Security Audit Report

Prepared by: CYBRING
Prepared for: Ferra Node Indexer

Version: 1.0 (public version)
Date: 7th October 2025
Commit/Hash: 4503f704c934258401ea8ad8fb8454a2f16f768d
Auditor: Anh Bui, Duc Cuong Nguyen

Contents

1 Introduction

This document presents the results of the initial security assessment of the `Ferra` Node Indexer service. A TypeScript-based blockchain indexer that monitors and processes Sui network checkpoint data in real-time. The application downloads and parses binary checkpoint files using dynamically-loaded external parsers, then extracts and processes protocol-specific events through configurable worker modules. It supports multiple data sources with automatic fallback and pluggable storage backends for maintaining indexer state.

1.1 Objective

This security audit report was initially provided by `CYBRING` on 3rd October 2025. This audit was conducted to assess the robustness, reliability, and security of the code base. The goal was to identify vulnerabilities, ensure compliance with best security practices, and provide mitigation measures.

After the initial audit report, a reassessment was conducted on 7th October 2025 to verify the status of the reported issues. All issues were fixed.

1.2 Disclaimer

This security audit is not produced to replace any other type of assessment and does not aim to guarantee the discovery of all security issues within the scope of the assessment. Further, economic modeling, business-logic desirability, and market comparisons were explicitly out of scope.

While this audit was carried out in good faith and technical proficiency, it's crucial to understand that no single audit can guarantee the absolute security of a backend service. To minimize risks, `CYBRING` recommends a multi-faceted approach involving multiple independent assessments. Please note that this report is not intended to provide financial advice.

This is the public version; some operational details and PoCs were generalized or removed.

2 Scope of Audit

The audit was conducted on commit 4503f704c934258401ea8ad8fb8454a2f16f768d from git repository `https://github.com/Ferra-Labs/node-indexer`. Further details regarding The `Ferra` audit scope are provided below:

- **Codebase details:**
  - Language: TypeScript
  - Runtime: Node.js
  - Initial audit's commit hash: 4503f704c934258401ea8ad8fb8454a2f16f768d
  - Reassessment audit's commit hash: a0e4e30812ed8e621a1f87aad670705c30fa540d

3 Audit Summary

CYBRING assessed Ferra Node Indexer for security, reliability, and operational hardening. The initial review found 4 issues (1 High / 3 Medium; no Critical). The highest-risk item is runtime inclusion of unverified code. The assessment report is summarized in Table 1. Details of findings can be found in Section 5.

Severity	Count
High	1
Medium	3

Table 1: Initial assessment summary

All issues were fixed and verified on 07/10/2025 (commit a0e4e30). We recommend locking integrity controls into CI/CD, enforcing strict TypeScript/ESLint settings to prevent type drift. No critical issues remain in scope as of this publication. Summary of reassessment can be found in Table 2.

Issue	Severity	Status	Remediation Evidence
5.1	High	Fixed	0664a42
5.2	Medium	Fixed	885b143
5.3	Medium	Fixed	e75d420
5.4	Medium	Fixed	e945b2b

Table 2: Final assessment summary

4 Methodology

4.1 Audit Items

CYBRING conducts the following procedures to enhance the security posture of Ferra backend services:

- **Pre-auditing:** Understand business workflows and trust boundaries, review architecture (API gateway, services, DB, message queues, cache, files/storage), enumerate external integrations, and collect artifacts (OpenAPI/Swagger, env/config, deployment manifests, CI/CD pipeline overview).
- **Auditing:** Examine the service from multiple perspectives:
  - **Manual code review:** Inspect request handlers and middleware for input validation, authentication/authorization, multi-tenant scoping, unsafe patterns (e.g., unsanitized dynamic queries, SSRF, command/OS calls, path traversal), error handling, logging/redaction, and privacy. Verify schema-first validation is enforced on every route and response.
  - **Static analysis & supply-chain review:** Run linters and type checks (strict TS config), secret scanning, and SCA (dependency vulnerability and license

checks). Review `package.json` scripts, lockfile hygiene, dependency pinning, and transitive risk (e.g., postinstall scripts).

- **Dynamic testing (DAST) & API fuzzing:** Exercise endpoints against the OpenAPI contract with negative tests and fuzzing (boundary sizes, invalid types, malformed JSON, stateful sequences). Probe for authz bypass, IDOR, business-logic flaws, CORS/CSRF misconfig, rate-limit gaps, regex DoS (ReDoS), large-payload DoS, and event-loop blocking.
- **Configuration & deployment review:** Evaluate security plugins (e.g., rate-limit, cors, CSRF protections when using cookies), HTTP security headers, TLS/HSTS at the edge, cookie attributes, environment configuration, secrets management, Docker/K8s hardening (user, fs perms, resource limits), and CI/CD guardrails.
- **Authentication & authorization testing:** Verify token/session lifecycle (rotation, revocation, device binding), claim validation (iss/aud/exp/nbf), privilege boundaries, tenant isolation, and reference-monitor enforcement on *every* sensitive action.
- **Resilience & availability checks:** Assess input size/time complexity limits, file upload controls, compression/zip-bomb defenses, outbound egress controls, and graceful error handling; sample load to identify hot paths that can block the event loop.

