T3N Vendor Risk Agent

Terminal 3 Agent Build Challenge submission · August 30, 2026

One-line pitch

Confidential enterprise vendor-security intake: deterministic risk scoring inside a Terminal 3 trusted execution environment, tenant-owned storage, zero HTTP egress, and mandatory human review for high-risk results.

What was built

The Rust contract accepts a structured vendor-security questionnaire, scores it from 0 to 100 inside a WebAssembly TEE component, stores the complete assessment in a private tenant KV map, and returns a decision-ready summary that excludes the vendor name and free-text notes. A TypeScript control layer deploys it, grants only two named functions, and runs the test assessment.

Why it is useful

Operational value: Repeatable vendor-security triage without copying sensitive questionnaires into a general-purpose model endpoint.

Auditability: A transparent, deterministic rule set rather than an opaque model judgment.

Human control: Scores of 50 or higher require manual review; the agent cannot autonomously approve a high-risk vendor.

Maintainability: Scoring policy lives in one Rust module with regression tests and explicit versioning instructions.

Privacy and authorization model

No network egress: The WIT world imports tenant context, logging, and KV storage only; it deliberately omits HTTP.

Tenant-owned data: The full questionnaire, vendor identity, and notes remain in a private assessments map.

Minimal response: The returned summary contains the assessment ID, score, findings, risk level, and review flag only.

Scoped delegation: Only assess-vendor and get-assessment are authorized, with an empty allowed-host list.

Deployment evidence

The contract was compiled, tested, linted, and deployed to the Terminal 3 public test network. SDK 5.2.0 was pinned to preserve attestation verification while working around the manifest compatibility defect described later in this report.

Tenant DID: did:t3n:3e113c39dd091fd46819fe76524613282730f79f

Contract ID: 803

Script: z:3e113c39dd091fd46819fe76524613282730f79f:vendor-risk

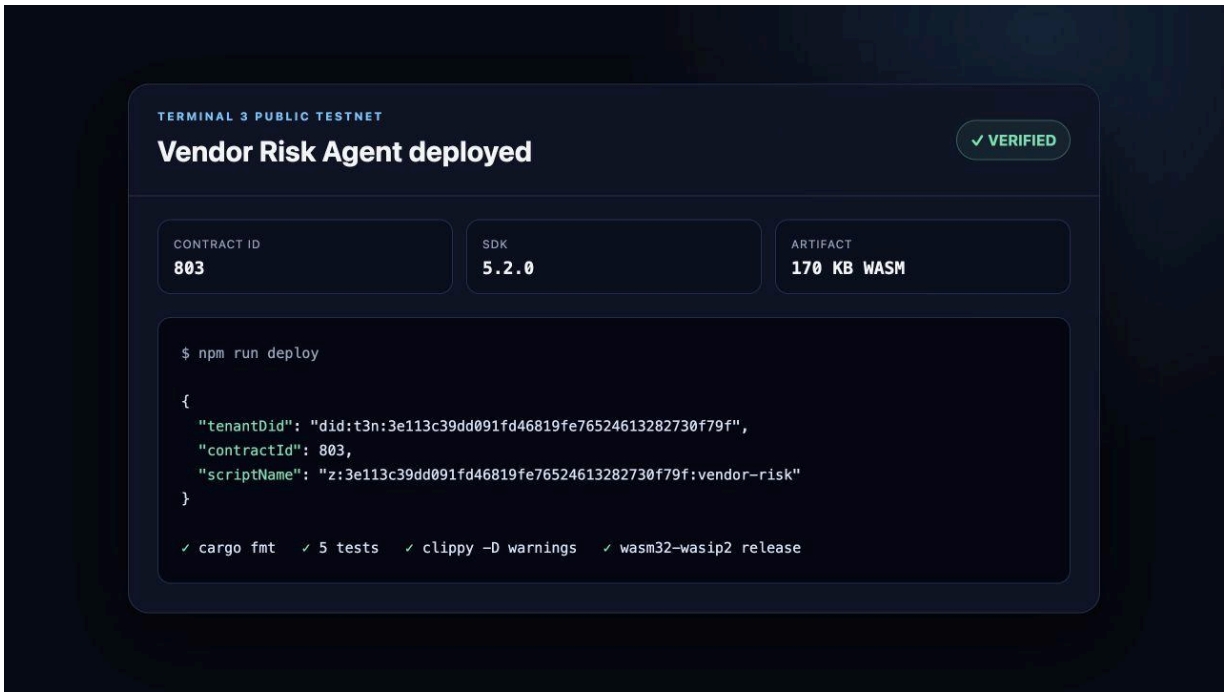

Figure 1. Verified contract deployment and local quality gates.

End-to-end result

A high-criticality payment processor questionnaire was evaluated through the deployed contract. The agent returned a score of 72, classified the vendor as high risk, and required human review. The summary omitted the vendor name and commercially sensitive notes as designed.

Figure 2. Scoped execution result with privacy-preserving response.

