Security Audit Report

Prepared by: CYBRING
for: Ferra CLMM Backend

Version: 0.1 (initial report)
Date: 29th Sep 2025
Commit/Hash: 3e51875e369956bee0bec879c8da60c7a4931a64
Auditor: Anh Bui, Duc Cuong Nguyen

Contents

1 Introduction

This document presents the results of a security assessment of the `Ferra`'s CLMM API service. The project is a backend API for a Concentrated Liquidity Market Maker, exposing structured endpoints for pool discovery, pricing, positions, and historical metrics derived from on-chain events on Sui. Implemented in TypeScript with Node.js, it ingests, normalizes, and aggregates protocol activity (pools, ticks, liquidity, rewards) into a relational store, layering caching and job queues for performance.

1.1 Objective

This security audit report was initially provided by `CYBRING` on 9th September 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 29th Sep 2025 to verify the status of the reported issues. All issues were acknowledged and will be prioritized for future improvement.

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 (e.g., target price ceilings relative to other launch platforms) 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 initial audit was conducted on commit 3e51875e369956bee0bec879c8da60c7a4931a64 from git repository `https://github.com/Ferra-Labs/ferra-CLMM-api/`. Further details regarding The `Ferra` audit scope are provided below:

- **Codebase details:**
  - Language: TypeScript
  - Runtime: Node.js
  - Initial audit's commit hash: 3e51875e369956bee0bec879c8da60c7a4931a64
  - Reassessment audit's commit hash: c7da5d8eaa2676f660c97a29f6497cfae7d59b6c

3 Audit Summary

As of 29th Sep 2025, we identified 13 issues (1 Critical, 2 High, 6 Medium, 4 Low). The Critical was a SQL injection in dynamic queries; High severities included an async retry/bail bug and an unscoped pool filter. Most High/Medium items were fixed, with two Mediums acknowledged for follow-up. Immediate priorities: keep queries parameterized, enforce strict TLS, restrict CORS, harden headers (HSTS/CSP), rotate secrets, and separate crypto keys.

The assessment report is summarized in Table 1. Details of findings can be found in Section 5.

Severity	Count
Critical	1
High	2
Medium	6
Low	4

Table 1: Initial assessment summary

Summary of reassessment can be found in Table 2.

Issue	Severity	Status	Remediation Evidence
5.1	Critical	Fixed	c7da5d8e
5.2	High	Fixed	c7da5d8e
5.3	High	Fixed	c7da5d8e
5.4	Medium	Fixed	c7da5d8e
5.5	Medium	Fixed	c7da5d8e
5.6	Medium	Fixed	c7da5d8e
5.7	Medium	Acknowledged	
5.8	Medium	Fixed	c7da5d8e
5.9	Medium	Acknowledged	c7da5d8e
5.10	Low	Acknowledged	
5.11	Low	Fixed	c7da5d8e
5.12	Low	Fixed	c7da5d8e
5.13	Low	Fixed	c7da5d8e

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), enu-

merate 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.

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

Table 3: Overall Risk Severity

4.3 Audit Categories

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

- **Location**: Pool service
- **Description:** String interpolation of user-controlled values into SQL instead of parameter binding.
- **Risk:** Critical (Impact: High, Likelihood: High)
- **CWE**: CWE-89: Improper Neutralization of Special Elements used in an SQL Command.
- **Mitigation:** Always parameterize SQL query/commands.
- **Status [29/09/2025]:** `Ferra` team has fixed this issue.

5.2 Transaction retry/bail bug due to missing await causes non-retryable errors to be retried

- **Location**: Database layer
- **Description:** If *singleTxFn* returns a Promise and isn't awaited, errors won't be caught by that try/catch. The bail path will not execute for async errors you intended to handle specially.
- **Risk:** High (Impact: High, Likelihood: Medium)
- **CWE**: CWE-754: Improper Check for Unusual or Exceptional Conditions
- **Mitigation**: Change return *singleTxFn(bail)* to return *await singleTxFn(bail)*.
- **Status [29/09/2025]:** `Ferra` team has fixed this issue.

