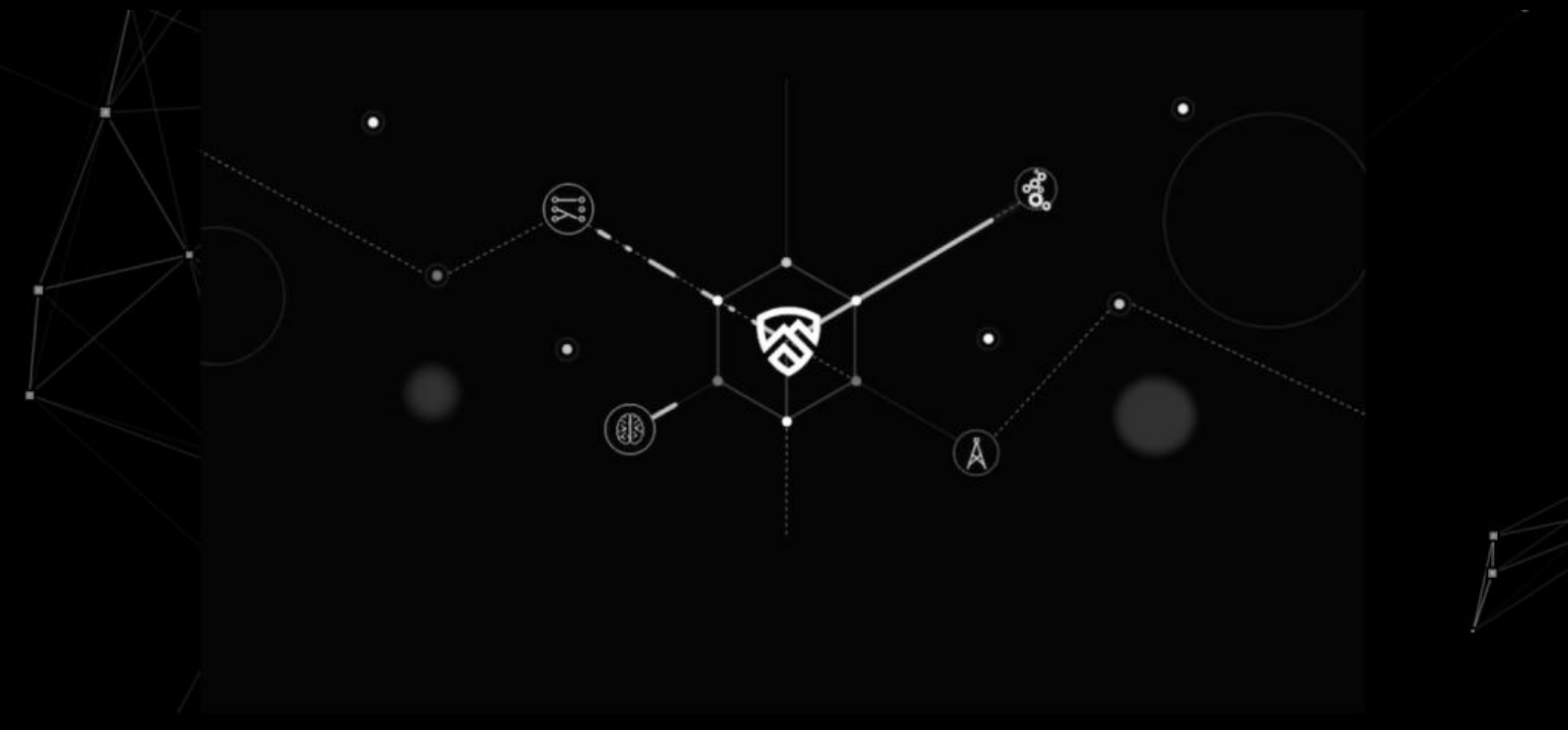
AEGIXE

Stock Security Assessment

Aegixe Assessed on April 27, 2026



DeShare

Aegixe Assessed on April 27, 2026

Stock

The security assessment was prepared by Aegixe

Executive Summary

TYPES	ECOSYSTEM	LANGUAGE	TIMELINE
A DeFi staking and settlement system deployed on Base	EVM Compatible	Solidity	inal report published on 08/05/2026

Vulnerability Summary

2	0	0	1	1
Total Findings	Critical	High	Medium	Low

Critical	Direct loss of user or protocol funds; exploitable by any user without special privileges
High	Significant loss of funds or severe protocol disruption; may require specific conditions
Medium	The order is represented solely by events, with no verifiable state on the chain.
Low	Centralization risk caused by excessively high administrator permissions.