Verification results

Rust formatting: Passed

Unit tests: 5 passed, 0 failed

Clippy: Passed with warnings treated as errors

WASM release build: Passed; 170 KB artifact

TypeScript typecheck: Passed

Dependency audit: 0 vulnerabilities

Public agent identity

Terminal 3 hosts the ERC-8004-compatible agent card and serves it publicly from the agent DID. The published body advertises the DID service and TEE attestation support; it contains no private key, questionnaire data, or tenant notes.

Public card:
https://cn-api.sg.testnet.t3n.terminal3.io/api/agent-card/did:t3n:3e113c39dd091fd46819fe76524613282730f79f

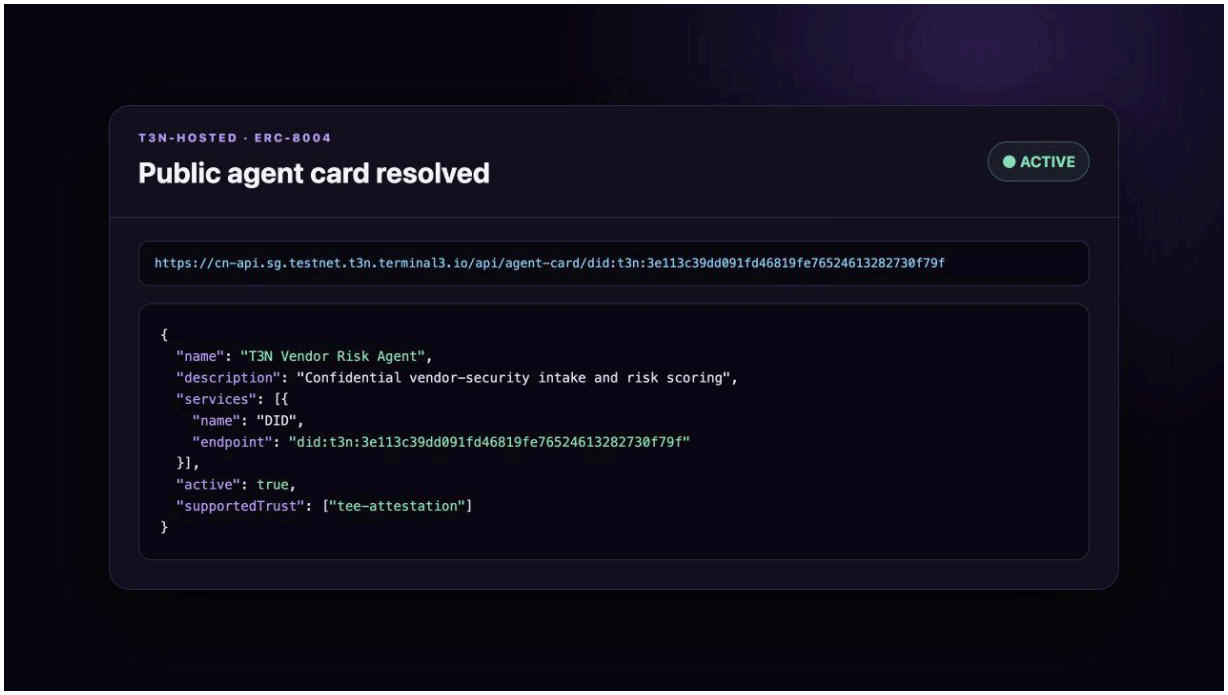

Figure 3. Publicly resolved T3N-hosted agent card.

Bugs and documentation friction

SDK 5.3.0 rejects the current testnet manifest

On August 30, 2026, the official trust-manifest endpoint returned HTTP 200 and a signed document containing rtmr3_allowlist but no rtmr1_allowlist. SDK 5.3.0 requires both fields and reported the manifest as malformed before authentication.

Pinning SDK 5.2.0 restored authentication and deployment without disabling attestation. The robust fix is to update the testnet manifest before the SDK rollout, or preserve backward compatibility during the transition.

A fresh agent key reused the existing DID

The public-agent guide says revisiting the claim page issues a fresh credited key for each agent and that the network binds the agent DID on first authentication. The page issued a cryptographically distinct second key, but reusing the same connected Google account caused both keys to authenticate as the same DID.

The page FAQ also states that signups are rate-limited to one per email. The platform should provision a distinct agent DID per freshly issued key or clearly require a different email and block the misleading success state.

Testnet and sandbox naming

The claim page uses sandbox while the quickstart uses testnet. In the current SDK both names resolve to the same public testnet node. A single canonical environment name would reduce setup ambiguity.

Maintenance handover

Scoring rules are isolated in src/risk.rs, and every rule change should add a regression test. Contract, Cargo, WIT, and deployment versions should be bumped together. Re-registering allocates a new contract ID, so the private map ACL must be updated to the new ID. Any future HTTP capability requires an explicit data-flow and authorization review.

The project is MIT licensed and prepared for continued maintenance after the challenge.