House Of Stake
Super Majority Voting
Security Review Report

01-15-2026

COMPREHENSIVE REPORT

Prepared by:
Valhalla Security Consulting LLC

DISTRIBUTION (Public Version)

This report is intended for general informational use and provides an overview of the security testing performed by Valhalla Security Consulting LLC (Valhalla Security). All proprietary or sensitive details from the original assessment have been removed to ensure this document can be shared publicly.
The testing approach followed industry-standard penetration testing practices, adapted by Valhalla Security for this engagement. This public version may be freely distributed, referenced, or included in external communications as needed.

CONFIDENTIALITY (Public Version)

This public report has been prepared to summarize the high-level findings of the security assessment in a manner suitable for external audiences. All confidential, sensitive, or identifying information has been excluded.
Valhalla Security maintains strict confidentiality regarding all client security data. More detailed technical results, artifacts, or sensitive findings are contained only in the private version of this report and are shared exclusively with authorized client stakeholders.

ASSESSMENT TEAM

The following Valhalla Security personnel are involved in this engagement. Contact the appropriate personnel on this team to discuss the contents of this document or the work performed during this engagement.

PRIMARY SECURITY CONSULTANT

Ron S

Sr. Security Auditor

ADDITIONAL SECURITY CONSULTANTS

Sean M
CEO – Valhalla Security

Alex B
CTO – Valhalla Security

House of Stake - Super Majority Voting Security Review

Executive Summary

House of Stake enlisted the services of Valhalla Security to perform a time-boxed assessment of the organization's proposed Metadata-Encoded Proposal Description & Enhanced Proposal Flow. The engagement started on December 22nd, 2025 for a one week review, and ended on December 31st, 2025.

A phased approach to the code audit included information gathering, manual and automated testing, source code review, and adherence to best practices.

Valhalla Security identified no immediate critical high or low risks that would be blockers of this change that would lead to compromise or downstream risks. All findings were remediated, verified, and thoroughly tested.

Scoped Repositories and Commits

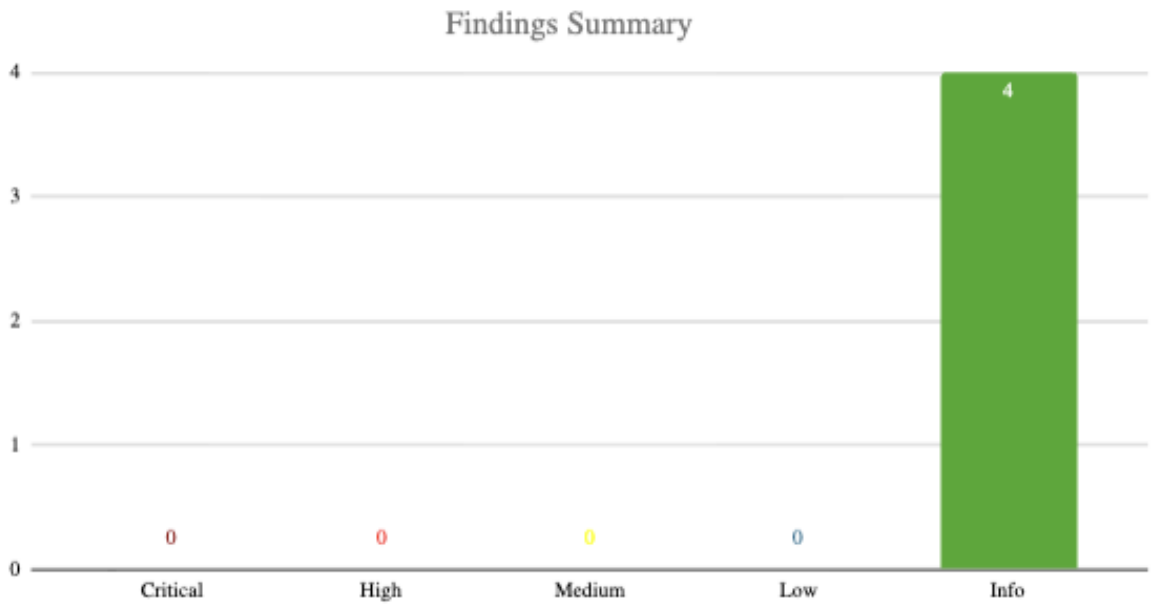
agora-near (frontend)
Repository: https://github.com/voteagora/agora-near
Commit hash: 39a4ab502bac80fb10ff79c05296277a1f085711
Proposals PR: https://github.com/voteagora/agora-near/pull/237

agora-near-be (backend)
Repository: https://github.com/voteagora/agora-near-be
Commit hash: b91dada3aa6419567e58ceb41644922693f074d1
Proposals PR: https://github.com/voteagora/agora-near-be/pull/61

