Web3User Permission Lab

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

Version 0.2.0 • Development Draft • 22 October 2026

GitHub: syedsenterprises/Web3User-Permission-Lab • Branch: main • Current commit: e8db545

Planned public domain: web3user.tech

M0–M3 complete

M4–M9 planned

Document map

• Document status and how to read this paper

• 1. Executive summary

• 2. Vision, problem, and intended users

• 3. Product scope: what it is and is not

• 4. How the evaluator works

• 5. Threat model and security principles

• 6. System mindmap (text form)

• 7. Technology stack: current tools and intended jobs

• 8. Website and domain deployment plan

• 9. Website information architecture

• 10. Roadmap with deliverables and acceptance criteria

• 11. Governance, privacy, and release policy

• 12. Risks and open decisions

• 13. Summary

• Appendix A — Reproducibility commands

• Appendix B — Known repository snapshot

Status labels distinguish built features from planned work and unselected options.

Document status and how to read this paper

This is a development-stage technical and project white paper. It describes what exists today, what the proposed product could become, which tools are planned for which jobs, how the website could be hosted, and what must be built before real-world use. It is not a claim that the full product already exists.

Status labels: BUILT = supported by the project outputs shared so far; PLANNED = a future milestone; OPTION = a candidate technology that has not been selected or configured; NOT IN SCOPE = deliberately excluded from the initial MVP.

Current reality: the repository contains a TypeScript policy evaluator for one ERC-20 transfer call type and 34 passing tests. It is not yet a wallet, browser extension, smart contract, transaction blocker, custody product, or production authorization service.

1. Executive summary

Web3User Permission Lab is a proposed user-permission and transaction-policy safety layer for Web3. The long-term idea is to help a user or integrating application define understandable rules for blockchain actions—such as which token contract may be called, who may receive a transfer, how much can be sent in one action, and when a permission is valid—then evaluate a proposed transaction against those rules before signing or execution.

The first prototype is intentionally narrow. It evaluates ERC-20 transfer(address,uint256) calldata using selector 0xa9059cbb and returns ALLOW, DENY, or UNKNOWN. It validates the expected 68-byte calldata layout, decodes recipient and amount, checks allowlists, amount limits, policy validity, revocation, and malformed input.

The product should grow through evidence-based milestones: exact transaction binding; fail-closed enforcement integration; chain-aware transaction context; persistent budget and replay protections; adversarial testing; usability; and independent security review. A decision engine is not a security guarantee until a trusted enforcement boundary ensures the evaluated transaction is the transaction actually signed or executed.

The domain web3user.tech is planned as the public project identity. Given the current development stage and the previously stated ■0-first constraint, the recommended approach is to keep the domain unconfigured until the project is ready for a controlled preview. Prepare the website and white paper now, host a preview on a free provider URL if useful, and connect the custom domain only after the owner explicitly approves DNS changes and public disclosure.

2. Vision, problem, and intended users

Problem

Wallet users often face complex transaction details: contract addresses, calldata, token amounts, approvals, chain identity, and permissions. Users may not easily see whether a transaction fits their own intentions. Applications may also need a clear, testable way to express allowed and disallowed actions.

Proposed vision

Make transaction permissions more explicit, understandable, testable, and enforceable—without pretending that a simple rules engine can guarantee safety on its own.

Potential users and integrators (hypotheses, not validated market findings)

- Individual wallet users who want understandable rules around selected transaction types.
- Wallet, dApp, or account-abstraction teams that may want a policy evaluation interface.
- Developers building transaction previews, permission dashboards, or controlled demo environments.
- Security researchers and educators who need a small reference implementation for policy evaluation.

Value proposition to test

A narrowly scoped, inspectable policy engine that explains why an action is allowed, denied, or uncertain, with explicit limitations and tests. Market demand, user willingness to adopt, and integration feasibility still require interviews and prototype testing.

