ZEALYNX

Your Web3 Security partner

Novaswap

Blackbox Pentesting

January 17, 2026

Prepared by Zealynx

contact@zealynx.io

Fernando @0xMrjory

Contents

1. About Zealynx

Zealynx, founded in January 2024 by Carlos (Bloqarl), specializes in smart contract audits, and development. Our services include comprehensive smart contract audits, application security audits, such as pentesting, and AI Audits. We are trusted by clients such as Badger DAO, Ample protocol, Lido, Inverter, Matchain, and Golden Grid.

Our team believes in transparency and actively contributes to the community by creating educational content. This includes challenges for Foundry and Rust, security tips for Solidity developers, and fuzzing materials like Foundry and Echidna/Medusa. We also publish articles on topics such as preparing for an audit and battle testing smart contracts on our website Zealynx.io and our blogs.

Zealynx has achieved public recognition, including winning 1st prize in the Uniswap Hook Hackathon and a Top 5 position in the Beanstalk Audit public contest. Our ongoing commitment is to provide value through our expertise and innovative security solutions for smart contracts.

2. Disclaimer

This penetration testing assessment was conducted within the defined scope and timeframe agreed upon with the client. While every effort was made to identify security vulnerabilities, this assessment cannot guarantee the discovery of all weaknesses or ensure complete security of the tested systems. The findings represent a point-in-time evaluation; subsequent changes to systems, configurations, or the threat landscape may introduce new vulnerabilities. The assessor assumes no liability for security incidents occurring after the assessment or for vulnerabilities outside the agreed scope. This report is confidential and intended solely for the authorized recipient. Testing was performed with proper authorization and in compliance with applicable laws and regulations.

3. Overview

3.1 Project Summary

A blackbox penetration testing assesment of the Novaswap frontend dApp and Mynth API endpoints was conducted by Zealynx Security with a focus on security assessment and risk identification.

We performed the security evaluation based on the agreed scope during 7 days. Following our systematic approach and methodologies, testing was performed on both the application surface and underlying infrastructure. Our security assessment revealed 9 issues, comprising 1 Medium, 4 Low severity, and 4 Informational findings.

3.2 About Novaswap

Novaswap is a non-custodial, web-based frontend that interfaces with the Mynth Protocol, facilitating cross-chain token swaps across multiple blockchain networks.

3.3 Audit Scope

The assessment covered a blackbox security evaluation of the Novaswap frontend application and four associated Mynth API endpoints:

• Frontend:
  ◦ https://www.novaswap.io
• API Endpoints:
 - POST https://www.mynth.ai/api/address/generate
 - GET https://www.mynth.ai/api/address/balance
 - GET https://www.mynth.ai/api/address/track
 - POST https://www.mynth.ai/api/address/cancel

4. Audit Methodology

Approach

During our security assessments, we uphold a rigorous approach to maintain **high-quality standards. Our methodology encompasses thorough TypeScript Code Review, Web Application Security Testing**, **Infrastructure Assessment**, and meticulous manual **penetration testing**.

Throughout the TypeScript application audit and penetration testing process, we prioritize the following aspects to uphold excellence:

1. **Front-End Security Review**: We comprehensively evaluate client-side code quality, security implementations, and potential attack vectors including XSS, CSRF, and clickjacking vulnerabilities.
2. **Back-End Architecture Analysis**: Our assessments emphasize secure coding practices, authentication mechanisms, authorization controls, session management, and data validation to ensure robust server-side security.
3. **Infrastructure and Configuration**: We meticulously review deployment configurations, environment variables, database connections, and third-party integrations to identify misconfigurations and security gaps.
4. **Penetration Testing**: We conduct both automated and manual testing against OWASP Top 10 vulnerabilities and custom threat scenarios, including real-world exploitation of identified weaknesses to demonstrate actual risk impact.

