CoinFabrik

Security Audit Report

ALEX - Self-Listing Pool

February 2025

V202411

Executive Summary 3
Scope 3
Findings 3
Critical Severity Issues 4
High Severity Issues 4
Medium Severity Issues 4
Low Severity Issues 4
LO-01 Unvalidated Lock Parameter Values 4
LO-02 Empty Initial Token Template 5
LO-03 Limited Parameter Validation 5
LO-04 Duplicated Error Codes 6
Enhancements 6
EN-01 Limited Event Logging 7
EN-02 Remove Debug Mode Functionality 7
Other Considerations 8
Centralization 8
Upgrades 8
About CoinFabrik 8
Methodology 8
Severity Classification 10
Issue Status 11
Disclaimer 11
Changelog 12

Executive Summary

CoinFabrik was asked to audit the contracts for **ALEX's Self-Listing Pool** project.

During this audit we found four low severity issues. Also, two enhancements were proposed.

Two issues were resolved, one mitigated and one acknowledged. Both enhancements were implemented.

Scope

The audited files are from the git repository located at https://github.com/alexgo-io/alex-dao-2, in the `./contracts/` directory. The audit is based on the commit `206981bd24497e54730870c1afbf7fb4e5227c59`. Fixes were checked on commit `7d8ebf4291ef49a2889b9eb66904f37867a8a17e`.

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

- `./aux/clarity-stacks/clarity-stacks.clar`: verifies transaction inclusions in the blockchain.
- `./aux/clarity-stacks/clarity-stacks-helper.clar`: helper for `clarity-stacks`.
- `./aux/clarity-stacks/code-body-prover.clar`: verifies contract deployments.
- `./aux/wrapped-token.clar`: wrapped token implementation that interfaces with the underlying token-alex.
- `./extensions/self-listing-helper-v3.clar`: enables permissionless creation of AMM pools.

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.

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.

Each severity label is detailed in the Severity Classification section. Additionally, the statuses are explained in the Issues Status section.

Id	Title	Severity	Status
LO-01	Unvalidated Lock Parameter Values	Low	**Resolved**
LO-02	Empty Initial Token Template	Low	**Acknowledged**
LO-03	Limited Parameter Validation	Low	**Mitigated**
LO-04	Duplicated Error Codes	Low	**Resolved**

Critical Severity Issues

No issues found.

High Severity Issues

No issues found.

Medium Severity Issues

No issues found.

Low Severity Issues

LO-01 Unvalidated Lock Parameter Values

Location

- `./extensions/self-listing-helper-v3.clar: 243-244`

Classification

- CWE-20: Improper Input Validation[1].

Description

The contract processes the `lock` parameter in pool creation without validating that it matches one of the expected constants (`LOCK` or `BURN`).

[1] https://cwe.mitre.org/data/definitions/20.html

If a value other than the defined constants is provided, no locking or burning action occurs. This may be intentional design to allow pools without locked liquidity, but could cause confusion if users expect their liquidity to be locked or burned.

Recommendation

If allowing a "no-action" option is intentional, consider adding it as an explicit constant (e.g., NO_LOCK) for clarity. Alternatively, add validation to ensure only valid action types are accepted. Documentation should clearly explain all possible options and their effects.

Status

Resolved. A NONE constant was added, and an assertion was implemented to ensure that only valid actions are used.

LO-02 Empty Initial Token Template

Location

- `./extensions/self-listing-helper-v3.clar: 59`

Description

The wrapped token template data variable is initialized as an empty list. If the `set-wrapped-token-template` function isn't called before `create2` is used, the verification process will fail.

Recommendation

Consider implementing a check in the `create2` function to verify that the template has been set before attempting to use it.

Status

Acknowledged.

LO-03 Limited Parameter Validation

Location

- `./extensions/self-listing-helper-v3.clar: 245-246`

Description

The contract does not perform explicit validation on pool parameters like fee rates, ratios, and thresholds before passing them to external contracts.

While users can inspect these parameters before interacting with pools, explicit validation would add an additional layer of protection.

Recommendation

Consider adding reasonable boundary checks for critical parameters to prevent extreme values.

Status

Mitigated. Within the call to the AMM Pool contract, some of these values are validated.

LO-04 Duplicated Error Codes

Location

- `./aux/clarity-stacks/clarity-stacks.clar: 6-7`

Description

The contract defines two different error conditions using the same error code: `err-proof-too-short` and `err-block-header-too-short` use `u2001`.

This duplication makes it impossible to distinguish between a failure due to an insufficient proof length versus a block header that is too short, as both scenarios would return the same error code.

Recommendation

Assign unique error codes to each error condition.

Status

Resolved. `err-block-header-too-short` was assigned code `u2005`.

Enhancements

These items do not represent a security risk. They are best practices that we suggest implementing.