3. Product scope: what it is and is not

Current prototype

- A TypeScript policy evaluator for a single supported ERC-20 transfer call format.
- A demo policy model with allowed target, allowed recipient, selector, per-action maximum, validity window, and revocation state.
- Conservative outcomes: ALLOW, DENY, UNKNOWN.
- 34 automated tests reported passing; TypeScript check passed; npm audit reported zero vulnerabilities at the time checked.

Not built yet

- No browser extension or wallet interface.
- No signing, transaction submission, RPC-based transaction simulation, or transaction blocking.
- No smart contract deployed on a blockchain.

- No chain-aware binding for chain ID, sender, native value, nonce, deadline, or token identity.
- No cumulative budget accounting, replay protection, or concurrent-spend controls.
- No production policy service, user authentication, database, monitoring, or independent audit.

Initial exclusions

The first MVP should not launch a token, custody user assets, process real-money transactions, deploy an on-chain contract, or require paid services. These remain out of scope unless explicitly reconsidered in a future project decision.

4. How the evaluator works

Inputs

- Policy: target contract address, allowed recipient, allowed selector, maximum amount, validFrom, expiresAt, and revoked status.
- Proposed action: transaction target, calldata, and a compatibility amount field.
- Evaluation context: trusted nowSeconds supplied by the caller.

Evaluation sequence

1) Validate the input shapes. 2) Validate the policy and trusted time. 3) Check the supported selector. 4) Require the exact expected calldata length. 5) Validate ABI address padding. 6) Decode recipient and amount from calldata. 7) Verify the compatibility amount matches the decoded amount. 8) Check target, recipient, selector, per-action limit, policy time window, and revocation. 9) Return ALLOW, DENY, or UNKNOWN.

Decision semantics

Result	Meaning	Integration rule
ALLOW	The supported action passed the implemented checks.	Not execution approval by itself; bind it to the exact transaction.
DENY	A policy restriction is violated.	Must not proceed.
UNKNOWN	Data is missing, malformed, inconsistent, or unsupported.	Fail closed; do not interpret as approval.

The current amount limit is per action, not cumulative. A series of individually allowed transfers could exceed an intended aggregate budget. The evaluator also does not verify that an address is a trustworthy token contract.

5. Threat model and security principles

Current checks

- Reject unsupported selectors and malformed or trailing calldata.
- Decode recipient and amount from the calldata, not just a user-supplied claim.
- Enforce target and recipient allowlists, amount maximum, policy time window, and revocation.
- Return UNKNOWN for missing, malformed, or inconsistent data.

Key unresolved threats

- Time-of-check/time-of-use mutation: target or calldata changes after evaluation.
- Unmodeled transaction fields: chain ID, sender, native value, nonce, deadline, token identity, gas-related fields where relevant, and transaction type.
- Compromised or mutable policy source, user session, signer, wallet, browser, or enforcement component.
- Replay, duplicate submissions, concurrent actions, and aggregate spending beyond limits.
- Malicious, upgradeable, or deceptive contracts and incorrect token metadata.

Design principles

- Fail closed: DENY and UNKNOWN never mean approval.
- Bind decision to transaction: evaluate a canonical representation and ensure the same representation is consumed by the signing/enforcement boundary.
- Least privilege: policy grants the smallest useful scope and duration.
- Explainability: show which rule failed, without leaking sensitive data.
- Auditable change: version policies and record testable decisions where privacy allows.
- Defense in depth: the policy engine is one layer, not a substitute for wallet security, simulation, contract review, or user judgment.

