ASSURE DEFI®

THE VERIFICATION GOLD STANDARD

SECURITY ASSESSMENT REPORT

NAME:	ZYGOSWAP - DAPP
STATUS:	FAIL
DATE:	12/02/2026

Risk Analysis

Vulnerability summary

Classification	Description
🔴 High	Vulnerabilities that lead to direct compromise of critical assets, large-scale data exposure, unauthorized fund transfers, or full system takeover.
🟠 Medium	Flaws that weaken security posture or privacy but do not immediately enable catastrophic failures.
🟡 Low	Issues that have minimal direct impact, often involving best-practice deviations or potential future risks.
🟢 Informational	Observations, style concerns, or suggestions that do not constitute vulnerabilities but may improve security hygiene.

SCOPE

Target Code And Revision

Project	ZYGO DEX
Codebase	https://github.com/goran9999/zygo-dex Commit [a6542bd6a0e7441d680d76a70f777b9b314637c2]
Audit Methodology	Static, Manual

Detailed Technical Report

1. Unbounded WebSocket “room” creation leads to memory exhaustion (remote DoS)

Location:

zygo-api/src/services/ws.rs:39–58, 116–137

Issue:

The server creates and stores a new RoomState (with a broadcast channel buffer of 1000) for any subscribe request where message is a new room name. Because room names are fully client-controlled and there are no global/per-connection limits or TTL eviction, an attacker can generate unlimited unique room names and force unbounded memory growth until the process is OOM-kille

Remediation:

Validate/allowlist room names (e.g., only known prefixes like transaction:<pool>, ohlcv:<pool>; reject arbitrary strings).

Enforce hard limits: max subscriptions per connection, max rooms globally, connection limits per IP.

Add eviction/TTL for idle/empty rooms.

Add WS handshake + per-message rate limiting and max frame/message size limits.

2. Malicious ERC20 decimals() can overflow conversions = crash or corrupted balance math

Location:

zygo-api/src/routers/wallet.rs:112–118, 187–197

(Also impacts swap paths via shared conversions) zygo-api/src/services/token.rs:31–47

Issue:

Balance/amount conversions compute powers of 10 using unbounded token decimals. A token can return extreme values (like, 255), causing overflow in exponentiation for example 10u128.pow(decimals) / similar) and leading to panics or invalid math. Additionally, parse_hex_balance() truncates balances that don't fit in u128 by returning 0, silently corrupting balance/value results.

Remediation:

Strictly bound-check decimals (reject or fail closed if decimals > 18 or a chosen safe upper bound like 36).

Replace lossy u128 parsing with U256 parsing end-to-end; only convert to display decimals safely (or return balances as strings).

On overflow/invalid decimals, return a per-token error instead of defaulting to 0.

3. Swap amount conversion overflow → incorrect quotes/tx params (integrity) + possible DoS

Location:

zygo-api/src/routers/swap.rs:33–38

zygo-api/src/services/token.rs:31–47

Issue:

decimal_to_u256() uses 10u64.pow(decimals) without validating decimals. For large decimals this can overflow/wrap, producing wrong amount_in (often 0 or nonsense). This corrupts swap quotes and any transaction-building that depends on these values, and can cascade into downstream service calls.

Remediation:

Apply the same decimals bounds as above at the token metadata boundary (get_token_decimals).

Use checked exponentiation / big-int math for scaling factors; if out of range, fail closed with an explicit error.

4. Overly permissive CORS (Any origin/method/header)

Location:

zygo-api/src/main.rs:64–68

Issue:

CORS is configured to allow any origin, method, and header. Even if you currently don't use cookies, this broad policy increases abuse surface (browser-based botnets). It becomes immediately dangerous if cookie/session auth or privileged endpoints are introduced later, because any site could call the API from a user's browser context.

Remediation:

Restrict allow_origin to the exact trusted frontend origins.

Allow only required methods/headers.

If cookies are ever used: enable credentials only with strict origin allowlist and add CSRF protections.

5. No WebSocket message size / rate limits = CPU/memory DoS risk

Location:

zygo-api/src/services/ws.rs:116–152

Issue:

There are no limits on message frequency or payload size. Attackers can spam subscribe/unsubscribe loops or send large frames, consuming CPU, spawning work, and stressing allocators/channels.

Remediation:

Enforce max frame size and max JSON payload size.

Apply per-connection and per-IP rate limiting.

Add strict schema validation for incoming WS messages.

