ALEX Audit

Orderbook

October 2022

By CoinFabrik

v202209

Executive Summary

CoinFabrik was asked to audit the contracts for Alex's Orderbook project. The audited files are from the git repository located at github.com/alexgo-io/alex-v2-orderbook. The audit is based on the commit `fef1e56765300690218602cd4fe698e6e0db8280`. Fixes were reviewed on commit `ef0260e8900cd12ccb9bcc28e523c46baf08e0bf`.

The scope for this audit includes and is limited to the following files:

- `contracts/redstone-verify.clar`: A stateless library contract that other contracts can call into to verify `RedStone` messages.
- `contracts/stxdx-exchange-zero.clar`: Exchange contract where orders
- are matched.
- `contracts/stxdx-exchange-perp.clar`: Exchange contract where orders
- are matched (for perpetual orders).
- `contracts/stxdx-registry.clar`: Users, assets, order fills and approvals
- are registered in this contract.
- `contracts/stxdx-sender-proxy.clar`: Proxy for stxdx-exchange-zero.
- `contracts/stxdx-utils.clar`: Utility functions for the system.
- `contracts/stxdx-wallet-zero.clar`: Wallet where users store their
- assets to operate with the exchange.

No other files in this repository were audited. Its dependencies are assumed to work according to their documentation. Also, no tests were reviewed for this audit.

During this audit, we found three minor issues. Enhancements were not proposed.

Two issues were fixed and one mitigated by the development team.

Methodology

CoinFabrik was provided with the source code, including automated tests that define the expected behavior, and general documentation about the project. Our auditors spent two weeks auditing the source code provided, which includes understanding the context of use, analyzing the boundaries of the expected behavior of each contract and function, understanding the implementation by the development team (including dependencies beyond the scope to be audited) and identifying possible situations in which the code allows the caller to reach a state that exposes some vulnerability. Without being limited to them, the audit process included the following analyses:

- Arithmetic errors
- Race conditions
- Misuse of block timestamps
- Denial of service attacks
- Excessive gas usage
- Missing or misused function qualifiers
- Needlessly complex code and contract interactions
- Poor or nonexistent error handling
- Insufficient validation of the input parameters
- Incorrect handling of cryptographic signatures
- Centralization and upgradeability

After delivering a report with our findings, the development team had the opportunity to comment on every finding and fix the issues they considered convenient. Once fixed and/or commented, our team ran a second review process to verify that the changes to the code effectively solve the issues found and do not unintentionally add new ones. This report includes the final status after the second review.

Findings

In the following table we summarize the security issues we found in this audit. The severity classification criteria and the status meaning are explained below. This table does not include the enhancements we suggest to implement, which are described in a specific section after the security issues.

ID	Title	Severity	Status
MI-01	Request Grace Period Not Constrained	Minor	Resolved
MI-02	Insecure Authentication Through tx-sender	Minor	Mitigated
MI-03	Missing Validation For Left-sided Order	Minor	Resolved

Severity Classification

Security risks are classified as follows:

- **Critical:** These are issues that we manage to exploit. They compromise the system seriously. They must be fixed **immediately**.
- **Medium:** These are potentially exploitable issues. Even though we did not manage to exploit them or their impact is not clear, they might represent a security risk in the near future. We suggest fixing them **as soon as possible**.
- **Minor:** These issues represent problems that are relatively small or difficult to take advantage of, but might be exploited in combination with other issues. These kinds of issues do not block deployments in production environments. They should be taken into account and be fixed **when possible**.

Issues Status

An issue detected by this audit has one of the following statuses:

- **Unresolved**: The issue has not been resolved.
- **Acknowledged**: The issue remains in the code, but is a result of an intentional decision.
- **Resolved**: Adjusted program implementation to eliminate the risk.
- **Partially resolved**: Adjusted program implementation to eliminate part of the risk. The other part remains in the code, but is a result of an intentional decision.

- **Mitigated**: Implemented actions to minimize the impact or likelihood of the risk.

Critical Severity Issues

No issues found.

Medium Severity Issues

No issues found.

