

Web3User Permission Lab

Project White Paper, System Mindmap, Technology Stack & Deployment Roadmap

Version: 0.3.0 — Milestone 6.4 Local Validation Draft · **Date:** 23 October 2026

Repository: syedsenterprises/Web3User-Permission-Lab · **Branch:** main

Committed Baseline HEAD: f1bed7f51e6a3ccb024b17a86993faf07bc48349 (origin/main: 763d3b8)

Status: M0–M6.3 Complete · Milestone 6.4 Locally Validated (Working Tree Verified, Unpublished) · M7–M9 Planned

Automated Test Suite: 238 passed (14 test files) · TypeScript check clean (npx tsc --noEmit)

Planned Domain: web3user.tech (DNS unconfigured · Zero-cost / Approval-first policy)

Development-Stage Notice: Web3User Permission Lab is an experimental prototype and has not been independently security-audited. Passing automated test suites is an engineering baseline and not a substitute for formal security verification or third-party audits. The prototype does not custody digital assets, sign blockchain transactions, or submit transactions to live networks. Do not use to protect real funds.

Document Status Taxonomy

- **BUILT / COMPLETE:** Implemented, integrated, and verified in repository history (M0–M6.3).
- **LOCALLY VALIDATED:** Verified in local working tree on top of commit f1bed7f (Milestone 6.4, unpublished).
- **PLANNED:** Architected milestone scheduled for subsequent development phases (M7–M9).
- **OPTION:** Evaluated design candidate not yet configured or integrated.
- **NOT IN SCOPE:** Explicitly excluded from the initial prototype MVP boundary.

1. Executive Summary

Web3User Permission Lab is an off-chain transaction authorization and policy evaluation engine designed to make Web3 permissions explicit, testable, and auditable. Rather than forcing wallet users or autonomous agents to blindly approve opaque transaction payloads, the engine validates proposed blockchain actions against declarative, fine-grained policies before an action can proceed.

The initial implementation targets ERC-20 transfer actions (transfer(address,uint256), selector 0xa9059cbb). The engine enforces strict calldata layout validation (68 bytes), extracts transaction arguments from calldata rather than untrusted user assertions, checks allowlisted token target contracts and recipients, enforces per-action amount caps using lossless 256-bit integer arithmetic, and verifies time-bounded policy validity.

Beyond simple boolean evaluation, Milestones 6.1–6.4 provide an end-to-end cryptographic and transactional authorization lifecycle: (1) **Asymmetric Digital Signatures (Ed25519)** binding transaction parameters and policy constraints to an authoritative issuer key using deterministic length-prefixed canonical serialization; (2) **Persistent Two-Phase Replay Protection** backed by SQLite in Write-Ahead Logging (WAL) mode; (3) **Transactional Monotonic Timekeeping** detecting and rejecting wall-clock rollbacks; (4) **Non-Spinning Contention Management** with bounded retries via `Atomics.wait`; (5) **Safe TTL Arithmetic** eliminating integer overflow risks up to `Number.MAX_SAFE_INTEGER`; and (6) **Process-Crash Recovery** verified through multi-process crash injection tests.

2. Vision, Problem Statement, and Intended Users

The Problem: Interacting with decentralized applications today frequently requires users to sign hexadecimal calldata and broad token allowances without clear insight into what the transaction actually executes. Users face severe security challenges: calldata blindness, unlimited approvals, phishing, address poisoning, time-of-check to time-of-use (TOCTOU) exploits, and absence of persistent replay controls across restarts.

Proposed Vision: Web3User Permission Lab establishes an off-chain verification layer where every sensitive action must satisfy explicit, verifiable rules. Policies are machine-readable, deterministic, and auditable. Authorization decisions are issued as tamper-evident, time-bounded cryptographic capabilities that can be verified and persistently retired upon single use.

Intended Users and Integrators: (1) Smart Contract Wallets & Account Abstraction (ERC-4337) bundlers/paymasters enforcing spending limits; (2) Autonomous Agent & AI Execution Frameworks operating under bounded budgets; (3) DeFi Treasury & Multi-Sig Operations enforcing dual-control policies; and (4) Security Researchers requiring a reference implementation for formal authorization lifecycles.