Testing Execution

During the Novaswap engagement, Zealynx performed the following actions to ensure thorough coverage of the defined scope:

- Confirmed the precise scope of systems, endpoints, and testing windows in coordination with Novaswap.
- Mapped the public attack surface and reviewed application flows to identify critical assets and business logic.
- Conducted automated vulnerability scanning on the frontend application and Mynth API endpoints, targeting OWASP Top 10 risks and common web vulnerabilities.
- Performed manual penetration testing to simulate real-world attacker behavior, including input fuzzing, parameter tampering, authentication and authorization bypass attempts, and abuse of rate-limiting and access controls.

- Submitted malformed and edge-case requests to validate server-side input handling, error management, and resilience to unexpected input.
- Reviewed infrastructure and deployment configuration, including WAF effectiveness and exposure of documentation or sensitive endpoints.
- Assessed API security, focusing on allowlisting, validation, and safe failure modes.

Validation Process

After initial testing and reporting, Zealynx followed a structured process to ensure the accuracy and effectiveness of remediation:

- Documented all findings with supporting evidence, impact analysis, and actionable remediation guidance.
- Provided detailed reports to Novaswap and communicated directly to clarify technical details and remediation steps.
- For each issue marked as “fixed” by Novaswap, re-tested the affected components to confirm successful mitigation and closure.
- Updated the final report to reflect the status of each finding, distinguishing between issues confirmed as resolved and those pending further action.

5. Severity Classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

6. Executive Summary

Over a 7-day engagement, the Zealynx Security team conducted a blackbox security assessment and penetration testing of Novaswap frontend and its associated API endpoints. The assessment was conducted from January 8-14, 2026.

The audit focused on identifying vulnerabilities, logic flaws, and implementation-level issues affecting the Novaswap decentralized token swapping platform.

A total of 9 issues were identified and categorized as follows:
• 1 Medium severity
• 4 Low severity
• 4 Informational severity

6.1 The Key Findings

• **Insufficient Entropy in Access Code–Based Access Control Mechanism:** The access code relies on limited character sets and length, making it vulnerable to brute-force attacks under realistic conditions.
• **Reusable Shared Access Code Allows Unlimited Unauthorized Access:** A single valid access code can be reused an unlimited number of times across multiple browser sessions, different browsers, and separate devices.
• **WAF Rate Limiting Applied Per Browser Session Rather Than Per Client:** The rate-limiting control is scoped to browser context rather than being enforced consistently at the client, IP, or network level.
• **Inactive QR Code Scanning Feature Exposed in Desktop Web Interface:** A QR code scanning feature intended for mobile environments is exposed in the desktop interface, causing potential user confusion.

6.2 Architectural Security Observations

During the black-box assessment, we observed several architectural decisions and controls that contribute to Novaswap's security posture:

• **Active Perimeter Protection:** Novaswap is protected by a Web Application Firewall (WAF) that actively enforces browser verification and rate limiting. This reduces exposure to automated abuse and opportunistic attacks.
• **Strict Server-Side Validation:** The Mynth API consistently enforces allowlisting and input validation for critical parameters. Invalid or malformed requests are rejected server-side, not just at the UI.
• **Predictable Failure Behavior:** Across tested endpoints, the API fails safely with structured JSON errors, reducing the risk of crashes, debug data leakage, or undefined application states.
• **API Abuse Resistance:** Rate-limit headers and enforcement at the API edge protect backend resources and enable responsible client behavior.
• **Controlled Exposure:** Access gating mechanisms are in place for staged rollouts, with documented improvements on code entropy and reuse.
• **Operational Security Maturity:** Multiple findings were remediated during the engagement, demonstrating a mature security feedback loop and rapid response capability.

