 timestamp>"
}
```

Production authorization should additionally bind the exact Git commit containing the file and the capability scope into the transaction commitment. Revocation and renewal should be separately versioned operations.

Appendix C. Product Status Matrix

Component	Status in this paper	Notes
TWIX Mainnet	LIVE	Independent production network with Cosmos/EVM public interfaces.
Explorer	LIVE	Public indexed network visibility.
Validators / Governance	LIVE	Live chain state; values change and should be queried.
Data Attestation	LIVE / evolving	Separate application; not consensus validation.
Trading Tools	LIVE / analytical	Public non-custodial calculators/labs.
TWIX Lock frontend	DEMONSTRATED	Full product demo and authorization UX; backend control plane not yet represented as production.
TWIX Lock control plane/workers	PLANNED	Production repository acquisition, model, sandbox, simulation and monitoring.
Lock Rail	PLANNED	Confidential security-report custody inspired by Securail principles.
Lock Bounty	PLANNED	Private bounty verification and TWIX escrow/rewards.
TWIX Twin	PLANNED	Policy-bounded autonomous entity layer.
TWIX Gateway	PLANNED	Cosmos/IBC/EVM intent routing.
TWIX Exchange expansion	PLANNED	Native liquidity/execution engine.
TWIX GameVerse	PLANNED	Unity/Unreal-facing gaming platform and flagship world.
Xeteria TWIX integration	INTEGRATION DIRECTION	Wallet/browser layer being aligned with TWIX.
Meta Account / binding	RESEARCH ARCHIVE	Earlier Cosmos Hub experiments; not current mainnet configuration.
FENS ↔ Lock convergence	RESEARCH	Longer-horizon specification/implementation equivalence work.
Hardened Cosmos EVM stack	**LIVE / QUALIFIED**	August 2026 15-point remediation program completed at 15/15 PASS; ongoing regression and review remain required.

Appendix D. Glossary

Term	Definition
Dual-Proof Authorization	Lock authorization requiring both repository control and a matching cryptographic TWIX wallet transaction.
Executable software twin	A reconstructable private model/environment representing an authorized software target and its behavior.
Invariant	A condition expected to remain true across valid system behavior.
Deterministic reproduction	Repeatable execution showing that a claimed failure condition actually occurs against a defined target state.
Lock Rail	Proposed confidential reporting/custody layer for findings, remediation and verification evidence.
Watch	Continuous analysis of source, dependency, deployment or governance changes for behavioral drift.
Shield	Proposed pre-authorized defensive policy layer that acts only on verified conditions through narrow adapters.
TWIX Twin	User/organization-owned autonomous counterpart operating under explicit policy across Cosmos/EVM.
TWIX Gateway	Proposed intent routing layer abstracting Cosmos/IBC/EVM execution paths while disclosing final route/cost.
GameVerse	Proposed TWIX gaming ecosystem, SDK and flagship game/world.
atwix	Base denomination of native TWIX; 18 decimals.
FENS	The FENS programming language and deterministic consequence-oriented research lineage that inspires part of Lock's implementation-to-model philosophy.
15/15 Hardened Qualification	The TWIX post-remediation result showing all fifteen defined Cosmos EVM audit targets passed the qualification criteria. It is a security checkpoint, not a permanent guarantee.

TWIX Chain Whitepaper + Roadmap v1.1.0 — September 2026

This document is a technical/product roadmap and does not constitute investment, legal, tax or security assurance.