3. Product Scope: Implemented Reality vs. Future Roadmap

Capability / Component	Status	Implementation Details
ERC-20 Calldata Evaluator	BUILT	Strict 68-byte validation, ABI padding verification, BigInt amount decoding.
Per-Action Spending Caps	BUILT	Exact ceiling enforcement against decoded calldata amounts.
Address Allowlists	BUILT	Case-insensitive token target and recipient verification.
Canonical Binding Digest	BUILT	SHA-256 hash over length-prefixed representation of request and policy.
Ed25519 Artifact Signing (V3)	BUILT	Asymmetric Ed25519 signing (node:crypto) with length-prefixed canonical encoding.
SQLite WAL Persistent Store	BUILT	Two-phase state machine (issued -> consumed), version discriminator column.
Persistent Monotonic Clock	BUILT	Transaction-locked high-water mark with rollback detection.
Contention Retries & Backoff	BUILT	5-attempt / 250ms budget, non-spinning backoff via Atomics.wait.
Safe TTL Arithmetic	BUILT	Subtraction-first safe-integer calculations up to Number.MAX_SAFE_INTEGER.
Process-Crash Recovery	BUILT	Verified WAL journal rollback on unexpected process termination.
Cumulative Multi-Tx Budgets	PLANNED	Multi-transaction aggregate limits (daily, weekly, monthly budgets).
Arbitrary Function Decoders	PLANNED	Support for Uniswap, Aave, and arbitrary complex smart contract calldata.
On-Chain Validator Contracts	PLANNED	Solidity verification contracts validating Ed25519/Secp256k1 authorization proofs.
Production Wallet RPC Proxy	PLANNED	Transparent EIP-1193 / JSON-RPC middleware proxy intercepting transactions.
Independent Security Audit	PLANNED	Formal external review by reputable smart contract security auditing firms.

4. Core Policy Evaluator Engine

The core evaluator (src/policy.ts) is a zero-dependency, deterministic TypeScript library. It receives a policy configuration, a proposed transaction action, and a trusted timestamp, returning one of three explicit decisions:

- **ALLOW:** The action strictly complies with all allowlists, boundaries, and timestamps.
- **DENY:** An explicit policy restriction was violated (e.g., recipient not allowlisted, amount exceeded, policy revoked).
- **UNKNOWN:** Input data was missing, malformed, unsupported, or inconsistent.

Evaluation Sequence: (1) Input structural validation; (2) Trusted timestamp validation (must be non-negative safe integer supplied by caller); (3) Policy validity and revocation check; (4) Function selector dispatch (requires 0xa9059cbb); (5) Calldata length and ABI padding check (requires exactly 68 bytes and 12 leading zero bytes for addresses); (6) Argument decoding (extracts recipient and BigInt amount); (7) Compatibility field reconciliation; and (8) Policy rule checks (target, recipient, amount ceiling).

5. Threat Model and Security Architecture

Addressed Threats: Calldata tampering, off-by-one expiry attacks, floating-point precision loss, signature forgery, double-consumption replay, clock rollbacks, and multi-process lock contention.

Critical Cryptographic Distinction: Off-Chain vs. On-Chain Signing: It is vital to distinguish between authorization artifact signing and blockchain transaction signing:

- **Authorization Artifact Signing (Off-Chain):** Web3User uses Ed25519 keys (node:crypto) to attest that an off-chain policy evaluation authority reviewed and permitted a specific transaction proposal.
- **Ethereum Transaction Signing (On-Chain/Wallet):** The wallet or smart account retains custody of its Secp256k1 private keys. Web3User does NOT hold user private keys, sign raw EVM transactions, or broadcast transactions.
- **Enforcement Layer Responsibility:** An integrating wallet or node must verify the signed authorization artifact before signing the blockchain transaction.

6. Versioned Authorization Artifacts & Cryptographic Binding

To eliminate time-of-check to time-of-use (TOCTOU) substitution, evaluation decisions are bound into tamper-evident artifacts. Milestone 6.4 implements Version 3 signed authorization artifacts:

Artifact Schema V3:

```
version: 3
authorizationId: string (non-empty, trimmed)
keyId: string (registered issuer key identifier)
issuedAtSeconds: number (safe integer timestamp)
expiresAtSeconds: number (safe integer timestamp > issuedAtSeconds)
policyId: string (authoritative policy identifier)
policyRevision: number (positive safe integer revision >= 1)
bindingDigest: string (sha256:<64 hex characters>)
signature: string (128-char hex Ed25519 digital signature)
```

Length-Prefixed Canonical Serialization Header: web3user-signed-authorization-v3\n
Field Encoding: name:length:value\n (eliminates delimiter collision and ambiguity)

7. Persistent Replay Protection, SQLite WAL Storage, and Concurrency Governance

Authorization state is maintained in SQLite with Write-Ahead Logging (PRAGMA journal_mode = WAL;) and immediate transaction isolation (BEGIN IMMEDIATE;).

