LISA Audit

DAO and Liquid Stacking

April 2024

By CoinFabrik

v202404

Executive Summary

CoinFabrik was asked to audit the contracts for the LISA project, a liquid stacking project on Stacks.

During this audit we found one high issue and one minor issue.

The high issue was acknowledged and the minor issue was partially resolved.

Scope

The audited files are from the git repository located at https://github.com/lisalab-io/liquid-stacking. The audit is based on the commit `0f56793ebc8d20a06aee53de94a507f2bdbb7ac3`. Fixes checked on commit `885af2c533994e9aa5a143d8980ddc2ae9fdeb7b`.

The scope for this audit includes and is limited to the following 11 files:

- `contracts/lisa-dao.clar`: Core contract for executing proposals and bootstrapping the LISA DAO.
- `contracts/token-lqstx.clar`: Contract for lqSTX, the rebase token for STX liquid stacking.
- `contracts/token-vlqstx.clar`: Contract for wrapping lqSTX rebase token into a non-rebasing one called vlqSTX, enhancing interoperability with existing DeFi protocols.
- `contracts/aux/lqstx-mint-registry.clar`: Serves as a registry for lqSTX minting and burning activities such as tracking request statuses, mint pending amounts and nonces. It acts as the data counterpart for lqSTX mint/burn operations.
- `contracts/extensions/lqstx-mint-endpoint.clar`: Operational interface for minting and burning lqSTX, serving as portal for LISA users to convert STX into the lqSTX rebasing token and vice versa. This contract executes minting/burning logic while relying on the aforementioned registry to store and retrieve data.
- `contracts/extensions/lqstx-vault.clar`: Vault that holds the STX of the members, allowing the exchange of STX with strategies contracts for stacking (fund) and retrieving back funds (refund).
- `contracts/extensions/operators.clar`: Proposal voting extension. Allowed operators and voting threshold are settled here. Proposals are automatically executed once the specified voting threshold is reached.

- `contracts/extensions/public-pools-strategy-manager.clar`: Manages the funding and refunding of the public pools strategy via authorized accounts, serving as the administrative layer for managing public pools operations.
- `contracts/extensions/token-vesting.clar`: Vesting contract for LISA token.
- `contracts/extensions/treasury.clar`: Treasury for holding and transferring STX and SIP-010 tokens.
- `strategies/public-pools/public-pools-strategy.clar`: Strategy contract for public stacking pools. Holds the logic for funding/refunding the pools via the vault and is able to aggregate and report the total amount of pooled STX (locked and unlocked).

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.

Methodology

CoinFabrik was provided with the source code, including automated tests that define the expected behavior. 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
HI-01	Proposal Voting Bias due to Race Condition	High	Acknowledged
MI-01	Misimplementation of SIP-010 Fungible Token Standard	Minor	Partially Resolved

Severity Classification

Security risks are classified as follows:

- **Critical:** These are issues that we manage to exploit. They compromise the system seriously. Blocking bugs are also included in this category. They must be fixed **immediately**.
- **High:** These refer to a vulnerability that, if exploited, could have a substantial impact, but requires a more extensive setup or effort compared to critical issues. These pose a significant risk and **demand immediate attention**.
- **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. The reported risk is accepted by the development team.
- **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.

High Severity Issues

HI-01 Proposal Voting Bias due to Race Condition

Location:

- `contracts/extensions/operators.clar: 43-66`

Proposals are voted and executed through the `operators.clar` contract. The requirements for a proposal to be executed are:

- The difference between positive and negative votes (signals) must be equal or higher than the `proposal-threshold`.
- Proposal must be in a valid time frame given by `proposal-validity-period`.
- Neither the voting threshold nor the authorized operators should have been changed after proposal's creation (`proposed-at` greater than `operators-update-height`).

Since there is no minimum timeframe nor minimum number of votes to execute the proposal, a race condition can happen for executing calls to `signal` function.

Suppose a simple scenario were `proposal-threshold` is 3 and there are a total of 10 operators, half of them are in favour and half of them are against. Here overall count is zero, but the actual result depends on the transaction execution order. Operators in favor could try to front-run negative votes or node operators could order the transactions if they are interested in a specific outcome. Note once the threshold is met, proposal execution is within the same `signal` call.

Recommendation

Add a minimum waiting period since proposal creation or require a minimum amount of total votes before proposals can be executed.

Status

Acknowledged. The development team stated this is an intentional decision.

Medium Severity Issues

No issues found.

Minor Severity Issues

MI-01 Misimplementation of SIP-010 Fungible Token Standard

Location:

- `contracts/token-lqstx.clar: 116`
- `contracts/token-vlqstx.clar: 42`
- `contracts/extensions/treasury.clar: 5, 23`
- `contracts/traits/sip-010-transferable-trait.clar` (out of this audit scope)
- `contracts/token-lisa.clar` (out of this audit scope)

The fungible tokens within the LISA protocol (lqSTX, vlqSTX and LISA) utilize a `transfer` function with the signature

`(transfer (uint principal principal (optional (buff 2048))) (response bool uint))`.

This signature deviates from the SIP-010 Fungible Token Standard, which specifies the last parameter (typically named `memo`) as `(optional (buff 34))`[1]. While the implemented signature technically adheres to the lower bounds set by the SIP-010 Standard Trait[2], it is not in complete alignment with it. This discrepancy can lead to unexpected behavior, especially when operating with external DeFi protocols that expect a memo field of 34 bytes.