6. System mindmap (text form)

The detailed mindmap is also provided as a separate landscape PDF. The branches below show the whole project at a glance.

```
WEB3USER PERMISSION LAB
■■ Product: explicit permissions for Web3 actions
■ ■■ Users: wallet users, dApps, wallet/account teams (hypotheses)
■ ■■ Outcomes: ALLOW / DENY / UNKNOWN
■ ■■ UX later: rule builder, explanation, policy history
■■ Core engine [BUILT prototype]
■ ■■ TypeScript: src/policy.ts
■ ■■ ERC-20 transfer(address,uint256), selector 0xa9059cbb
■ ■■ 68-byte calldata, ABI padding validation, recipient/amount decode
■ ■■ target + recipient allowlists, maxAmount, time window, revocation
■ ■■ tests: 34 passing; TypeScript check passes
■■ Security [PARTIAL]
■ ■■ malformed inputs -> UNKNOWN
■ ■■ per-action cap (not cumulative)
■ ■■ TODO: immutable transaction binding
■ ■■ TODO: fail-closed enforcement boundary
■ ■■ TODO: chain/sender/value/nonce/deadline context
■ ■■ TODO: replay, concurrency, audit review
■■ Website: web3user.tech [PLANNED]
■ ■■ Home / mission / how it works
■ ■■ White paper + version history
■ ■■ Roadmap + achievement ledger
■ ■■ Security model + limitations
■ ■■ Demo playground only with simulated data
■ ■■ Contact / contribution / disclosure process later
■■ Hosting and delivery [OPTIONS]
■ ■■ First preview: free static hosting provider URL
■ ■■ Candidate: Cloudflare Pages or GitHub Pages
■ ■■ Domain/DNS: defer until explicit approval
■ ■■ No API/database needed for static white-paper site
■■ Tools
■ ■■ TypeScript + Node.js: implementation/runtime tooling
■ ■■ Vitest: automated unit/regression tests
■ ■■ TypeScript compiler: static type checking
■ ■■ npm: dependencies and audit
■ ■■ Git + GitHub: version history and source repository
■ ■■ Markdown: docs / README / security model
■ ■■ Later candidates: wallet test framework, ABI library, CI, static analysis
■■ Roadmap
■ ■■ M0-M3 COMPLETE: foundation, evaluator, tests, docs, private GitHub publish
■ ■■ M4: transaction binding
■ ■■ M5: enforcement interface
■ ■■ M6: chain-aware context
■ ■■ M7: budgets/replay/concurrency
■ ■■ M8: adversarial/property tests
■ ■■ M9: independent review + demo readiness
■■ Constraints and governance
■■ ■0-first; no paid service dependency for initial MVP
■■ no token launch, custody, real-money transactions, or on-chain deployment initially
■■ private repo unless owner changes visibility
■■ never claim audit or production readiness without evidence
```

7. Technology stack: current tools and intended jobs

Tool / technology	Purpose	Status / notes
TypeScript	Main language for policy logic; types reduce some classes of integration mistakes but do not prove security.	In use.
Node.js	Runs the local development toolchain and test scripts.	In use; version recorded as v26.11.1 in project context.
npm + package-lock.json	Install and lock project dependencies; npm audit checks known advisory database results.	In use; audit result is time-specific.
Vitest	Automated unit/regression tests for allowed, denied, boundary, and malformed cases.	In use; 34 tests reported passing.
TypeScript compiler (tsc)	Static type-checking using npx tsc --noEmit.	In use; no errors reported.
Git	Local history, commits, branch and change review.	In use; first commit e8db545.
GitHub	Remote source repository and collaboration/version history.	In use; private repository per current instruction.
Markdown	Human-readable README and security/design documentation.	In use.
Static site host (Cloudflare Pages or GitHub Pages)	Serve public informational website and white paper without a backend.	Candidate only; not selected/configured.
Custom domain web3user.tech	Project identity and future public URL.	Planned; do not change DNS without approval.
Wallet/chain integration libraries	Could later parse chain-specific transactions and connect to a test wallet/network.	Not selected; wait until transaction-binding requirements are designed.
CI workflow	Automatically run tests and type checks on changes.	Future option; not configured in current snapshot.
Database/API	Could later store users, policy versions, revocation and budget accounting; adds security/operations burden.	Not needed for static website; not built.
Monitoring/logging	Observe website uptime and, later, backend errors/security events.	Future option; avoid collecting sensitive transaction/user data unnecessarily.