Additive Schema Architecture:

- consumed_authorization_ids: Tracks authorization_id (PK), binding_digest, state ('issued' | 'consumed'), consumed_at, and artifact_version (cross-version discriminator).
- issuer_keys: Tracks key governance (active, retired, revoked) and validity windows (validFromSeconds <= retiredAtSeconds <= revokedAtSeconds).
- policies & policy_revisions: Monotonically increasing revision numbers preventing policy rollback.
- system_clock_state: High-water mark tracking for monotonic time enforcement.

Two-Phase State Invariant: Authorizations transition atomically from issued to consumed via single-row UPDATE matching authorization ID, binding digest, version, and issued state. Duplicate attempts return DENY.

8. Transactional Monotonic Clock, Bounded Retries, and Process-Crash Recovery

Monotonic Clock High-Water State: Within immediate write transactions, the store validates that trustedNowSeconds >= highWaterSeconds. Regressive timestamps abort with an error, preventing clock-skew attacks.

Bounded Retries & Non-Spinning Backoff: SQLite lock contention (SQLITE_BUSY) is handled through a deterministic 5-attempt / 250ms budget with randomized exponential backoff. Synchronous delays use Atomics.wait on a SharedArrayBuffer (sleepSync), yielding the CPU thread without spinning.

Process-Crash Recovery vs. Physical Power-Loss Durability: Tests verify that abrupt worker termination (SIGKILL or process.exit(1)) midway through uncommitted write transactions triggers automatic SQLite WAL recovery upon reopen, rolling back all uncommitted writes and clock updates cleanly. *Important qualification: This verifies process-crash recovery; physical power-loss durability depends on operating system page caches, disk controller caches, and filesystem fsync barriers outside this library's boundary.*

9. Safe TTL Arithmetic and Integer Overflow Hardening

In JavaScript runtimes, integer arithmetic exceeding Number.MAX_SAFE_INTEGER ($2^{53} - 1 = 9,007,199,254,740,991$) causes precision loss and broken timestamp comparisons.

Subtraction-First Expiration Calculation: To prevent overflow when adding large TTLs to current time:

1. Determine ceiling deadline maxAllowedExpiresAt as min of policy expiry, key expiry, and Number.MAX_SAFE_INTEGER.
2. If maxAllowedExpiresAt <= effectiveNowSeconds, validity interval is empty and issuance is denied immediately.
3. Compute remaining safe window: remainingValidSeconds = maxAllowedExpiresAt - effectiveNowSeconds.
4. If ttlSeconds >= remainingValidSeconds, clamp expiration to maxAllowedExpiresAt.
5. Otherwise, effectiveExpiresAt = effectiveNowSeconds + ttlSeconds, which is mathematically guaranteed not to overflow.
6. Strict validation asserts Number.isSafeInteger(effectiveExpiresAt) and effectiveExpiresAt > effectiveNowSeconds.

10. System Mindmap (Text Form)

The complete hierarchical system mindmap is published as a companion landscape document (assets/web3user-project-mindmap-v0.3.0.pdf). Its high-level branches comprise: (1) Product Scope & Foundation; (2) Core Evaluation Engine; (3) Cryptographic Binding & Signing; (4) Persistent Storage & Governance; (5) Concurrency, Contention & Crash Resilience; (6) Static Website Preview; (7) Tools & Testing Harness; and (8) Development Roadmap.

11. Technology Stack: Tools, Libraries, and Roles

Tool / Technology	Role in Project	Status / Notes
TypeScript 5.7+	Core implementation of evaluator, cryptography, and storage.	IN USE (strict type checking)
Node.js (v26.11+)	Runtime platform for local execution and test execution.	IN USE (Node runtime)
node:sqlite	Embedded SQLite binding for WAL persistence and locking.	IN USE (storage engine)
node:crypto	Ed25519 digital signing, key generation, and SHA-256 digests.	IN USE (cryptography)
Vitest 5.0+	Automated test runner for unit, concurrency, and crash suites.	IN USE (238 passing tests)
tsc (TypeScript)	Static compilation and type analysis (npx tsc --noEmit).	IN USE (clean type check)
npm + lockfile	Deterministic dependency management and security audits.	IN USE (clean audit)
Git / GitHub	Version control repository (origin/main, private repository).	IN USE (HEAD: f1bed7f)
Static Site Host	Cloudflare Pages or GitHub Pages for static preview hosting.	OPTION (candidate preview)
Custom Domain	web3user.tech project identity.	PLANNED (DNS unconfigured)

12. Static Website Preview and Domain Deployment Strategy

The static website preview bundle (website-preview/) is self-contained and ready for offline review. It comprises a responsive dark-themed landing page (index.html) and downloadable distribution assets (assets/).

