# Bitflow Audit

December 2023

By CoinFabrik

v202307

# Executive Summary

CoinFabrik was asked to audit the contracts for the Bitflow project.

During this audit we found one critical issue, one high-severity issue and two minor-severity issues. Also, several enhancements were proposed.

All the issues were resolved, except for one of the minor issues that was mitigated. Some of the enhancement proposals were implemented.

# Scope

The audited files are from the git repository located at https://github.com/BitflowFinance/bitflow.git. The audit is based on the commit `f211029a06c1a3ee9cf72f5b5d0be08cb8a20ecc`.

Fixes for EN-01 Make Tests Work were checked on commit `b1e76a72989d2eb3ed23c69f80b05726d6c01b09.`

The rest of the fixes were checked on commit `a95b033ef93803979885b8d95d721b15375ff9e1`.

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

- `contracts/lp-token.clar`: liquidity-pool token contract
- `contracts/stableswap-stackingDAO.clar`: DEX to exchange between STX and a SIP10 token.
- `contracts/stableswap.clar`: DEX to exchange between two SIP10 tokens.
- `contracts/staking-and-rewards-stackingDAO.clar`: Rewards contract for `contracts/stableswap-stackingDAO.clar`.
- `contracts/staking-and-rewards.clar`: Rewards contract for `contracts/stableswap.clar`.

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 three 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
- Reentrancy attacks
- 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 |
|---|---|---|---|
| CR-01 | Orphan staking-and-reward | Critical | Resolved |
| HI-01 | Authentication via tx-sender | High | Resolved |
| MI-01 | Rogue Admin Can Take Over Stableswap | Minor | Resolved |
| MI-02 | Fee Avoidance Via Liquidity | Minor | Mitigated |

# 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

## CR-01 Orphan staking-and-reward

**Location**:

- `contracts/stableswap-stackingDAO.clar`
- `contracts/stableswap.clar`

- `contracts/staking-and-rewards-stackingDAO.clar`
- `contracts/staking-and-rewards.clar`

An admin may set a different contract to do the staking via the `set-staking-contract` function in both `contracts/stableswap-stackingDAO.clar` and `contracts/stableswap.clar`. This will starve the old staking contract of funds, but the old staking contract does not know that the stableswap has a new staking contract, as the stableswap contract referred by it is fixed[1]. This means that funds are awarded as if the fees collected to give rewards were transferred to the staking contract, even when they are not, eventually leading to failed transactions while collecting the rewards[2].

This is aggravated by the fact that the default staking contract for the `stableswap-stackingDAO` contract is wrong (see `contracts/staking-and-rewards-stackingDAO.clar:45`)

### Recommendation
Do not make it possible to change the staking contract in the stableswap contracts. It may be even better to just handle the staking and rewards functionality inside the stableswap contract itself. If the contracts are not merged, fix the setting in `contracts/staking-and-rewards-stackingDAO.clar:45`).

### Status
**Resolved**. The staking contracts can now be set only once via the set-staking-contract function in both `contracts/stableswap-stackingDAO.clar` and `contracts/stableswap.clar`.

# High Severity Issues

## HI-01 Authentication via tx-sender

**Location**:

- `contracts/stableswap-stackingDAO.clar`
- `contracts/stableswap.clar`
- `contracts/staking-and-rewards-stackingDAO.clar`
- `contracts/staking-and-rewards.clar`
- `contracts/lp-token.clar`

[1] See `contracts/staking-and-rewards.clar:92,132,140,291,296,349,442,475` and `contracts/staking-and-rewards-stackingDAO.clar:91,129,137,287,292,326,395,428`.
[2] See `contracts/staking-and-rewards.clar:318,320,326,329,364,366,372,375` and `contracts/staking-and-rewards-stackingDAO.clar:308,337`.

The system utilizes `tx-sender` for its authentication processes. This method, while functional, presents latent vulnerabilities, particularly exposing actors within the system to threats known as phishing[3].

Actors could inadvertently activate a malicious contract. Once activated, the deceptive contract can access and initiate certain functions, presenting actions as if they were done by the original actor. This impersonation potential poses risks, depending on the specific function being accessed.