5.3 Incorrect filter in getPools

- **Location**: Pool service
- **Description:** In *getPools*, the code builds a string *statusFilter* tries to add coin filters only if *statusFilter* is already non-empty. When status=ALL (default), *statusFilter* is "", so the coin clause is never appended.
- **Risk:** High (Impact: High, Likelihood: Medium)
- **CWE**: CWE-840: Business Logic Errors.
- **Mitigation:** Do not filter coins with statusFilter. Collect predicates in an array and always join them into a WHERE,
- **Status [29/09/2025]:** `Ferra` team has fixed this issue.

5.4 Hardcoded credentials

- **Location**: Core library
- **Description:** API key is hardcoded.
- **Risk:** Medium (Impact: Medium, Likelihood: Medium)
- **CWE**: CWE-798: Use of Hard-coded Credentials
- **Mitigation:** To fix this issue and prevent it from happening again, follow these steps:
  - Remove the key from your codebase and its Git history.
  - Immediately rotate the key and load it from a secret manager.
  - Implement a pre-receive secret scanning to block commits containing secrets.
  - Add a CI/CD secret scanning step to your pipeline.
  - Block the commitment of `.env` files and similar sensitive configuration files.
- **Status [29/09/2025]:** `Ferra` team has fixed this issue.

5.5 Sentry handler registered but no initialization

- **Location**: Server's config
- **Description:** The application attempts to set up a Sentry error handler, but the Sentry SDK has not been properly initialized.
- **Risk:** Medium (Impact: Medium, Likelihood: Medium)
- **CWE**: CWE-778: Insufficient Logging.
- **Mitigation:** Initialize Sentry (DSN, environment, tracesSampleRate) or remove setup call
- **Status [29/09/2025]:** `Ferra` team has fixed this issue.

5.6 Using the same key for JWT and Encryption

- **Location**: environment variables (e.g., .env)
- **Description:** *[redacted 1]* defaults to *[redacted 2]*. Reusing the same key for symmetric JWT and AES increases blast radius if one leaks.
- **Risk:** Medium (Impact: Medium, Likelihood: Medium)
- **CWE**: CWE-323: Reusing a Nonce, Key Pair in Encryption
- **Mitigation:** Require separate strong keys
- **Status [29/09/2025]:** `Ferra` team has fixed this issue.

5.7 Wide open CORS

- **Location**: Server's config
- **Description:** *[redacted]* with no options effectively allows any origin.
- **Risk:** Medium (Impact: Medium, Likelihood: Medium)
- **CWE**: CWE-942: Permissive Cross-domain Policy with Untrusted Domains
- **Mitigation**: *[redacted]* needed.
- **Status [29/09/2025]:** `Ferra` team has acknowledged this issue. All endpoints are currently read-only.

5.8 Missing important security headers

- **Location**: Server's config
- **Description:** There are several missing items with the application's HTTP security headers: *[redacted]*
- **Risk:** Medium (Impact: Medium, Likelihood: Medium)
- **CWE**: CWE-1173: Improper Use of Validation Framework
- **Mitigation**: To improve the security of your application's HTTP headers, the following actions should applied: *[redacted]*
- **Status [29/09/2025]:** `Ferra` team has fixed this issue.

5.9 Errors may leak internal logic details

- **Location**: Error handler
- **Description:** The API responds to invalid requests with potential validation artifacts.
- **Risk:** Medium (Impact: Medium, Likelihood: Medium)
- **CWE**: CWE-209: Generation of Error Message Containing Sensitive Information
- **Mitigation**: Wrap PG/Redis errors (provide generic responses) and never JSON.stringify(err) to clients.
- **Status [29/09/2025]:** `Ferra` team has acknowledged this issue.

5.10 JWT is not used

- **Location**: Core library
- **Description:** JWT is registered, imported but never used.
- **Risk:** Low (Impact: Low, Likelihood: Low)
- **CWE**: CWE-561: Dead Code
- **Mitigation**: To resolve this issue, you must secure the APIs by either removing all authentication-related code and documentation if they're meant to be public and read-only, or by implementing robust authentication and authorization.
- **Status [29/09/2025]:** `Ferra` team has acknowledged this issue.

5.11 Redis client is created but not closed on shutdown

- **Location**: Server's config
- **Description:** A new Redis client is created for the rate-limit plugin (line 13) and never closed on shutdown.
- **Risk:** Low (Impact: Low, Likelihood: Low)
- **CWE:** CWE-772: Resource Leak.
- **Mitigation**: Add hook on 'onclose' for quit (*rateLimitRedis.quit*) or disconnect (*rateLimitRedis.disconnect*)
- **Status [29/09/2025]:** `Ferra` team has fixed this issue.