Zero-Cost and Approval-First Policy: In accordance with project governance rules, the custom domain web3user.tech remains unconnected and DNS settings are unmodified. The site can be reviewed locally or deployed to a free static provider preview URL if authorized. Public domain connection is deferred until a formal public preview release is approved.

13. Development Roadmap (M0–M6.3 Complete, M6.4 Locally Validated, M7–M9 Planned)

- **M0–M3: Foundation, Evaluator & Baseline Tests (Complete):** Evaluator, parser, amount checks, and initial 34 tests.
- **M4–M5: Transaction Binding & Persistence (Complete):** Canonical encoding, binding digests, SQLite two-phase store.
- **M6.1–M6.2: Asymmetric Signatures & Governance (Complete):** Ed25519 signing, key registry, monotonic revisions.
- **M6.3: Concurrency Hardening & Monotonic Clock (Complete):** Clock rollback prevention, non-spinning backoff (209 tests).
- **M6.4: Post-Commit Security & Crash-Recovery Validation (Locally Validated · Unpublished):** Safe TTL arithmetic, multi-process race barriers, process crash recovery verification (238 tests). Locally validated in working tree on top of commit f1bed7f51e6a3ccb024b17a86993faf07bc48349.
- **M7: Cumulative Spending Budgets (Planned):** Rolling multi-transaction budgets across time windows.
- **M8: Adversarial Corpus & Fuzzing (Planned):** Property-based fuzzing and malformed payload corpora.
- **M9: Independent Security Review & Public Demo (Planned):** Third-party audit and controlled testnet demo.

14. Governance, Privacy, Secret Hygiene, and Release Policy

1. No token launch, custody of digital assets, or real-money transactions in prototype.
2. Zero secret commitments: private keys and seeds are never committed to source control.
3. Truth in advertising: internal test passage is never claimed as a third-party security audit.
4. All releases cite reproducible Git commit hashes, test metrics, and working-tree status.
5. Fail closed: ambiguous, malformed, or error conditions return DENY or UNKNOWN by default.

15. Key Technical Risks and Architectural Decisions

- **Storage Durability:** Process-crash recovery is proven via SQLite WAL rollback; physical power-loss durability requires evaluation of host filesystem fsync guarantees.
- **Clock Source:** Backward clock shifts are rejected; forward clock drift can prematurely expire authorizations. Production environments require synchronized chrony/NTP daemons.
- **Bearer Capabilities:** Unconsumed signed artifacts are bearer capabilities requiring TLS transport protection.

16. Summary & Next Steps

Web3User Permission Lab represents the current locally validated security architecture. With 238 passing tests, clean TypeScript compilation, and verified process crash recovery, the architecture provides a solid, reproducible foundation for Milestone 7 cumulative budget modeling.

Appendix A — Reproducibility Commands & Test Harness

```
cd C:\Users\syeds\Projects\Web3User

# Execute automated test suite (238 passed across 14 files)
npm test

# Static TypeScript type check
npx tsc --noEmit

# Execute isolated persistent replay demo
npm run demo:persistent-replay -- --verify

# Inspect working tree status and baseline commit
git status
git log -1 --oneline
```

Appendix B — Known Repository Snapshot & Invariants

- **Committed Baseline HEAD:** f1bed7f51e6a3ccb024b17a86993faf07bc48349
- **Tracked Source Files:** src/policy.ts, src/authorization-binding.ts, src/authorization-crypto.ts, src/authorization-lifecycle.ts, src/authorization-store.ts, src/clock.ts, src/key-registry.ts, src/key-store.ts, src/policy-store.ts, src/sqlite-transaction.ts, src/transaction-envelope.ts.
- **Test Suite Files (14 files, 238 tests):** test/authorization-boundaries.test.ts, test/authorization-lifecycle.test.ts, test/calldata-evaluator.test.ts, test/concurrency-governance.test.ts, test/crypto-signatures.test.ts, test/edge-cases.test.ts, test/fail-closed.test.ts, test/persistence-recovery.test.ts, test/policy-evaluator.test.ts, test/policy-store.test.ts, test/signed-authorization-v3.test.ts, test/sqlite-concurrency.test.ts, test/sqlite-store.test.ts, test/sqlite-transaction.test.ts.