The treasury extension contract, which consists of three governance functions, is also affected by this issue:

- `stx-stransfer:` Executes STX transfers from the treasury.
- `sip010-transfer`: Intends to manage transfers of tokens compliant with the SIP-010 Fungible Token Standard from the treasury.
- `proxy-call`: Handles special ad-hoc situations without overloading the treasury functionality.

The bug appears in `treasury::sip010-transfer` function, which takes as parameter the `sip-010-transferable-trait` defined with the aforementioned function signature, setting a lower bound of `(optional (buff 2048))` for the last `transfer` parameter. This setting results in a mismatch, as `sip010-transfer` might fail when dealing with tokens

[1] https://github.com/stacksgov/sips/blob/main/sips/sip-010/sip-010-fungible-token-standard.md
[2] SP3FBR2AGK5H9QBDH3EEN6DF8EK8JY7RX8QJ5SVTE.sip-010-trait-ft-standard.sip-010-trait

that adhere strictly to the SIP-010 Standard Trait with a memo field less than 2048 bytes. Although the three LISA protocol tokens comply with this `sip-010-transferable-trait`, and thus work with the `treasury::sip010-transfer` function, this does not hold universally for all tokens following the SIP-010 Standard.

Recommendation

Make LISA tokens fully compatible with SIP-010 Fungible Token Standard to avoid compatibility issues. Adding an `impl-trait` line at the beginning of each contract is strongly recommended.

Additionally, modify `sip-010-transferable-trait` to be a true subset of SIP-010 Standard transfer functionalities, ensuring that every token adhering to SIP-010 Standard Trait will implicitly adhere to SIP-010 Transferable Trait.

Status

Partially Resolved. The treasury contract was updated in order to support SIP-010 trait instead of `sip-010-transferable-trait`. Transferable trait was removed. Tokens were not modified since they are already deployed.

Enhancements

No enhancements are proposed.

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

LISA DAO

The `lisa-dao.clar` contract is responsible for initializing the DAO, enabling/disabling extensions and executing proposals. Governance functions throughout the audited contracts are guarded by `is-dao-or-extension` function.

Key observations:

- The proposal executed in the construct must set one or more extensions; otherwise, the DAO becomes unusable.
- Extensions hold a lot of power. They can set new extensions (without needing to do so through a proposal) and can also execute proposals by calling `execute`. There is total trust in the extensions. They should be thoroughly reviewed when voted.

- As of this audit, the only contract capable of executing proposals via the DAO is the `operators.clar` contract.

Mint endpoint

Governance functions of the endpoint consists of the following.

- `set-paused`

The functionality of the endpoint as a whole can be paused, since the core public functions (`require-mint`, `require-burn`, `revoke-mint`, `revoke-burn`, `finalize-mint` and `finalize-burn`) are pausable. This could imply users STX amount freezed due to pending mints and the impossibility to get stacked funds back. On the other hand, this feature can be useful in a threatening situation.

- `set-mint-delay`

Users mint requests of lqSTX token have a mint delay. Requestors' STX are held by the vault from request until mint can be finalized (when the waiting period ends). This delay is controlled by the global variable `mint-delay` which is unbounded and can be arbitrarily set. Note this affects the current pending mints request; however, they can `revoke-mint` and get STX back.

- `set-use-whitelist, set-whitelisted-many, set-whitelisted`

When whitelisting is enabled (`use-whitelisted` set to `true`), only those principals registered in the whitelist can call the `request-mint` function.

Upgrades

All extensions within the LISA DAO can be upgraded via proposals.

Mint endpoint

`lqstx-mint-endpoint.clar` contract serves as the operational interface for minting and burning lqSTX and it is a fundamental part This audit focuses on the version `lqstx-mint-endpoint-v1-02`.

Currently, there's only one strategy contract, the `public-pools-strategy.clar`. As it stands, the mint endpoint is designed to handle exclusively this strategy. This can be seen on how `rebase` is built; it takes the total deployed amount of STX from the total amount in the public pools strategy. However, there is defined an unused private function called `sum-strategy-amounts` suggesting that multi-strategy logic will be available on an upcoming version.

Different Decimals for lqSTX and vlqSTX

The `token-decimals` variable is set independently in each of the `token-lqstx.clar` and `token-vlqstx.clar` contracts. This does not represent a functional issue per se, as the conversion between shares and tokens does not rely on decimals and units remain consistent all along the way. However, differing `token-decimals` could lead to wallets displaying confusing values for lqSTX and vlqSTX in users' balances. It is recommended that, in the event of a change in decimals, both variables should be updated within the same transaction for consistency.

Updating Operators Invalidates Ongoing Proposals

Calling `set-operators` (changes operators) and `set-proposal-threshold` (modifies threshold) invalidates current proposals. However, these actions can only be executed through the DAO and its extensions. Moreover, proposals can be resubmitted if they are invalidated in this manner.

Treasury SIP-010 Transfers Works for Tokens Authenticating via contract-caller

The treasury's `sip010-transfer` function does not employ `as-contract` to wrap the transfer call. Consequently, attempts to transfer tokens that authenticate solely via `tx-sender` will fail. However, this limitation does not result in a freeze of funds; in such cases, transfers can still be executed through `treasury::proxy-call`.

Changelog

- 2024-04-12 – Initial report based on commit `0f56793ebc8d20a06aee53de94a507f2bdbb7ac3`.
- 2024-04-18 – Final report based on commit `885af2c533994e9aa5a143d8980ddc2ae9fdeb7b`.