6.3 Security Strengths Observed

Based on our assessment, Novaswap demonstrates the following security strengths:

• **Layered Defenses:** The combination of WAF, rate limiting, and server-side validation provides robust, multi-layered protection against common web and Web3-specific threats.
• **Defensive-by-Default Design:** The platform rejects unsafe inputs predictably and avoids processing malformed requests, supporting reliability and resilience.
• **No Critical Compromise Paths Identified:** The majority of findings were low or informational, focused on hardening and hygiene rather than systemic weaknesses or direct fund-loss paths.
• **Responsiveness to Security Feedback:** The client remediated several issues during the assessment, showing strong operational discipline and a commitment to continuous improvement.
• **Transparency in Reporting:** The security process included clear methodology, scope discipline, and a point-in-time disclaimer—key indicators of professional maturity.

Summary of Findings :

Vulnerability	Severity	Status
[M-01] WAF Rate Limiting Applied Per Browser Session Rather Than Per Client	Medium	Fixed
[L-01] Reflected Input in API Validation Error Messages	Low	Fixed
[L-02] Reflected Request Path and Query String in API "Route Not Found" Error Responses	Low	Fixed
[L-03] Reflected User Input in API Validation Error Messages (POST /api/address/generate)	Low	Fixed
[L-04] Malformed Validation Error Message	Low	Fixed
[I-01] Inactive QR Code Scanning Feature Exposed in Desktop Web Interface	Informational	Fixed
[I-02] Insufficient Entropy in Access Code–Based Access Control Mechanism	Informational	Fixed
[I-03] Reusable Shared Access Code Allows Unlimited Unauthorized Access Across Sessions and Devices	Informational	Fixed
[I-04] Publicly Accessible OpenAPI Specification	Informational	Fixed

7. Audit Findings

7.1 Medium Severity Findings

[M-01] WAF Rate Limiting Applied Per Browser Session Rather Than Per Client

Affected files

Vercel WAF

Description

The behavior described above is based on observed external responses and may reflect intentional design choices within the WAF configuration.

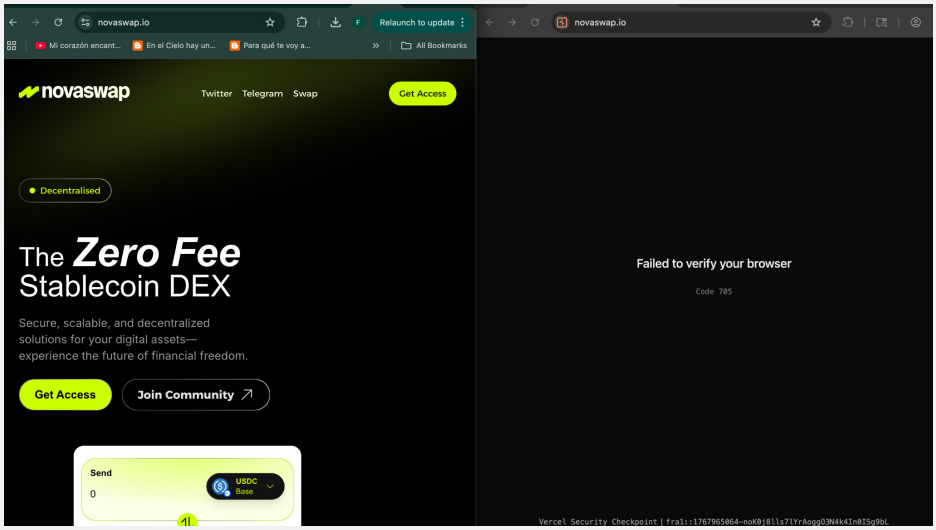
During testing, the application's Web Application Firewall (WAF) triggered a rate-limiting or verification block, returning a "Failed to verify your browser" message (Code 705) from the Vercel Security Checkpoint.

However, it was observed that the restriction appears to be scoped to the affected browser session only. When accessing the application from a different browser on the same machine and network, normal access was restored immediately, without additional verification or cooldown.

