A3E9 TINL — Transaction Intent Normalization Layer

One product line. Two surfaces: Wallet Security, and Price Impact.

Author: Dennis DZ Zweigle, Founder & Chief Architect, A3E9
Date: August 2026
Document Type: Product-line whitepaper
Audience: Wallet / retail-security evaluators, Chrome and Android tracks (separate)

Document scope

This paper is about one product line: TINL. It has two user-facing surfaces. Wallet Security is the product people connect a wallet to. Price Impact is a TINL surface, not a fourth A3E9 product line — counting it as one would imply an independent product with independent evidence behind it, and there is none.

Three product lines, four surfaces. The other two lines are HSM Normalization / Virtual PKCS#11 and A3E9 II Institutional Trust Hub. Neither is described further here. TINL does not use the HSM process attestation library, and a green HSM evaluator tile is not TINL evidence.

> *Guarantees are per decision, not one blanket promise. BLOCK is enforced. WARN and NORMALIZE are advisory in the shipping browser extension.*

1. The problem TINL actually addresses

Wallets ask people to authorize machine-encoded state transitions. Unlimited ERC-20 approvals, malicious EIP-2612 permits, and SPL delegates are signed because the screen did not say what the bytes do. Simulation is evadable. TINL decodes the payload against a registry, evaluates it against a policy profile, and returns ALLOW, WARN, BLOCK, or NORMALIZE before the wallet’s signing prompt.

The same decode and policy engine (a3e9-tinl/core, Rust) compiles to two transports. Mode A is embedded WebAssembly — no network, no server to trust. Mode B is a REST service running that same core, with a durable HMAC-chained audit log written before every response. Clients pick a transport per request. institutional_strict fails closed if the service is unreachable; the other shipped profiles fall back to the last-known Mode A engine.

2. Surface: Wallet Security

This is the surface with a Chrome evaluator track and a separate Android in-wallet track. They are not the same demo. Do not treat a pass on one as a pass on the other.

Path	Chains	What it can enforce	What it cannot
Browser extension (MV3)	*EVM and Solana*	*BLOCK: the request is refused before the wallet is invoked. The real provider is held where the page cannot reach it.*	*WARN and NORMALIZE consent travel over a page-visible channel a hostile page can forge. The wallet confirmation screen is the backstop. Reproduced in extension/e2e.*
MetaMask Snap	*EVM*	*Insight only. MetaMask calls it; there is no provider race.*	*A Snap cannot reject or rewrite a transaction. It is not the enforcement path.*
JS interceptor library	*EVM*	*When the wallet app owns the hook, all four decisions can be enforced.*	*When it runs in the page, the same MAIN-world limits as the extension.*
Android / WalletConnect	*In-wallet path*	*Own evaluator track and runbook in a3e9-tinl/doc.*	*Not a substitute for the Chrome track. Own UAT checklist.*

NORMALIZE no longer silently rewrites an unlimited approval to zero. The user is asked: this much, nothing (the transaction still costs a fee), or the unlimited value they were shown. Zero-as-default was safe as an allowance and misleading as a product.

Inside the cooperative evaluator portal, the portal owns both the interceptor and the consent UI, so WARN and NORMALIZE are enforced end to end in that demo. That is a property of the demo, not of the shipping extension. This paper will not collapse those two facts.

3. Surface: Price Impact

Price Impact is how TINL decides whether a swap moves the pool too far for the active profile. It is not a market-data product, not a price feed, and not a separate SKU. The evidence is in the same repository as Wallet Security.

What is implemented: constant-product ($x \cdot y = k$) math, shared by Uniswap V2 on EVM and Raydium AMM v4 on Solana — verified against each program's own source, not assumed because both are called AMMs. Mode B fetches live reserves, then calls that function, then applies the profile's WARN/BLOCK percentage thresholds. Integration tests drive the real router with a local RPC mock; they never guess a number when the pool is empty or the input is non-positive (the function returns None).

Precision is f64. That is enough for a percentage threshold, and it is not an accounting result. USD display enrichment (Coinbase / Chainlink / Pyth / CoinGecko, depending on the path) is a separate tier. If an oracle is unreachable the decision path degrades; it does not invent a price. That contract is tested by leaving those oracles genuinely unreachable.

Mode A (WASM) has no network by design. Live reserve fetch and live USD oracles are Mode B. Do not describe Price Impact as an offline embedded capability. Do not describe it as covering concentrated liquidity, order books, or every AMM on either chain. Constant-product is what is evidenced.

4. Profiles and the registry

Four shipped profiles: retail_strict, trader_balanced, institutional_strict, and dev_passthrough. Unlimited approve, setApprovalForAll, permits / SPL delegates, and unknown selectors are profile-specific. The open base registry is compiled into every build. A threat-intel overlay can be fetched in Mode B. Unknown selectors are not a uniform ALLOW.

An older engineering yellow paper under yellow_papers/TINL/ described a 24-hour SLA against named third-party feeds. This paper does not repeat that. The overlay schema and the compiled base registry are what the repository proves.

5. Auditability

Mode B writes an HMAC-chained record before it returns a decision. A failed write is fail-closed. Mode A cannot take a filesystem; the WASM build exposes a hash function and each host persists the chain (extension storage, Snap state, WalletConnect host storage). Those are not the same durability story. Say which mode you evaluated.

6. What this line is not

It is not HSM process attestation. a3e9-attestation is the signing daemon's binary and library story.

It is not "we intercept 100% of signing requests on every wallet." Coverage is the paths in the table in section 2, on the chains those rows name.

It is not a claim that WARN and NORMALIZE survive a hostile page in the shipping extension. They do not, until the consent channel is no longer page-visible. That limitation is reproduced, not merely reasoned about.

Related reading

Evaluator brief and threat model: a3e9-tinl/doc/TINL_EVALUATOR_BRIEF.md and TINL_THREAT_MODEL_AND_SAFEGUARDS.md. Chrome and Android UAT are separate checklists in that repository. HSM line: “The PKCS#11 Fragmentation Problem — And Why A3E9 Exists.” Trust Hub: “A3E9 II — Institutional Trust Hub.”

An earlier engineering paper remains at yellow_papers/TINL/TINL.docx (July 2026). Where that file and this one disagree, this file is the product-line paper and the July file is the older engineering draft. Where this paper and a3e9-tinl disagree, the repository is correct.