CoinFabrik

Security Audit Report

ALEX-Discrete Automated Market Maker

May 2025

V202411

Executive Summary

CoinFabrik was asked to audit the contracts for the **Discrete Automated Market Maker** project.

During this audit we found one high issue, and two medium issues. Also, an enhancement was proposed.

The high issue was resolved. One medium issue was resolved and the other was mitigated.

Scope

The audited files are from the git repository located at https://github.com/alexgo-io/alex-v3, in the `./contracts/` directory. The audit is based on the commit `180b7bdd1d6608928ec9590155add68f5dc68284`. Fixes were checked on commit `0af5cc609fc9d17c83b2546f666c798181877322`.

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

- `./amm-liquidity-token-v3.clar`: Manages the SIP-013 semi-fungible LP tokens that represent ownership of concentrated liquidity positions in the AMM pools.
- `./amm-pool-v3.clar`: Contains the core logic for the concentrated liquidity pools, managing token reserves, executing swaps within price ticks, calculating virtual balances, and handling fees.
- `./amm-pool-v3-helper.clar`: Provides user-friendly helper functions for common actions like multi-tick swaps and batch position management, interacting with the main pool contract.

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
HI-01	Tick Price Calculation Failures within Allowed Range	High	**Resolved**
ME-01	Discrepancies Between Code Logic and Whitepaper Formulas	Medium	**Mitigated**
ME-02	Incomplete Post-condition Verifiability for Specific Token IDs	Medium	**Resolved**

Critical Severity Issues

No issues found.

High Severity Issues

HI-01 Tick Price Calculation Failures within Allowed Range

Location

- `./amm-pool-v3.clar: 295-311`

Classification

- CWE-682: Incorrect Calculation[1]

Description

The contract permits ticks in the range `[-10000, +10000]`. However, the `tick-to-price` function, implemented using fixed-point math and pre-calculated constants (`PRICES_*`), fails to return a valid, non-zero price for a significant portion of this range. This occurs for two main reasons:

1. The `PRICES_*` constants only cover a limited tick magnitude, which shrinks as `bin-size` increases (e.g., `|t| <= 255` for `bin-size=10`, `|t| <= 127` for `bin-size=20`). Ticks exceeding these magnitudes result in `price=0`.

[1] https://cwe.mitre.org/data/definitions/682.html

2. For negative ticks within the magnitude limits, the inverse calculation `div-down(ONE_8, price)` truncates to `0` if the price for `abs(t)` is sufficiently large (e.g., for `bin-size=10, t <= -194`; `bin-size=20, t <= -102`).

When `tick-to-price` returns `0`, `get-virtual-balances` fails, causing transaction reverts. Additionally, fuzzing found 3185 inputs where `get-virtual-balances` failed due to a `ZeroDivisionError` because of `dd=0`, often occurring for ticks immediately adjacent to those causing the `price=0` failure (e.g., `bin-size=10, t=-193`), because `tick-to-price` returns a near-zero value insufficient for subsequent calculations.

Swaps attempting to cross into these non-calculable tick ranges will fail. Liquidity provision and removal within these ranges are impossible.

Recommendation

Modify `ensure-tick-in-range` and add checks within functions like `add-to-position` and the swap functions (`swap-*-ioc`) to strictly enforce the actual calculable tick limits based on `bin-size`.

Status

Resolved. The tick range validation was replaced with a direct assertion on the calculated price, ensuring it falls within a defined range – [`u10000`, `u1000000000000000`] – and that the constants fully covered the tick's magnitude, otherwise the transaction reverts.