### Recommendation

It is advisable to switch from using `tx-sender` to `contract-caller` for a more reliable and secure authentication method. It must be noted that when `tx-sender` is used as part of an `as-contract` invocation it does not lead to this issue, as it evaluates to the contract's principal.

### Status

**Resolved**. All authentication is made via the `contract-caller` now except for the transfer function in the `contracts/lp-token.clar` file. This use can be mitigated by postconditions [4].

## Medium Severity Issues

No issues found.

## Minor Severity Issues

### MI-01 Rogue Admin Can Take Over Stableswap

**Location**:

- `contracts/stableswap-stackingDAO.clar: 945`
- `contracts/stableswap.clar: 910`

A single rogue admin can kick out the rest of the admins for either of the stableswap contracts by calling the `remove-admin` function for the rest of the admins.

### Recommendation

Either do not allow a standard admin to add and remove administrators or require more than one admin to kick out another admin. If the second option is taken, adding an admin should have the same requirements.

[3] https://www.coinfabrik.com/blog/tx-sender-in-clarity-smart-contracts/
[4] See https://docs.stacks.co/docs/stacks-academy/post-conditions.

### Status
**Resolved**. The deployer of the contract cannot be removed from the admins.

## MI-02 Fee Avoidance Via Liquidity

**Location**:
- `contracts/stableswap-stackingDAO.clar`
- `contracts/stableswap.clar`

A user may avoid some fees (swap fees in `contracts/stableswap.clar`, buy and sell fees in `contracts/stableswap-stackingDAO.clar`) by exchanging tokens using the `add-liquidity` and `remove-liquidity` functions instead of the `swap-x-for-y` or `swap-y-for-x` functions.

### Recommendation
Do not have separate liquidity fees. Instead apply the same fees when swapping and adding liquidity.

### Status
**Mitigated**. While the issue is there the admins can mitigate it by having a close `liquidity-fee` setting.

# Enhancements

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

| ID | Title | Status |
|---|---|---|
| EN-01 | Make Tests Work | Implemented |
| EN-02 | Tests Should Check Values | Partially implemented |
| EN-03 | Lack of Tests | Implemented |
| EN-04 | Prevent Arbitration | Not implemented |
| EN-05 | Wrong Documentation | Implemented |

## EN-01 Make Tests Work

Calling `clarinet test` in the console fails.

### Recommendation

Make the tests run and pass when the `clarinet test` command is run.

### Status

**Implemented**. While making the tests work the development team discovered and fixed a bug in the `contracts/stableswap.clar` file that triggered 7 of the tests to fail. On commit `b1e76a72989d2eb3ed23c69f80b05726d6c01b09`.

## EN-02 Tests Should Check Values

The `stableswap` and `staking-and-rewards` contracts are the only ones that have test coverage. But those tests are faulty. They do not check that any value returned by any contract is correct nor that any token is transferred between principals.

Please note that the logic of a distributed exchange that distributes rewards based upon staked tokens is not simple. While we did our best to audit this source code in order to have a good chance of not having catastrophic issues the code needs proper testing.

### Recommendation

When testing contracts, the expected outcome of the operations should be checked, not just if the transaction failed or not. Make sure that all the different scenarios are tested.

### Status

**Partially implemented**.

## EN-03 Lack of Tests

The `lp-token`, `stableswap-stackingDAO` and `staking-and-rewards-stackingDAO` contracts do not have any automated tests.

### Recommendation

Make tests for these contracts as well, following the recommendations stated in EN-02 Tests Should Check Values.

### Status

**Implemented**.

## EN-04 Prevent Arbitration

**Location**:

- `contracts/stableswap-stackingDAO.clar`
- `contracts/stableswap.clar`

As mentioned in Other Considerations, stableswap Contracts Upstream subsection, each pair in both the `stableswap` and the `stableswap-stackingDAO` contracts needs to be approved individually and has separated liquidity. This may lead to situations where it is better to trade indirectly to exchange tokens, and may even lead to arbitration loops where tokens are extracted from the system for free.

### Recommendation

Follow the original design in the curve.fi contracts and allow all the possible trade pairs between the managed tokens.