Id	Title	Status
EN-01	Limited Event Logging	**Implemented**
EN-02	Remove Debug Mode Functionality	**Implemented**

EN-01 Limited Event Logging

Location

- `./extensions/self-listing-helper-v3.clar`

Description

The contract only logs events for the create2 function. Adding similar logs for other significant operations would improve transparency and monitoring capabilities.

Status

Implemented.

EN-02 Remove Debug Mode Functionality

Location

- `./aux/clarity-stacks/clarity-stacks.clar`

Description

The contract contains commented out code intended to prevent debug mode from being used in the mainnet environment. While debug mode is currently set to `false`, debugging functionality is unnecessary.

Remove the debug functionality from production code.

Status

Implemented.

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

A privileged role defines parameters like the contract template, the approved tokens and fee rebates.

Upgrades

The contract does not implement upgradeability mechanisms.

About CoinFabrik

CoinFabrik is a research and development company specialized in Web3, with a strong background in cybersecurity. Founded in 2014, we have worked on over 500 decentralization projects, including EVM-based and other platforms like Solana, Algorand, and Polkadot. Beyond development, we offer security audits through a dedicated in-house team of senior cybersecurity professionals, working on code in languages such as Substrate, Solidity, Clarity, Rust, TEAL, and Stellar Soroban.

Our team has an academic background in computer science, software engineering, and mathematics, with accomplishments including academic publications, patents turned into products, and conference presentations. We actively research in collaboration with universities worldwide, such as Cornell, UCLA, and École Polytechnique in Paris, and maintain an ongoing collaboration on knowledge transfer and open-source projects with the University of Buenos Aires, Argentina. Our management and people experience team has extensive expertise in the field.

Methodology

CoinFabrik was provided with the source code. Our auditors spent one week 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 runtime 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.

Severity Classification

Security risks are classified as follows[2]:

Critical	• Manipulation of governance voting result deviating from voted outcome and resulting in a direct change from intended effect of original results • Direct theft of any user funds, whether at-rest or in-motion, other than unclaimed yield • Direct theft of any user NFTs, whether at-rest or in-motion, other than unclaimed royalties • Permanent freezing of funds • Permanent freezing of NFTs • Unauthorized minting of NFTs • Predictable or manipulable RNG that results in abuse of the principal or NFT • Unintended alteration of what the NFT represents (e.g. token URI, payload, artistic content) • Protocol insolvency
High	• Theft of unclaimed yield • Theft of unclaimed royalties • Permanent freezing of unclaimed yield • Permanent freezing of unclaimed royalties • Temporary freezing of funds • Temporary freezing NFTs

[2] This classification is based on the smart contract Immunefi severity classification system version 2.3. https://immunefi.com/immunefi-vulnerability-severity-classification-system-v2-3/

Medium	• Smart contract unable to operate due to lack of token funds • Block stuffing • Griefing (e.g. no profit motive for an attacker, but damage to the users or the protocol) • Theft of gas • Unbounded gas consumption • Security best practices not followed
Low	• Contract fails to deliver promised returns, but doesn't lose value • Other security issues with minor impact

Issue Status

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

- **Unresolved:** The issue has not been resolved.
- **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.
- **Acknowledged:** The issue remains in the code, but is a result of an intentional decision. The reported risk is accepted by the development team.
- **Mitigated:** Implemented actions to minimize the impact or likelihood of the risk.

Disclaimer

This audit report has been conducted on a **best-effort basis within a tight deadline defined by time and budget constraints**. We reviewed only the specific smart contract code provided by the client at the time of the audit, detailed in the Scope section. We do not review other components that are part of the solution: neither implementation, nor general design, nor business ideas that motivate them.

While we have employed the latest tools, techniques, and methodologies to identify potential vulnerabilities, **this report does not guarantee the absolute security of the contracts, as undiscovered vulnerabilities may still exist.** Our findings and recommendations are

suggestions to enhance security and functionality and are not obligations for the client to implement.

The results of this audit are valid solely for the code and configurations reviewed, and any modifications made after the audit are outside the scope of our responsibility. CoinFabrik disclaims all liability for any damages, losses, or legal consequences resulting from the use or misuse of the smart contracts, including those arising from undiscovered vulnerabilities or changes made to the codebase after the audit.

This report is intended exclusively for the **ALEX** team and should not be relied upon by any third party without the explicit consent of CoinFabrik. Blockchain technology and smart contracts are inherently experimental and involve significant risk; users and investors should fully understand these risks before deploying or interacting with the audited contracts.

Changelog

Date	Description
2025-02-28	Initial report based on commit `206981bd24497e54730870c1afbf7fb4e5227c59`.
2025-03-12	Final report based on commit 7d8ebf4291ef49a2889b9eb66904f37867a8a17e.