Stock Contract Formal Audit Report

Project: Stock Token Factory & Stock Trading Contracts
Auditing Firm: Aegixe
Audit Date: 2026-04-27

1. Executive Summary

This review covered `StockTokenFactory` and `StockTrading`, together with their `StandardStockToken` implementations. Overall, the current source code does **not** contain a confirmed High- or Critical-severity composite vulnerability that can be triggered independently by an external attacker. The remaining issue to pay attention to is that the order lifecycle is expressed only through events, with no verifiable on-chain order state, which is a Medium-severity design risk.

2. Scope

- `StockTokenFactory/StockTokenFactory.sol`
- `StockTrading/StockTrading.sol`
- `StockTokenFactory/StandardStockToken.sol`
- `StockTrading/StandardStockToken.sol`

3. Overall Conclusion

1. The system has a strongly centralized control plane. The owner can modify fund receiver addresses, the USDT address, the factory address, fee settings, the minimum fee, and can also perform minting, burning, withdrawals, and upgrade-related actions.
2. The most obvious issue is the lack of an on-chain order state, which creates reconciliation and risk-control issues.
3. No High- or Critical-severity vulnerability has been confirmed.

4. Finding Summary

ID	Severity	Affected Area	Summary
MED-01	Medium	`StockTrading`	Orders are event-only and have no verifiable on-chain state
LOW-01	Low	`StockTokenFactory` / `StockTrading`	Misuse of admin-controlled settings and configuration switching can cause fund misrouting, failed refunds, or unbalanced supply control

5. Detailed Findings

MED-01 Orders are event-only and have no verifiable on-chain state

Affected code: `StockTrading/StockTrading.sol:177-358`
Severity: Medium

`createBuyOrder()` and `createSellOrder()` only emit `OrderCreated` events and do not store any order state on chain. `markOrderFilled()` and `markOrderCancelled()` also do not verify whether an

order really exists, whether it has already been completed, whether it has expired, or whether it matches the original order parameters. In short, the chain records the money flow and event flow, but does not record a verifiable order lifecycle.

This design does not allow a user to complete an incorrect refund or an incorrect fill by itself, but it will significantly increase centralized risk.

Recommended fix:

- Introduce an on-chain `struct` and status flags for orders.
- Snapshot the user, asset address, quantity, price, fee rate, margin, interest, expiration time, and status fields when placing an order.
- Validate state transitions and synchronize the status during fills and cancellations.

LOW-01 Misuse of admin-controlled settings and configuration switching can cause fund misrouting, failed refunds, or unbalanced supply control

Affected code: `StockTokenFactory/StockTokenFactory.sol:21-39`, `StockTrading/StockTrading.sol:84-175`, `StockTrading/StockTrading.sol:361-443`
Severity: Low

The following functions are privileged or operational entry points: `initialize()`, `_authorizeUpgrade()`, `setReceivers()`, `setFeeRate()`, `setAssetFeeRate()`, `setMinFeeAmount()`, `setUsdtContract()`, `setStockTokenFactory()`, `mintStockTokens()`, `burnStockTokens()`, `markOrderFilled()`, `markOrderCancelled()`, `withdrawFees()`, `setTradingContract()`, and `createStockToken()`.

These functions share a common property: if an administrator makes a mistake, if an address is configured incorrectly, or if an upgrade path is switched incorrectly, the system may experience misrouted fund addresses, failed historical refunds, incorrect USDT recognition, mismatched stock-token ownership, abnormal fee rates, uncontrolled minting/burning, or withdrawal accounting errors. Because the basis for these issues is an administrator-side failure rather than a direct source-level exploit by an external attacker, they are rated Low, although sensitive operations still require caution.

Recommended fix:

- Move critical parameters behind multisig or timelock governance.
- Add event logging, version tracking, and change-confirmation workflows for the USDT address, factory address, and fund receiver addresses.
- Avoid relying on mutable receivers for order-related funds; prefer contract custody or an immutable vault.
- Apply stricter audit and approval workflows to minting, burning, withdrawals, and upgrade permissions.

AEGIXE Chain Shield

safeguarding web3security and protecting digital assets

Founded in 2019, Aegixe is a company focused on Web3 security. It was co-founded by cybersecurity engineers from Israel and Hong Kong with extensive experience in on-chain attack and defense, cryptographic protocols, and cross-chain security. Team members have participated in the analysis and vulnerability disclosure of numerous globally renowned security incidents and possess rich experience in auditing and conducting attack and defense operations on large-scale Web3 systems.

Aegixe currently serves hundreds of projects across various sectors, including DeFi, GameFi, RWA, L1/L2 public chains, cross-chain bridges, and payment protocols, providing the rapidly evolving Web3 industry with faster detection speeds, broader coverage, and more intelligent security capabilities.