### Status

**Not implemented**.

## EN-05 Wrong Documentation

**Location**:

- `contracts/stableswap-stackingDAO.clar`
- `contracts/staking-and-rewards-stackingDAO.clar`

A lot of the comments in the `*-stackingDAO.clar` files refer to the behavior of the non-stackingDAO contracts. For example, a lot of the functions don't have an `x-token` parameter but they are in documentation.

### Recommendation

Review all the documentation and properly document the contracts. If EN-04 Prevent Arbitration is fully implemented this item may become obsolete, as the stackingDAO contracts separation would not be possible.

### Status

**Implemented**. Comments referring to the `x-token` parameter were removed where it is appropriate.

# 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.

# stableswap Contracts Upstream

The development team informed us that the arithmetic calculations made in the stableswap contracts are based upon the curve.fi StableSwap3Pool contract[5]. But there are significant changes in the functionality provided. This is a non-exhaustive list of the differences:

- The upstream contract does not have the staking logic in the analyzed stableswap contracts.
- The admin actions in the upstream contract have a delayed time, allowing the users to withdraw funds before these settings take effect. This functionality is not present in the analyzed contracts.
- The upstream contract uses a single pool for all the tokens to be exchanged, and all the possible exchanges between tokens are supported. This is not true in the analyzed contracts. See EN-04 Prevent Arbitration.
- The way that convergence of the algorithm is detected was changed. In upstream there is no equivalent to the `convergence-threshold` setting.

# Centralization

The `stableswap-stakingDAO.clar` and `stableswap.clar` contracts have an admin role that can change a lot of configurations, including enabling and disabling each individual exchange pair and reducing to zero the funds for the rewards awarded in the `staking-and-rewards*` contracts. See the Priviledged Roles section for more information.

In the final version, the deployer of the contract cannot be removed from the admins in both `stableswap-stakingDAO.clar` and `stableswap.clar`. This was made to resolve MI-01 Rogue Admin Can Take Over Stableswap.

# Privileged Roles

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

## stableswap-stakingDAO.clar

### Admin

An account with the admin role can:

- pay different fees when buying and selling tokens via the `swap-x-for-y` and `swap-y-for-x` functions.
- create new pairs via the `create-pair` function.

[5] The code is based on https://github.com/curvefi/curve-contract/blob/master/contracts/pools/3pool/StableSwap3Pool.vy.

- approve or disapprove a pair via the `set-pair-approval` function. If a pair is disapproved, no exchange or liquidity operations can be made on it.
- add a new admin to the admins set via the `add-admin` function.
- remove an admin of the admin set via the `remove-admin` function.
- change the buy fee via the `change-buy-fee` function.
- change the sell fee via the `change-sell-fee` function.
- change the admin swap fee via the `change-admin-swap-fee` function.
- change the liquidity fee via the `change-liquidity-fee` function.
- change the amplification coefficient of a pair via the `change-amplification-coefficient` function.
- change the convergence threshold via the `change-convergence-threshold` function.
- change the staking contract via the `set-staking-contract` function.
- change the staking dao contract via the `set-staking-dao-contract` function.
- change the bitflow contract via the `set-bitflow-contract` function.

On deployment there is a single admin, which is the contract's deployer.

### stableswap.clar

#### Admin

An account with the admin role can:

- create new pairs via the `create-pair` function.
- approve or disapprove a pair via the `set-pair-approval` function.
- add a new admin to the admins set via the `add-admin` function.
- remove an admin of the admin set via the `remove-admin` function.
- change the lps fee and the protocol fee via the `change-swap-fee` function.
- change the liquidity fee via the `change-liquidity-fee` function.
- change the amplification coefficient of a pair via the `change-amplification-coefficient` function.
- change the convergence threshold via the `change-convergence-threshold` function.
- change the staking contract via the `set-staking-contract` function.

On deployment there is a single admin, which is the contract's deployer.

## Changelog

- 2023-12-13 – Initial report based on commits `f211029a06c1a3ee9cf72f5b5d0be08cb8a20ecc` and `b1e76a72989d2eb3ed23c69f80b05726d6c01b09`.

- 2023-12-22 – Fixes checked on commit `a95b033ef93803979885b8d95d721b15375ff9e1`.

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