Area of Focus

- Web3 Integration
- Payload Manipulation
- Field Injection
- Metadata Max / Min Lengths
- Pipeline Termination
- Cross Site Scripting (XSS)
- Quota and Majority Manipulation
- Third Party Libraries and Dependencies
- Github Actions and CI best practices review

Finding Summary



Opportunities for improvements are identified below:

Informational
1 - Github Code Signing Not Enforced - ACKNOWLEDGED
2 - Missing Pull Request and Merge Strategy - ACKNOWLEDGED
3 - Github Commits Expose Developer Email Addresses - ACKNOWLEDGED
4 - Vulnerable Third Party Libraries and Packages - FIXED

Positive Findings

The security audit revealed several positive findings that establish a solid security foundation and contribute to a robust overall posture, in addition to identifying areas for improvement. The key strengths where the system demonstrates effective security practices include:

- **Fund Security:** Changes to the specification ensure no direct risk to funds, as assets are not automatically moved upon a proposal's passage.
- **Secure Development Practices:** The design methodology and process were sound, with the web front-end framework implementing input validation and sanitation to prevent HTML and JavaScript injection.
- **Absence of Hardcoded Secrets:** No hardcoded secret variables (such as tokens, private keys, DB credentials, or environment variables) were found in the repositories.

These positive attributes reflect strong adherence to secure coding practices. They collectively enhance product resilience against common threats, support long-term security objectives, and should be maintained and leveraged in future development and revisions.

Metadata Encoding Standard
Proposals encode governance metadata in the description field using a binary-safe format:

Format: **[PREFIX][VERSION][CONTENT]**

- **Prefix**: \u001E\u001E\u001E\u001E (4 bytes - ASCII Record Separators)
- **Version**: \u0001\u0001 (2 bytes - Version 1)
- **Content**: |proposal_type=SimpleMajority (Pipe-delimited key-value pairs)

Informational Risk

1 - Github Code Signing Not Enforced - ACKNOWLEDGED	
Severity	**Informational**
Repository	**agora-near / agora-near-be**
Description	The audited GitHub repository does not require or enforce commit signing with GPG keys. This means there's no cryptographic verification to ensure commits come from trusted developers.
Impact	Not a direct vulnerability, but lack of code signing could make it easier for malicious code from a compromised developer account or laptop to be merged through unnoticed.
Proposed Fix	Enforce Code Signing on All Repositories Step 1: Access Branch Protection Settings Step 2: Configure Branch Protection to Require Signed Commits Step 3: Check **Require signed commits**

2 - Missing Pull Request and Merge Strategy - ACKNOWLEDGED	
Severity	**Informational**
Repository	**agora-near / agora-near-be**
Description	The audited GitHub repository allows development activity to be committed directly to the **main/master** branch without requiring pull requests (PRs). There's no enforced process to ensure a separate QA/merge role reviews changes before they're merged, which could allow unverified or malicious code to be pushed, especially if a developer's account or laptop is compromised.
Impact	Malicious code risk if developer account or laptop is compromised.

2 - Missing Pull Request and Merge Strategy - ACKNOWLEDGED	
Proposed Fix	Set Branch Protection Settings **Step 1: Access Branch Protection Settings** 1. Navigate to the repository 2. Click the **Settings** tab at the top. 3. In the left sidebar, under **Code and automation**, click **Branches**. 4. Under **Branch protection rules**, find the main or master branch or click **Add rule** to create a new one. **Step 2: Configure Branch Protection to Require Pull Requests** 1. In the branch protection rule settings, check **Protect this branch**. 2. Check **Require a pull request before merging** to block direct commits to main or master. 3. Check **Require approvals** and set the minimum number of reviewers (e.g., 1 or 2) to ensure a separate QA/merge role reviews the PR. 4. Check **Require signed commits** to ensure commits are cryptographically verified, further reducing risk from compromised accounts. **Step 3: Additional Protections and Verification** 1. Check **Dismiss stale pull request approvals when new commits are pushed** to ensure fresh reviews if code changes after initial approval.

