Security Audit Report

Prepared by: `CYBRING`
Prepared for: `Ferra Aggregator`

Version: 1.0 (public version)
Date: 29th Sep 2025
Commit/Hash: 03e0b18c13a3c47bf70baf123cba01024e662770
Auditor: Anh Bui, Duc Cuong Nguyen

Contents

1 Introduction

This document presents the results of the initial security assessment of the Ferra's Aggregator. It is a modular smart-routing quote engine that models on-chain liquidity as a directed graph, enumerates bounded multi-hop swap paths, simulates execution to rank routes, and (optionally) splits order flow across top candidates for efficiency.

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 audit was conducted on commit 03e0b18c13a3c47bf70baf123cba01024e662770 from git repository `https://github.com/Ferra-Labs/ferra-agg/`. Further details regarding The Ferra audit scope are provided below:

- **Codebase details:**
  - Language: TypeScript
  - Runtime: Node.js
  - Initial audit's commit hash: 03e0b18c13a3c47bf70baf123cba01024e662770
  - Reassessment audit's commit hash: c7da5d8eaa2676f660c97a29f6497cfae7d59b6c

3 Audit Summary

CYBRING assessed Ferra Aggregator through 9th September 2025, focusing on routing correctness, API safety, and operational hardening. We initially identified 11 issues (3

High, 3 Medium, 5 Low). The most material risks involved token/route context mix-ups during path optimization and cache keying gaps that could return stale/cross-direction quotes. As of commit c7da5d8e, the team remediated all High and Medium findings; one Low (abuse-control depth) is acknowledged for future improvement. We recommend continuing defense-in-depth hardening and dependency hygiene.

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

Severity	**Count**
High	3
Medium	3
Low	5

Table 1: Initial assessment summary

Summary of reassessment can be found in Table 2.

Issue	**Severity**	**Status**	**Remediation Evidence**
5.1	High	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	Low	Fixed	c7da5d8e
5.8	Low	Fixed	c7da5d8e
5.9	Low	Fixed	c7da5d8e
5.10	Low	Acknowledged	
5.11	Low	Acknowledged	

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.

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 Incorrect tokenIn used during iterative optimization

- **Location**: Path finding module
- **Description:** `rateLow` uses `tokenIn` from `highPath` instead of `lowPath`, resulting in incorrect rates.
- **Risk:** High (Impact: High, Likelihood: High)
- **CWE**: CWE-840: Business Logic Errors
- **Mitigation**: *[redacted]*
- **Status** [29/09/2025]**:** `Ferra` team has fixed this issue.

5.2 Incorrect tokenIn when re-normalizing allocations

- **Location**: Path finding module
- **Description**: After low-weight paths are filtered out, the allocator re-normalizes the remaining allocations so that the sum equals the original input amount. During this re-normalization pass, each quote is recomputed via

  [redacted]

  which hard-codes the first path's `tokenIn` for *all* paths. This overwrites per-path input context previously used.
- **Risk:** High (Impact: High, Likelihood: High)

- **CWE**: CWE-840: Business Logic Errors
- **Mitigation**: Carry tokenIn with each WeightedQuote (extend type) and use that in recompute.
- **Status [29/09/2025]:** `Ferra` team has fixed this issue.

5.3 Cache key collisions / stale or wrong quotes

- **Location**: Path finding module
- **Description:** The *[redacted]* function contains a logic flaw because it ignores the trade's direction. This oversight can lead to the system providing a quote for the wrong direction or from stale reserves.
- **Risk:** High (Impact: High, Likelihood: High)
- **CWE:** CWE-694: Use of Multiple Resources with Duplicate Identifier
- **Mitigation**: *[redacted]*
- **Status [29/09/2025]:** `Ferra` team has fixed this issue.

5.4 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.

5.5 DLMM price impact hardcoded to zero

- **Location**: Path finding module
- **Description:** The aggregator's price-impact helper returns 0 for all DLMM pools, effectively declaring that trades against DLMM liquidity incur no slippage. This is incorrect for Meteora-style Dynamic/Discrete Liquidity Market Makers where execution walks through bins with finite per-bin depth. Once a bin's inventory is consumed, execution moves to the next bin at a different price.
- **Risk:** Medium (Impact: Medium, Likelihood: Medium)
- **CWE:** CWE-840: Business Logic Errors

- **Mitigation**: Implement DLMM curve impact or remove the function until implemented.
- **Status [29/09/2025]:** `Ferra` team has fixed this issue.

5.6 Errors may leak internal logic details

- **Location**: Error handler
- **Description:** The API responds to invalid requests with verbose validation artifacts (validationError.message, fieldErrors from the schema). These payloads often include internal field names, enum/regex rules, range limits, and branching logic. Returning them to untrusted clients reveals how inputs are validated and how business rules are enforced, aiding targeted bypasses and automated probing.
- **Risk:** Medium (Impact: Medium, Likelihood: Medium)
- **CWE:** CWE-209: Generation of Error Message Containing Sensitive Information
- **Mitigation**: Return a safe error envelope (fixed code, brief message, opaque correlationId); log full stack/details server-side only. Mask validation artifacts (schema paths, regexes, enum lists). Wrap DB/cache/lib errors into generic messages.
- **Status [29/09/2025]:** `Ferra` team has fixed this issue.

5.7 Slippage/minAmountOut rounding issues

- **Location**: Path finding module
- **Description:** Line 145-146 use `toFixed(0)` while line 155 used `toFixed(6)` despite all variables being "amount" related.
- **Risk:** Low (Impact: Low, Likelihood: Low)
- **CWE:** CWE-1339: Insufficient Precision/Accuracy.
- **Mitigation**: Consistent precision config: set a global Decimal precision (e.g., 40) and explicit rounding mode (e.g., ROUND_DOWN for outputs). Avoid *Number*, *toFixed*, and implicit JS float ops in core math.
- **Status [29/09/2025]:** `Ferra` team has fixed this issue.

5.8 Price impact calculation vulnerability

- **Location**: Quote service
- **Description:** Precision loss *const sqrtPrice = Math.sqrt(currentPrice.toNumber());* Overflow vulnerability: *[redacted]*. The code derives currentPrice from the executed trade (amountOut/amountIn after decimal scaling), then computes *currentSqrtPrice* via sqrt(currentPrice) and scales by 2ˆ64 using JS floats.
- **Risk:** Low (Impact: Low, Likelihood: Low)

- **CWE:** CWE-190: Integer Overflow or Wraparound
- **Mitigation**: No JS floats in financial math. Keep values as Decimal/BN.js/BigInt throughout. Bound checks before scaling.
- **Status [29/09/2025]:** `Ferra` team has fixed this issue.

5.9 CSRF not used

- **Location**: Server's config
- **Description:** csrfPlugin exists but is not registered.
- **Risk:** Low (Impact: Low, Likelihood: Low)
- **CWE:** CWE-352 Cross-Site Request Forgery
- **Mitigation**: If the API is token/Authorization-header only and no cookies, remove CSRF code as it's not needed.
- **Status [29/09/2025]:** `Ferra` team has fixed this issue.

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

- **Location**: *[redacted]*
- **Description:** Abuse controls omit *[redacted]* callbacks for temporary blocking.
- **Risk:** Low (Impact: Medium, Likelihood: Low)
- **CWE:** CWE-770; Allocation of Resources Without Limits or Throttling
- **Mitigation**: *[redacted]*
- **Status [29/09/2025]:** `Ferra` team has acknowledged this issue.

5.11 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.
- **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 acknowledged 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`.