Tooling rule: only label a tool BUILT or IN USE if it is evidenced in the current project. Cloudflare Pages, GitHub Pages, wallet libraries, CI, databases, and monitoring are options—not existing integrations.

8. Website and domain deployment plan

Recommended approach

Prepare a static informational website now, but do not point web3user.tech at it until you explicitly decide to activate DNS. A development-stage white paper can be useful publicly, but publishing it on the custom domain is a separate decision from writing it. Because the project has a ■0-first constraint and previously excluded domain/DNS changes during the initial phase, preserve the domain settings for now.

Phase A — prepare locally

- Create a static site with Home, White Paper, Roadmap, Security Model, and Status pages.
- Label the document “Development Draft v0.2.0” and state what is built versus planned.
- Use a simulated demo only; do not connect a wallet or ask for seed phrases/private keys.
- Keep the GitHub repository private if that remains the owner's choice; public website hosting and public source code are separate decisions.

Phase B — free preview (optional)

Deploy the static site to a provider-generated preview URL, if the owner approves. Cloudflare Pages and GitHub Pages are candidate static hosts; no provider has been configured in this project yet. A static white-paper site does not need a database, API, smart contract, or paid backend.

Phase C — custom domain (later, explicit approval)

- Confirm domain registrar access and current DNS records before changing anything.
- Choose the static host; follow its current custom-domain instructions and verify domain ownership.
- Add only the required DNS records, preserving existing email and other service records.
- Enable HTTPS, verify redirects and canonical URL, test mobile layout, and check that only intended documents are public.
- Publish a release-tagged white paper and a clear changelog; preserve previous versions.

Should the domain stay unused until the project is complete?

Do not leave all preparation until the project is complete: build and review the site and draft now. But keep web3user.tech unconnected until a public-preview milestone is approved. The domain can be connected while the product is still in development if the site clearly says “experimental / development stage” and avoids promises of security or production functionality. The conservative recommendation is to use a free preview URL first and activate the custom domain only after reviewing the public content.

9. Website information architecture

Page	Purpose	Content
Home	Explain the idea in 30 seconds.	Mission, problem, current status, CTA to read white paper.
White Paper	One canonical explanation of the project.	PDF + accessible HTML/Markdown version, version/date, changelog.
How It Works	Explain evaluator in plain language.	Policy → parse calldata → evaluate → decision; clear limitations.
Roadmap	Show evidence-based progress.	M0–M3 complete; M4–M9 planned; dated release notes.
Security Model	Build trust through transparent boundaries.	Threat model, supported selector, limitations, responsible disclosure contact when ready.
Demo Playground	Teach policy outcomes safely.	Local/simulated sample data only; no real wallet connection in initial phase.
Documentation	Developer onboarding.	Setup, tests, policy schema, architecture, contributing guidelines.
Status / Updates	Communicate changes.	Latest version, test status, known limitations, last updated.
Contact / Disclosure	Provide a channel for feedback and security reports.	Add only when a reliable mailbox or process exists.

