1Inch Farming Audit

December 2022

By CoinFabrik

v202209

Executive Summary

CoinFabrik was asked to audit the contracts for the 1inch's Farming project. The audited files are from the git repository located at https://github.com/1inch/farming. The audit is based on the commit `8403ce63df903db66c5c8fd7958470c6827df7e6`. The fixes were added to the commit `bbb8b0db45dfb4ae0be860fef3ca4df9a761845f`.

During this audit we found no critical issues, no medium issues and two minor issues.

Scope

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

- `contracts/FarmingLib.sol`: Farming logic.
- `contracts/FarmingPod.sol: Pod allowing single reward token.`
- `contracts/FarmingPool.sol`: Farming for backward compatibility.
- `contracts/MultiFarmingPool.sol`: Pod allowing multiple reward tokens.
- `accounting/FarmAccounting.sol`: Reward storage and calculation.
- `accounting/UserAccounting.sol`: User Balance storage and calculation.
- `interfaces/IFarmingPod.sol`: Farming Pod interface.
- `interfaces/IFarmingPool.sol`: Farming Pool interface.
- `interfaces/IMultiFarmingPod.sol`: MultiFarming interface.

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, and general documentation about the project. 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
- Outdated version of Solidity compiler
- 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

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	Multiple Possible Reentrancy points	Minor	Resolved
MI-02	Can't Remove Reward Tokens From MultiFarmingPod	Minor	Acknowledged

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 Multiple Possible Reentrancy Points

Location:

- `contracts/FarmingPool.sol:110`,
- `contracts/MultiFarmingPod.sol:73`

There is a reentrancy point at `FarmingPool.rescueFunds()`:

```
            token.safeTransfer(distributor, amount);
```

As the `token` variable can be any arbitrary token including a ERC-777 derived one, transfer hooks can be used to do reentrancy attacks.

The same situation applies to `MultiFarmingPod.sol.startFarming()`:

```
        rewardsToken.safeTransferFrom(msg.sender, address(this), amount);
```

The impact of both reentrancy points is low because in the FarmingPool.sol case, it can only be triggered by a distributor role.

Recommendation
Refactor the function so it follows the check-effects-interaction pattern, or implement reentrancy guards.

Status
All reentrancies were fixed by reordering calls to follow the check-effects-interaction pattern at commit `bbb8b0db45dfb4ae0be860fef3ca4df9a761845f`.

MI-02 Can't Remove Reward Tokens From MultiFarmingPod

Location:

- `contracts/MultiFarmingPod.sol:65`

The `MultiFarmingPod` contract allows adding up to 5 reward tokens, using the `addRewardsToken()` function. However, there is no function to remove a reward token, so if you add the maximum amount (5 tokens) you can no longer add any other to the `MultiFarmingPool`.

Recommendation
Add the possibility to remove a token, taking in account that it will not be a trivial operation as all already staked tokens need to be correctly handled.

Status
The developer stated that in the case of exceeding the amount of possible rewards tokens, the user should create a new multifarm pod.

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 role of distributor can extract all funds from the `FarmingPool`, `FarmingPod` and `MultiFarmingPod` contracts, using the `rescueFunds()` functions present in all those contracts.

Upgrades

No contract audited has the capability to be upgraded.

Privileged Roles

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

FarmingPod

Owner

The owner of the contract can set the distributor address, and also can renounce ownership.

Distributor

The Distributor role can start Farming and rescue Funds. This role cannot be renounced.

Token

The Token role can update the balances. This role cannot be renounced

FarmingPool

Owner

The owner of the contract can set the distributor address, and also can renounce ownership.

Distributor

The Distributor role can start Farming and rescue Funds. This role cannot be renounced.

MultiFarmingPod

Owner

The owner of the contract can set the distributor address and add reward tokens. Also it can renounce ownership.

Distributor

The Distributor role can start farming and rescue funds. This role cannot be renounced.

Token

The Token role can update the balances. This role cannot be renounced.

Changelog

- 2022-12-2 – Initial report based on commit `8403ce63df903db66c5c8fd7958470c6827df7e6`.
- 2022-12-14 – Fixes checked on commit `bbb8b0db45dfb4ae0be860fef3ca4df9a761845f`.

Disclaimer: This audit report is not a security warranty, investment advice, or an approval of the 1Inch project since CoinFabrik has not reviewed its platform. Moreover, it does not provide a smart contract code faultlessness guarantee.