Medium Severity Issues

ME-01 Discrepancies Between Code Logic and Whitepaper Formulas

Location

- `./amm-pool-v3.clar: 295-332`

Classification

- CWE-682: Incorrect Calculation[2]

Description

Comparison between the Clarity fixed-point implementation and a high-precision decimal implementation based on the whitepaper formulas revealed significant differences:

[2] https://cwe.mitre.org/data/definitions/682.html

- Even with relaxed tolerances (10% relative / 100 absolute), fuzzing found 33 inputs where the `vx` calculated by the fixed-point code differed significantly from the whitepaper's calculation. This indicates that the accumulation of precision losses through multiple fixed-point steps (`mul-down`, `div-down`, `sqrti`) in the intermediate calculations (`t`, `pty`, `dd`, `sqrt_term`, `de`) causes noticeable divergence from the theoretical model under certain conditions. Some of the inputs for `bin-size`, `tick`, `balance-x` and `balance-y` were:
  - `(1, -3, 1, 1)`,
  - `(1, -409, 2, 230)`,
  - `(1, -11, 1, 1)`,
  - `(1, -380, 0, 212)`,
  - `(1, -119, 6, 6)`,
  - `(1, -158, 4, 4)`,
  - `(1, -1052, 0, 29609)`,
  - `(1, -79, 4, 4)`.
- Fuzzing found 150 inputs where the ratio Clarity's VY / Whitepaper's VY often deviated significantly from `1.0` (e.g., ratios of `1.88`, `1.71`, `1.59` observed). For instance, the following values for `bin-size`, `tick`, `balance-x` and `balance-y`:
  - `(1, -1269, 14458, 171)`
  - `(20, -86, 100000000000000000, 830351740316485680)`,
  - `(20, -85, 879016566995484806, 879016566995484806)`,
  - `(1, -1265, 886157511886338114, 886157511886338114)`,
  - `(10, -157, 968358720193511977, 968358720193511977)`,
  - `(5, -290, 35969, 35969)`.

Recommendation

If the final user outcomes exhibit acceptable differences, document these discrepancies. Include the specific input factors responsible, and the justification for accepting the associated risk. Otherwise, if the final differences are unacceptable, consider revising the implementation to enhance precision.

Status

Mitigated. The new code reduces the precision loss by pre-scaling the input balances and an intermediate term (`x-pty`) before core mathematical operations, then unscaling the final `vx` and `vy` results.

ME-02 Incomplete Post-condition Verifiability for Specific Token IDs

Location

- `./amm-liquidity-token-v3.clar`

Classification

- CWE-358: Improperly Implemented Security Check for Standard[3]

Description

While tracking per-owner balances of SIP-013 Semi-Fungible Tokens[4], it doesn't follow the recommendation of defining NFTs for granular post-condition checks on individual `token-id` balance changes. This limits the ability of users to reliably verify changes to specific `token-id` balances using standard post-condition checks.

Recommendation

Follow the SIP-013 recommendation.

Status

Resolved. Fixed according to the recommendation.

Low Severity Issues

No issues found.

Enhancements

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

Id	Title	Status
EN-01	Implement Property-Based Testing	Not implemented

[3] https://cwe.mitre.org/data/definitions/358.html

[4] https://github.com/stacksgov/sips/blob/main/sips/sip-013/sip-013-semi-fungible-token-standard.md#transfer

EN-01 Implement Property-Based Testing

Location

- `./amm-pool-v3.clar`

Description

To proactively detect the identified mathematical issues (range failures, discrepancies, precision limits) and improve overall robustness, implement property-based fuzz testing

Status

Not 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

The system is centralized in the DAO/extension role (`SP102V8P0F7JX67ARQ77WEA3D3CFB5XW39REDT0AM.executor-dao`), which is authorized via `is-dao-or-extension` checks. This role holds exclusive privileges to create new pools, update pool fees, pause or sunset pools, claim accumulated fees, mint new LP tokens, and modify LP token metadata.

Upgrades

The contracts do not contain explicit mechanisms for code upgrades.

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, including automated tests that define the expected behavior. 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[5]:

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

[5] 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
2025-05-05	Initial report based on commit `180b7bdd1d6608928ec9590155add68f5dc68284`.
2025-05-19	Final report based on commit `0af5cc609fc9d17c83b2546f666c798181877322`.