- **First deliverable and consulting:** Provide an initial report, affected endpoints, risk ratings, and prioritized remediation guidance. Offer implementation Q&A.
- **Reassessment:** Verify fixes, re-test edge cases, and check for regressions or newly introduced issues.
- **Final deliverable:** Deliver a comprehensive report with final statuses, risk summaries, and hardening checklist.

Details of audit items can be found in Appendix A

4.2 Risk Rating

The OWASP Risk Rating Methodology is applied to backend issues using:

- **Likelihood:** how easily the flaw can be discovered/exploited (preconditions, attack surface, required auth/role, exploit reliability).
- **Impact:** business impact on confidentiality, integrity, availability, financial loss, privacy, and compliance.

4.3 Audit Categories

- **Common vulnerabilities:** Assessed against OWASP Top 10 (Web) and OWASP API Security Top 10 (2023).

Impact		Likelihood		
		Low	*Medium*	*High*
High		Medium	High	Critical
Medium		Low	Medium	High
Low		Informational	Low	Medium

Table 3: Overall Risk Severity

- **Advanced vulnerabilities:** Targeted exploitation of business logic, multi-tenant isolation, and chained flaws (e.g., SSRF → metadata access, authn bypass → IDOR).
- **Security best practices:** Coding standards, TypeScript strictness, schema-first validation, secure plugin configuration, deployment hardening, observability, and incident readiness.

5 Detailed Findings

5.1 Dynamic remote code execution without integrity verification

- **Location**: SUI module
- **Description:** You fetch arbitrary JS (index.js) from GitHub releases and execute it via dynamic import / temp file creation. No checksum/signing/integrity check, so a compromised release (or MITM if TLS trust is broken) gives full code execution.
- **Risk:** High (Impact: High, Likelihood: Medium)
- **CWE:** CWE-494: Download of Code Without Integrity Check
- **Mitigation:** Verify downloaded parsers using a signed release (e.g., Sigstore/-Cosign) or, at minimum, SHA-256 pinned to a trusted public key. Fail-closed on mismatch;
- **Status [07/10/2025]:** `Ferra` team has fixed this issue. Release artifacts are signed and runtime verification blocks execution on mismatch.

5.2 Potentially invalid TypeScript causing runtime drift or build failure

- **Location**: Object module
- **Description:** Methods:
  - `public static isObject(self: &Object)`
  - `isPackage(self: &Object)`

  use Rust-like syntax `&Object` (invalid TS). If this file is actually compiled, it should fail; if type errors are being suppressed, you risk logic silently diverging or methods never used.

- **Risk:** Medium (Impact: Medium, Likelihood: Medium)
- **CWE:** CWE-561: Dead Code
- **Mitigation:** Change to self: Object and add proper type narrowing.
- **Status [07/10/2025]:** `Ferra` team has fixed this issue.

5.3 Event listener removal bug

- **Location**: Worker module
- **Description:** `this.listenter[eventName].filter(f => f === callback);` the result is unused, hence listener is not removed.
- **Risk:** Medium (Impact: Medium, Likelihood: Medium)
- **CWE:** CWE-404: Improper Resource Shutdown or Release
- **Mitigation:** Reassign:

  `this.listeners[eventName] = this.listeners[eventName].filter(...);` after filtering.
- **Status [07/10/2025]:** `Ferra` team has fixed this issue.

5.4 Incorrect version comparison

- **Location**: SUI module
- **Description:** String comparison for version numbers (e.g., `"1.9.0" > "1.10.0"` as strings). This means, the supposedly newer version might be older.
- **Risk:** Medium (Impact: Medium, Likelihood: Medium)
- **CWE:** CWE-1025: Comparison Using Wrong Factors
- **Mitigation:** Convert to number and compare using *semver* library.
- **Status [07/10/2025]:** `Ferra` team has fixed this issue.

A Appendix

Area	**Representative Checks (Indexer / TypeScript)**
Chain Data Authenticity	Multi-source cross-check; finality thresholds; digest/signature validation; fail-closed on mismatch.
Checkpointing & Durability	Per-item commits; idempotent upserts; atomic watermark; crash-consistent restart without gaps/dupes.
Reorg & Replay Handling	Bounded rollback; replay windows; clear at-least-once/exactly-once semantics; compensating actions.
Concurrency & Ordering	Key-based partitioning; ordered processing where required; dedupe keys; bounded queues.
Backpressure & Flow Control	Adaptive batching; queue depth caps; throttle RPC when sinks lag.
RPC Usage & Throttling	Timeouts; retries with jitter; rate limits; pagination correctness; exponential backoff.
Data Model & Serialization	Stable schemas; bigint/decimal precision; canonicalization; versioned migrations.
Storage & Indexing Safety	UPSERT/unique keys; transaction boundaries; isolation levels; hot-partition mitigation.
Data Quality & Gap Detection	Monotonic height invariants; gap/overlap detectors; duplicate-rate counters; alerting on drift.
Performance & DoS Resilience	Bounded memory; streaming parsers; limits for giant blocks/txs; avoid event-loop blocking.
Dynamic Code & Supply Chain	No network-loaded code; pinned versions + SHA/Sigstore; reproducible builds; audit postinstall scripts.
Observability & Ops	Metrics: head lag, throughput, retries, queue depth; SLO alarms; runbooks for reorg/catch-up.

Table 4: Core indexer audit items assessed by `CYBRING`.