This behavior suggests that the rate-limiting or verification control is enforced per browser context (e.g., cookies, local storage, or browser fingerprint) rather than being consistently applied at the client, IP, or network level.

Vulnerable Scenario:

1. Trigger WAF protection through repeated requests or automated interaction
2. Observe browser-level block with message: "Failed to verify your browser (Code 705)"
3. Open a different browser on the same host
4. Navigate to the application
5. Observe normal access without WAF restriction



The block does not persist across browsers under the same client environment.

Impact

This behavior does not constitute a full WAF bypass, but it may reduce the effectiveness of rate-limiting and abuse-prevention controls by allowing an attacker to:

- Evade temporary rate limits by switching browsers
- Continue automated or semi-automated testing across multiple browser contexts
- Increase request volume without triggering sustained blocking
- Circumvent browser-scoped verification challenges

Recommendation

- Review Vercel WAF rate-limiting and challenge configuration
- Prefer enforcement based on:
  - IP address
  - ASN
  - Client fingerprint
  - Session + IP correlation

- Ensure verification challenges persist across browser contexts when appropriate
- Align WAF behavior with the intended threat model (abuse prevention vs. user friction)

Novaswap:

Confirmed

Zealynx:

Fixed

7.2 Low Severity Findings

[L-01] Reflected Input in API Validation Error Messages

Affected files

/api/address/balance*

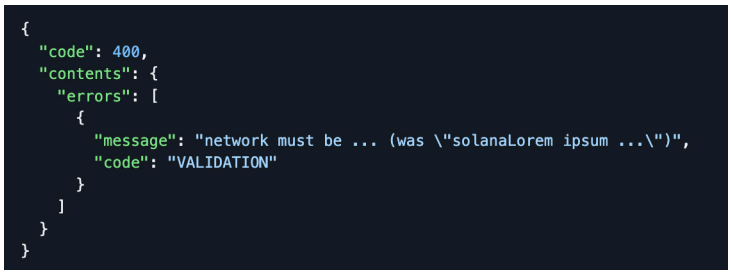
Description

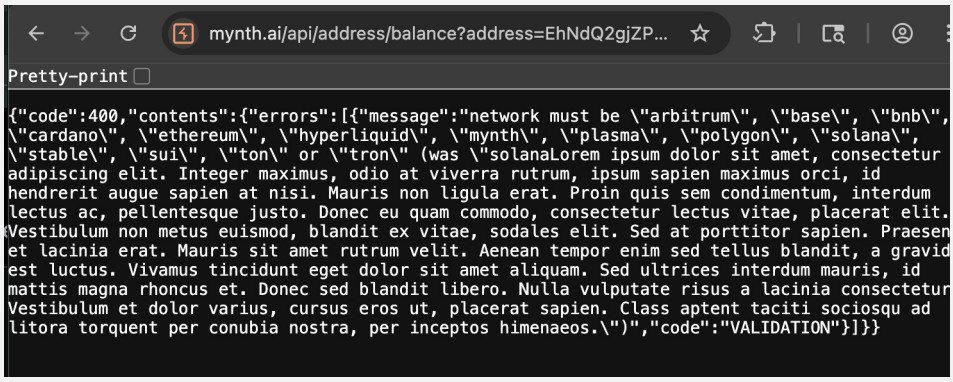
The endpoint /api/address/balance reflects untrusted user input from the network query parameter directly into the returned validation error message. When an invalid network value is provided, the API response includes the attacker-controlled value in contents.errors[].message without truncation.

While this behavior is common in validation frameworks, reflecting arbitrary input may increase risk if the message is subsequently rendered in a UI component, forwarded to logs/monitoring systems, or processed by downstream services that do not properly escape content.