Minor Severity Issues

MI-01 Request Grace Period Not Constrained

Location:

- `contracts/stxdx-wallet-zero.clar:41-46`

Request grace period defines how much time a user should wait to execute a transfer out of the wallet, when it does not get an approval. This period is initially set to 100 blocks, and the owner can change it to any number. The owner might set a big number in order to freeze users' funds in the wallet.

Recommendation

Add an assertion for the new period value restricting the maximum grace period length so that the owner cannot freeze users funds.

Status

Resolved. Maximum grace period was set to `1008` blocks, which is seven days based on Bitcoin/Stacks block-time (~10 minutes).

MI-02 Insecure Authentication Through tx-sender

Location:

- `contracts/stxdx-exchange-perp.clar`
- `contracts/stxdx-exchange-zero.clar`
- `contracts/stxdx-registry.clar`
- `contracts/stxdx-sender-proxy.clar`
- `contracts/stxdx-utils.clar`
- `contracts/stxdx-wallet-zero.clar`

Using `tx-sender` for authentication is not secure. Actors in the system could be targeted for phishing.

This issue was found in:

- `stxdx-sender-proxy::is-contract-owner()`,
- `stxdx-exchange-perp::is-contract-owner()`,
- `stxdx-exchange-perp::validate-authorisation()`,
- `stxdx-exchange-perp::approve-order()`,
- `stxdx-exchange-zero::is-contract-owner()`,
- `stxdx-exchange-zero::validate-authorisation()`,
- `stxdx-exchange-zero::approve-order()`,
- `stxdx-registry::set-contract-owner()`,
- `stxdx-registry::is-contract-owner()`,
- `stxdx-registry::set-order-approval()`,
- `stxdx-wallet-zero::is-contract-owner()`,
- `stxdx-wallet-zero::is-authorised-approver()`,
- `stxdx-wallet-zero::request-transfer-out()`.

Some functions that involve asset transfers cannot be called in the attack, thanks to post-conditions. Also, owner authentication will be less likely to be targeted once the DAO is set as owner.

Recommendation
Prefer `contract-caller` to `tx-sender` for authentication, unless it is specifically required and the risk is considered. Also, adding a mapping for trusted callers might be helpful if intermediary contracts are needed.

Status
Mitigated. All these contracts will be owned by a DAO. For other cases, given that post-conditions can prevent asset transfers triggered by malicious actors through tx-sender, the development team is in favour of keeping tx-sender in place for composability.

MI-03 Missing Validation For Left-sided Order

Location:

- `contracts/stxdx-exchange-perp.clar:446-471`

When an order match is validated in the perpetual exchange contract, both orders are checked in order to verify the data from the linked orders. However, one of the checks is not made on the linked order on the left, while it is made on the right side of the match.

On the right side, these checks are located at lines 507 and 520. This should also be placed on lines 456 and 468.

Recommendation
Add the missing validation on the left-sided linked order.

Status

Resolved. Validations added.

Other Considerations

The considerations stated in this section are not right or wrong. We do not suggest any action to fix them. But we consider that they may be of interest to other stakeholders of the project, including users of the audited contracts, token holders or project investors.

Centralization

Contract owner defines system parameters and assigns privileged roles.

From the privileged roles, the approved exchange in `stxdx-wallet-zero` can transfer on behalf of the user.

Upgrades

Contracts do not implement mechanisms for future upgrades.

Privileged Roles

These are the privileged roles that we identified on each of the audited contracts.

Exchange Contract and Proxy

Files: stxdx-exchange-perp.clar, stxdx-exchange-zero.clar and stxdx-sender-proxy.clar

Authorized sender

Whitelisted address allowed to cancel (only for exchanges) and match orders.

Wallet

Files: stxdx-wallet-zero.clar

Authorized approver

Address allowed to approve transfers out of the wallet.

Authorized exchange

This role can reassign assets from one user to another registered in the wallet.

Changelog

- 2022-10-20 – Initial report based on commit `fef1e56765300690218602cd4fe698e6e0db8280`.
- 2022-10-26 – Final report based on commit `ef0260e8900cd12ccb9bcc28e523c46baf08e0bf`.

Disclaimer: This audit report is not a security warranty, investment advice, or an approval of the Alex project since CoinFabrik has not reviewed its platform. Moreover, it does not provide a smart contract code faultlessness guarantee.