6. Panic paths via unwrap() on env parsing / upstream data leading crash-based DoS

Location:

zygo-api/src/services/solana_swap.rs:56–60, 70–77 (and other unwrap() usages in that flow)

Issue:

unwrap() can panic on malformed environment values or unexpected upstream responses (DFlow/Solana RPC). Since these sources are not fully under your control, a bad response (or misconfig) can crash the service

Remediation:

Replace unwrap() with error propagation and safe handling (return 4xx/5xx without crashing).

Validate upstream fields before indexing/using them; treat RPC/aggregator responses as untrusted input.

7. Wallet-signature auth replayable within time window (no server nonce)

Location:

zygo-api/src/services/user.rs:23–83

Issue:

The auth flow accepts a signed message containing a client-provided timestamp and allows +60 seconds. If a signature is captured (like client compromise, log leakage, or any non-TLS hop), it can be replayed within the allowed window because there's no server-issued, one-time nonce.

Remediation:

Introduce a server-issued nonce stored server-side (for ex, Redis) and enforce one-time use.

Use SIWE-style structured messages including domain/uri/chainId/nonce/issuedAt to reduce cross-context replay/phishing risk.

Ensure signatures and full auth payloads are never logged.

MEDIUM

1. Containers likely run as root (Docker hardening missing)

Location:

Dockerfile.api

Dockerfile.indexer

Issue:

The Dockerfiles don’t show non-root execution or common runtime hardening. If the process is compromised, root-in-container increases blast radius, especially with writable filesystem or mounted secrets.

Remediation:

Add a non-root user and USER directive.

Use read-only root filesystem where possible; drop Linux capabilities; set no-new-privileges.

Prefer minimal/distroless runtime images when feasible.

LOW

1. Insecure dev defaults in mobile env example (http:// / ws://)

Location:

zygo-mobile/.env.example

Issue:

Example configuration uses plaintext HTTP/WS endpoints. This is fine for local development, but production must enforce TLS.

Remediation:

Ensure production configs only permit https:// and wss://.

Consider failing startup if a non-TLS endpoint is configured outside dev.

2. Security headers not enforced (API + web app hardening)

Location:

API: (no explicit header middleware found in current review)

Web: Next.js config (not visibly enforcing headers in reviewed paths)

Issue:

Missing or unenforced headers (HSTS when TLS-terminated, X-Content-Type-Options, Referrer-Policy, CSP) increases exposure to common web attack classes, especially if third-party scripts exist.

Remediation:

Add standard security headers at the edge/API gateway or app middleware.

For web app: enforce a CSP tailored to your actual script/style sources.

INFORMATIONAL

No informational issues were found.

Technical Findings Summary

Findings

Vulnerability Level	Total	Pending	Not Apply	Acknowledged	Partially Fixed	Fixed
High	7					
Medium	1					
Low	2					
Informational						

Assessment Results

Score Results

Review	Score
Global Score	**60/100**
Assure KYC	Not completed
Audit Score	60/100

The Following Score System Has been Added to this page to help understand the value of the audit, the maximum score is 100, however to attain that value the project must pass and provide all the data needed for the assessment. Our Passing Score has been changed to 84 Points for a higher standard, if a project does not attain 85% is an automatic failure. Read our notes and final assessment below. The Global Score is a combination of the evaluations obtained between having or not having KYC and the type of contract audited together with its manual audit.

Audit FAIL

Following our comprehensive security audit of the backend/frontend for the **ZYGADEX DAPP** project, the project did not fulfill the necessary criteria required to pass the security audit.

Disclaimer

Assure Defi has conducted an independent security assessment to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the reviewed code for the scope of this assessment. This report does not constitute agreement, acceptance, or advocating for the Project, and users relying on this report should not consider this as having any merit for financial Token in any shape, form, or nature. The contracts audited do not account for any economic developments that the Project in question may pursue, and the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude, and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are entirely free of exploits, bugs, vulnerabilities or deprecation of technologies. All information provided in this report does not constitute financial or investment in Token, nor should it be used to signal that any person reading this report should invest their funds without sufficient individual due diligence, regardless of the findings presented. Information is provided ‘as is, and Assure Defi is under no covenant to audit completeness, accuracy, or solidity of the contracts. In no event will Assure Defi or its partners, employees, agents, or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions or actions with regards to the information provided in this audit report. The assessment of Tokens provided by Assure Defi are subject to dependencies and are under continuing development. You agree that your access or use, including but not limited to any Tokens, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies with high levels of technical risk and uncertainty. The assessment reports could include false positives, negatives, and unpredictable results. The Token may access, and depend upon, multiple layers of third parties.