Send a request with a valid address and an invalid network value containing attacker-controlled content:

GET /api/address/balance?address=<valid>&network=solanaLorem%20ipsum%20...
Host: www.mynth.ai
Origin: https://www.novaswap.io





Recommendation

- Avoid reflecting raw user input in error strings where possible
- If reflection is needed for debugging:
  - Truncate reflected values to a safe maximum length (e.g., 50–100 chars)
  - Normalize and escape untrusted content before including it in messages

Novaswap:

Confirmed

Zealynx:

Fixed

[L-02] Reflected Request Path and Query String in API "Route Not Found" Error Responses

Affected files

/api/address/generate*

Description

When requesting an invalid or non-existent API endpoint, the server returns a JSON error response that reflects the full requested route, including attacker-controlled query string parameters. This behavior was observed on the endpoint /api/address/generate, where the error message echoed back the full path:

Route GET: /api/address/generate?ael64smq30=1 not found

While "route not found" responses are expected, reflecting attacker-controlled input (the query string) in error messages can introduce downstream risk if these messages are displayed in a UI, logged verbatim, forwarded to monitoring/alerting systems, or rendered in contexts that may interpret special characters.

Request:
GET /api/address/generate?ael64smq30=1 HTTP/2
Host: www.mynth.ai

Response:
HTTP/2 404 Not Found
Content-Type: application/json; charset=utf-8
{"message":"Route GET:/api/address/generate?ael64smq30=1 not found","error":"Not Found","statusCode":404}

Recommendations

- Avoid reflecting full request paths and query strings in error messages returned to clients
- If reflection is required for debugging:
    - Truncate reflected content to a safe maximum length
    - Ensure proper escaping and normalization of any reflected input

Novaswap:

Confirmed

Zealynx:

Fixed

[L-03] Reflected User Input in API Validation Error Messages (POST /api/address/generate)

Affected files

api/address/generate

Description

The POST /api/address/generate endpoint reflects attacker-controlled request body values directly in validation error messages returned to the client. Invalid values supplied in fields such as source.blockchain, source.token, target.blockchain, and price are echoed verbatim in the error response.

Request:

```
{
  "price": "foo",
  "source": {
    "blockchain": "solanafoo",
    "token": "MynthbcejYyY3yg8fzgPv2Y7NZrF2S4Y13wdGzAZPMvfoo"
  },
  "target": {
    "address": "0x8fA08A667dDa20Ef8110CDfE5ba58eAfa4373b50",
    "blockchain": "basefoo",
    "token": "0x26D3C0D9F4cc4c130097b6aFdebe4F5e497E6Bdf"
  },
  "providerId": "novaswap"
}
```

Recommendations

Avoid reflecting raw user input in error messages

Novaswap:

Confirmed

Zealynx:

Fixed

[L-04] Malformed Validation Error Message

Affected files
/api/address/generate

Description
When the providerId field is manually provided as an empty string in POST /api/address/generate, the API returns a malformed validation error. While the feasibility of error is low, it might be useful to standarize an error handling logic for the API.

Request (providerId is empty):

{"price":"1","source":{"blockchain":"solana","token":"MynthbcejYyY3yg8fzgPv2Y7NZrF2S4Y13wdGzAZPMv"},"target":{"address":"0x8fA08A667dDa20Ef8110CDfE5ba58eAfa4373b50","blockchain":"base","token":"0x26D3C0D9F4cc4c130097b6aFdebe4F5e497E6Bdf"},"providerId":""}

Response:

"collected fees.;example:mynthswap (was \"\")"

Recommendations
Return concise, user-oriented validation messages (e.g., “providerId must be a non-empty string”)