10. Roadmap with deliverables and acceptance criteria

Milestone	Work packages	Acceptance criteria
M0 — Foundation (complete)	Project files, package scripts, .gitignore, local Git.	Repository reproducible; first commit exists.
M1 — Evaluator (complete)	ERC-20 transfer parser and demo policy checks.	Correct decisions for supported action; unknown on malformed/unsupported data.
M2 — Test hardening (complete)	Boundaries, malformed inputs, mismatch tests.	34 tests pass; type check passes; audit checked.
M3 — Docs and private remote (complete)	Security model, decisions, remote commit.	Commit e8db545 available on origin/main; repo kept private.
M4 — Exact transaction binding	Canonical transaction object; avoid mutation; decision token/hash or equivalent binding.	Mutation tests prove any relevant field change invalidates the decision; no unbound ALLOW consumed.
M5 — Enforcement adapter	A narrow interface between evaluator and signing/sending path, initially test-only.	DENY/UNKNOWN cannot progress; ALLOW applies only to same bound transaction; tests cover failures.
M6 — Chain-aware context	Chain ID, sender, native value, nonce/deadline, token identity; define supported networks.	Unsupported network/context fails closed; test fixtures document semantics.
M7 — Budgets and replay	Persistent budget accounting, idempotency, concurrency and reset semantics.	Atomic tests show parallel actions cannot exceed configured budgets; replay tests pass.
M8 — Adversarial testing	Property/fuzz tests, malformed ABI corpus, mutation tests, dependency checks.	Invariants documented; test coverage and unresolved risks reviewed.
M9 — Public demo readiness	Independent review, docs/site, disclosures, privacy review, release checklist.	No unresolved critical/high findings; no unsupported security claims; explicit owner approval to publish.

11. Governance, privacy, and release policy

- No token launch, custody, real-money transaction, or on-chain deployment in the initial MVP.
- Do not request or store seed phrases, private keys, wallet passwords, or unnecessary personal data.
- Keep the source repository private unless the owner explicitly changes it; website publication does not require public source code.
- White paper must separate implemented capabilities, planned work, assumptions, and unverified market hypotheses.
- Use semantic versioning or a clear version/date convention; keep a changelog and preserve prior white-paper releases.
- Any security claim must be supported by reproducible tests, documented review, or an independent assessment. Never describe the prototype as audited unless an actual audit has occurred.
- Before public release, review domain ownership, DNS, privacy, contact method, disclaimers, dependency health, and accidental secret exposure.

12. Risks and open decisions

- Product scope: policy evaluation library, developer API, wallet companion, or eventual wallet integration? Decide only after user interviews and technical proof.
- Target integration: which wallet, dApp, or transaction pathway could enforce a decision without a bypass?
- Domain activation: who controls web3user.tech registrar and DNS, and when is public activation approved?
- Hosting: choose a static host after reviewing account access, free-tier limits, build workflow, and custom-domain steps.
- Policy authority and storage: who can create/revoke policies, and how are changes authenticated?
- Chain scope: one EVM chain for the first integration, or broader EVM support later?
- Audit budget and independent reviewers: what must be completed before any real-value use?
- Threat model and user testing: which user problem is validated, and how will incorrect ALLOW/DENY outcomes be measured?

13. Summary

Web3User Permission Lab has a real, testable starting point: a narrow policy evaluator, 34 passing tests, documented boundaries, and a versioned commit in a private GitHub repository. The next goal is not to add marketing claims or deploy a token. It is to prove that authorization is bound to the exact transaction and that enforcement fails closed.

For the website, prepare content and a static preview first. Keep web3user.tech DNS unchanged until the owner approves public activation. A static site is sufficient for a white paper and roadmap; the evaluator and future enforcement system can remain separate until their architecture is proven.

Appendix A — Reproducibility commands

```
Set-Location "$HOME\Projects\Web3User"
npm test
npx tsc --noEmit
npm audit
git status
git log -1 --oneline
```

Recorded baseline from the development session: 34 tests passed; TypeScript check returned no errors; npm audit reported zero vulnerabilities at that time; commit e8db545. Re-run checks before each release.

Appendix B — Known repository snapshot

Repository: https://github.com/syedsenterprises/Web3User-Permission-Lab (private according to the user's current instruction).

Known files: .gitignore, README.md, package.json, package-lock.json, tsconfig.json, src/policy.ts, test/policy.test.ts, docs/DECISIONS.md, docs/SECURITY_MODEL.md.

This paper is a project planning document and does not itself change DNS, deploy the site, or publish the repository.