3 - Github Commits Expose Developer Email Addresses - ACKNOWLEDGED	
Severity	**Informational**
Repository	**agora-near / agora-near-be**
Description	Using personal email addresses for Github commits can expose this data in commit and patch data.
Impact	Developers may be more prone to phishing attacks and attacks on personal email accounts.
Proposed Fix	Github has privacy and protection controls in order to hide email addresses. It is recommended to implement this in order to prevent privacy leaks. **Steps** 1. Go to GitHub Settings → Emails 2. Navigate to https://github.com/settings/emails 3. Find your no-reply address at the bottom of the page GitHub provides a unique no-reply email in the format: - USERNAME@users.noreply.github.com - Or with ID: 12345+USERNAME@users.noreply.github.com 4. Enable privacy protection:

3 - Github Commits Expose Developer Email Addresses - ACKNOWLEDGED	
	- Check "Keep my email addresses private" - Check "Block command line pushes that expose my email" **Update Git Configuration** git config --global user.email "USERNAME@users.noreply.github.com" **Verify Configuration** git config --list | grep email **Test with a sample commit** git commit --allow-empty -m "Test commit to verify no-reply email" git log --pretty=format:"%an <%ae>" -1

4 - Vulnerable Third Party Libraries and Packages - FIXED	
Severity	**Informational**
Repository	**agora-near**
Description	The audited GitHub repository uses third-party packages and dependencies that may have known vulnerabilities due to outdated versions. These packages, such as libraries or frameworks, are not regularly updated to their latest secure versions, leaving the codebase exposed to potential exploits.
Impact	Failing to update vulnerable third-party dependencies can increase the risk of security breaches or exploits.
Affected Packages	**Critical Severity** **form-data** (< 2.5.4) Uses unsafe random function for boundary generation, leading to potential security issues **elliptic** (< 6.6.1) Private key extraction vulnerability in ECDSA when signing malformed input (e.g., strings) **High Severity** **next** (multiple versions) < 14.2.10: Cache poisoning vulnerability < 14.2.15: Authorization bypass < 14.2.25: Authorization bypass in middleware < 14.2.34: Denial of Service with Server Components < 14.2.35: Denial of Service with Server Components (incomplete fix follow-up) **secp256k1** (< 5.0.1) Private key extraction over ECDH allowing credential compromise

4 - Vulnerable Third Party Libraries and Packages - **FIXED**	
	base-x (< 3.0.11) Homograph attack allowing Unicode lookalike characters to bypass validation **cross-spawn** (< 7.0.5) Regular Expression Denial of Service (ReDoS) vulnerability **glob** (< 10.5.0) Command injection via -c/--cmd flag executing matches with shell:true **axios** (multiple versions) < 0.30.0: SSRF and credential leakage via absolute URL < 0.30.2: DoS attack through lack of data size check **qs** (< 6.14.1) arrayLimit bypass in bracket notation allows DoS via memory exhaustion **Moderate Severity** **@cypress/request** (< 3.0.0) Server-Side Request Forgery (SSRF) vulnerability **next** (multiple versions) < 14.2.7: Denial of Service in image optimization < 14.2.21: Denial of Service with Server Actions < 14.2.31: Content injection for image optimization < 14.2.31: Cache key confusion for image optimization API routes < 14.2.32: Improper middleware redirect handling leads to SSRF **esbuild** (< 0.25.0) Allows any website to send requests to development server and read responses **postcss** (< 8.4.31) Line return parsing error **vite** (multiple versions) < 6.1.2: Bypasses server.fs.deny when using ?raw < 6.1.3: server.fs.deny bypassed for inline and raw with ?import query < 6.1.4: server.fs.deny bypass with .svg or relative paths < 6.1.5: server.fs.deny bypass with invalid request-target < 6.1.6: server.fs.deny bypassed with /. for files under project root < 6.4.1: server.fs.deny bypass via backslash on Windows **@babel/helpers** (< 7.26.10) Inefficient RegExp complexity in generated code with .replace when transpiling named capturing groups **nanoid** (< 3.3.8) Predictable results in ID generation when given non-integer values

4 - Vulnerable Third Party Libraries and Packages - **FIXED**	
	mdast-util-to-hast (< 13.2.1) Unsanitized class attribute leading to potential XSS **axios** (< 0.28.0) Cross-Site Request Forgery (CSRF) vulnerability **Low Severity** **next** (multiple versions) < 14.2.24: Race condition leading to cache poisoning < 14.2.30: Information exposure in dev server due to lack of origin verification **brace-expansion** (< 1.1.12 or < 2.0.2) Regular Expression Denial of Service vulnerability **vite** (multiple versions) < 6.3.6: Middleware may serve files starting with same name as public directory < 6.3.6: server.fs settings not applied to HTML files **tmp** (< 0.2.4) Arbitrary temporary file/directory write via symbolic link in dir parameter **elliptic** (< 6.6.0) Valid ECDSA signatures erroneously rejected
Proposed Fix	- Update all identified and vulnerable packages - Implement “yarn audit” into development processes and CI/CD pipeline - Enable Github Dependabot for automatic alerting and update PR creation

Tools

- Burp-suite Community Edition
- HOT wallet extension
- Semgrep
- Snyk
- Trufflehog
- Yarn