Novaswap:
Confirmed

Zealynx:
Fixed

7.3 Informational Findings

[I-01] Inactive QR Code Scanning Feature Exposed in Desktop Web Interface

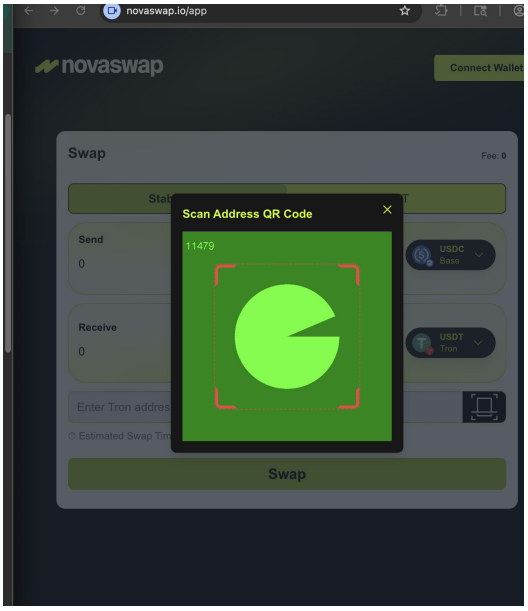
Affected files

Scan Address QR Code

Description

The web application exposes a “Scan Address QR Code” feature within the desktop web interface. When accessed from a desktop browser, the feature presents a camera scanning modal; however, the functionality is not operational in this context, as desktop browsers typically lack camera access or are not intended to use this feature.

Based on observed behavior, QR code scanning is intended for mobile environments, where the feature functions correctly. Its presence in the desktop web interface appears unnecessary and may cause confusion for
users.



Recommendations

Hide or disable the QR code scanning option when accessed from desktop browsers

Novaswap:

Confirmed

Zealynx:

Fixed

[I-02] Insufficient Entropy in Access Code–Based Access Control Mechanism

Affected files

Access-Page

Description

The application restricts access via an access code required at the /access-page endpoint. Based on observed behavior, the access code is a fixed-length, 5-character value. Two possible configurations were identified or considered:

Alphabetic characters only (A–Z)
Alphanumeric characters (A–Z, 0–9)
In both cases, the access code functions as a shared secret and serves as a gatekeeper to application functionality. Due to its limited length and character set, the resulting entropy is insufficient for a mechanism performing access control, rendering it vulnerable to guessing or brute-force attacks under realistic conditions.

Additionally, the access code is not bound to a specific user, wallet, session, or time window, further increasing the risk of unauthorized access if the code is disclosed, reused, or discovered.

Recommendations

- Increase access code length to at least 8 characters
- Use a larger character set (upper/lowercase letters and digits)
- Enforce strict rate limiting per IP and per session
- Introduce progressive delays or CAPTCHA after repeated failures
- Bind access tokens to a specific identity (wallet address, account, or email)

Novaswap:

Confirmed

Zealynx:

Fixed

[I-03] Reusable Shared Access Code Allows Unlimited Unauthorized Access Across Sessions and Devices

Affected files

Access-Page

Description

The application relies on a static access code (e.g., NVUMX) to grant access past the access gate endpoint. During testing, it was observed that a single valid access code can be reused an unlimited number of times, across:

- Multiple browser sessions
- Different browsers
- Separate devices (including mobile)

The access code is not bound to a specific user, wallet, session, IP address, or time window, and no apparent expiration or usage limit is enforced. As a result, the access code functions as a persistent shared secret, rather than a controlled access mechanism.

Recommendations

- Enforce single-use semantics for access codes
- Invalidate access codes after first successful use
- Introduce expiration timestamps (short-lived validity)
- Limit reuse across sessions and devices

Novaswap:

Confirmed

Zealynx:

Fixed

[I-04] Publicly Accessible OpenAPI Specification

Description

The application exposes its OpenAPI specification publicly via the endpoint /openapi.json. This document provides structured metadata describing the API, including endpoints, parameters, tags, and server configuration.

While OpenAPI documentation is commonly used for developers and it is not a security issue per se, making it publicly accessible might increase the attack surface by simplifying reconnaissance and automated testing.

Recommendations

Restrict access to /openapi.json to authenticated users, or to allow-listed IPs (e.g., internal/dev networks)

Novaswap:

Confirmed

Zealynx:

Fixed