5.12 Abuse-prevention controls insufficient (rate/ban/telemetry)

- **Location**: Server's config
- **Description:** Abuse controls omit onExceeded/ban callbacks for detection and temporary blocking.
- **Risk:** Low (Impact: Medium, Likelihood: Low)
- **CWE**: CWE-770: Allocation of Resources Without Limits or Throttling
- **Mitigation**: Use onExceeded/ban to log and optionally block abusive clients.
- **Status [29/09/2025]:** `Ferra` team has fixed this issue.

5.13 Outdated Third-Party Dependencies

- **Location**: dependency list
- **Description:** An analysis of the project's dependencies identified several third-party libraries that are either outdated or contain known security vulnerabilities. Using these vulnerable components can expose the application to various risks.

  The following dependencies were identified as outdated and should be updated to their specified minimum versions to resolve known vulnerabilities: *[redacted]*
- **Risk:** Low (Impact: Medium, Likelihood: Low)
- **CWE:** CWE-1104: Use of Unmaintained Third Party Components
- **Mitigation:** To mitigate this finding, all identified dependencies should be updated to their specified secure versions. Regularly updating dependencies is a critical practice for maintaining the security and stability of the application.
- **Status [29/09/2025]:** `Ferra` team has fixed this issue.

A Appendix

Area	Representative Checks (TypeScript)
Input & Data Validation	JSON schema on all routes (AJV/TypeBox/Zod); type coercion off where unsafe; max body size; strict content types; canonicalization; defense against prototype pollution and mass assignment.
Authentication	Passwordless/OIDC/JWT/session design; token storage (cookies vs. headers); rotation/revocation; session fixation; cookie flags (HttpOnly, Secure, SameSite); MFA hooks where applicable.
Authorization & Tenancy	RBAC/ABAC enforcement on every handler; resource-scoped checks; IDOR; insecure direct joins; cross-tenant data leakage; reference monitor centralization.
Crypto & Secrets	Key management, JWKS pinning/rotation, alg selection; env secret handling (no secrets in repo/logs); KMS/secret store usage; TLS config.
Transport & Headers	HSTS, CSP, X-Frame-Options/frame-ancestors, X-Content-Type-Options, cache policy; security-header middleware (Helmet-style) configuration; redirect safety.
CORS & CSRF	Origin/host validation; allowed methods/headers; credentialed requests; CSRF defenses; preflight handling.
Business Logic	Workflow abuse, replay, re-entrancy-like sequences across endpoints, order-of-operations, race conditions, money/-points/limits enforcement.
Storage & Query Safety	SQL/NoSQL injection; Prisma/TypeORM query patterns; transaction boundaries; isolation levels; migration safety; pagination/limit caps.
SSRF & Egress	URL fetchers (DNS rebinding, IP allow-list bypass, localhost/metadata access), redirect chaInspins, file://, gopher://, and %2F tricks.
File Handling	MIME sniffing, extension checks, upload size/number limits, image processing safety, decompression bombs, antivirus hooks (if applicable).
DoS	Performance Rate limits (middleware or gateway); per-IP/account quotas; slowloris protection; timeouts; ReDoS; event-loop blocking hotspots.
Error Handling & Logging	No stack traces to clients; safe error codes; PII/secret redaction; correlation IDs; security/audit logs with tamper resistance.
Dependency & Build	SCA results; lockfile hygiene; banned/postinstall scripts; reproducible builds (`npm ci`); TS compiler strictness; `ts-node` not in prod.
Config & Deploy	Docker/K8s least privilege (non-root, RO FS); health/readiness; env var validation; feature flags; production toggles; debug endpoints disabled.
Observability & IR	Metrics/alerts for auth failures, 4xx/5xx spikes, rate-limit hits; trace sampling; runbooks and incident hooks.
Privacy & Compliance	Data minimization; retention/TTL; access logging for sensitive records; export/delete workflows; consent and notice.
Documentation & API Contract	OpenAPI accuracy; response schemas; deprecation policy; versioning; idempotency keys for mutating endpoints.

Table 4: Backend (TypeScript) audit items assessed by `CYBRING`.