1inch Farming Security Analysis

by Pessimistic

This report is public

January 16, 2023

Abstract

In this report, we consider the security of smart contracts of 1inch Farming project. Our task is to find and describe security issues in the smart contracts of the platform.

Disclaimer

The audit does not give any warranties on the security of the code. A single audit cannot be considered enough. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. Besides, a security audit is not investment advice.

Summary

In this report, we considered the security of 1inch Farming smart contracts. We described the audit process in the section below.

The initial audit showed several low-severity issues.

The codebase relies on the `library` feature of Solidity, which complicates code structure and negatively affects the readability and maintainability of the project. Besides, the **MultiFarmingPod** contract lacks documentation. However, the overall code quality is high.

After the initial audit, the developers updated the codebase. In this update, they fixed two issues.

General recommendations

We recommend documenting the **MultiFarmingPod** contract, covering the codebase with NatSpec comments, and fixing the discovered issues.

Project overview

Project description

For the audit, we were provided with 1inch Farming project on a public GitHub repository, commit 8403ce63df903db66c5c8fd7958470c6827df7e6.

The scope of the audit included the whole repository.

The documentation for the project included only the **README.md** file.

All 64 tests pass successfully. The code coverage is 92%.

The total LOC of audited sources is 485.

Codebase update

After the initial audit, the developers updated the codebase. For the recheck, we were provided with commit ee7a4bd7d2655800a22eb450fd12125bd47b0198. In this update, the developers refactored existing functionality slightly and fixed a few issues mentioned in the initial report. We did not find new issues in this update.

In the updated codebase, all 66 tests pass. The overall code coverage is 88.82%.

Audit process

We started the audit on November 22, 2022, and finished on December 1, 2022.

We inspected the provided materials and documents left from the previous audit. Since the code had evolved significantly, we checked the whole codebase rather than the diff. Besides, the **MultiFarmingPod** contract was the latest addition to the repo, and thus we paid extra attention to its code.

We manually analyzed all the contracts within the scope of the audit and checked their logic. Among other, we verified the following properties of the contracts:

- Users' stake tokens are safe within **FarmingPool** contract.
- Staking logic correctly assigns rewards to the users.
- **FarmingPod** and **MultiFarmingPod** contracts are compatible with ERC20Pods contract.

We scanned the project with the static analyzer Slither extended with our own set of rules and then manually verified all the occurrences found by the tool.

We ran tests and calculated the code coverage.

We combined in a private report all the verified issues we found during the manual audit or discovered by automated tool.

After the initial audit, the developers improved the codebase and provided us with a new commit. They fixed some of the issues in this update and slightly refactored the existing code. We reviewed the updated codebase and updated the report.

Manual analysis

The contracts were completely manually analyzed, their logic was checked. Besides, the results of the automated analysis were manually verified. All the confirmed issues are described below.

Critical issues

Critical issues seriously endanger project security. They can lead to loss of funds or other catastrophic consequences. The contracts should not be deployed before these issues are fixed.

The audit showed no critical issues.

Medium severity issues

Medium issues can influence project operation in the current implementation. Bugs, loss of potential income, and other non-critical failures fall into this category, as well as potential problems related to incorrect system management. We highly recommend addressing them.

M01. Insufficient documentation

The documentation for the project includes only the **README.md** file, which does not describe the **MultiFarmingPod** contract. Besides, the codebase lacks NatSpec comments.

Proper documentation should explicitly describe the purpose and behavior of the contracts, their interactions, and main design choices. It is also essential for any further integrations.

Low severity issues

Low severity issues do not directly affect project operation. However, they might lead to various problems in future versions of the code. We recommend fixing them or explaining why the team has chosen a particular option.

L01. Inheritance order

Both **FarmingPod** and **MultiFarmingPod** contracts inherit from **Pod** and **Ownable** contracts (in this order). We recommend starting the inheritance list with standard contracts since their storage layout is less likely to change in the future.

> *Comment from the developers:* *Will not fix.*

L02. Code quality (fixed)

According to [OpenZeppelin documentation](#) `_beforeTokenTransfer` function should be declared as `virtual`. The **FarmingPool** contract does not follow this recommendation.

The issue has been fixed and is not present in the latest version of the code.

L03. Misleading comment (fixed)

Misleading comment on line 22 in **FarmAccounting** contract:

> */// @dev Requires extra 18 decimals for precision, result should not exceed 10**54*

In fact, the return value can be up to `1e60`. However, it does not affect the contract.

The issue has been fixed and is not present in the latest version of the code.

Notes

N01. Confusing events

`startFarming` function of **FarmingPool**, **FarmingPod**, and **MultiFarmingPod** contracts emit `RewardAdded` event. Its `reward` argument value is the sum of freshly added reward and leftovers from an unfinished farming campaign. Therefore, one cannot easily get the new reward size when observing emitted events. We recommend adding a new reward amount as an extra parameter of the `RewardAdded` event.

> *Comment from the developers:* *Will not fix.*

N02. Overpowered role

The distributor can use `startFarming` function to slow down existing farming. Alternatively, they can withdraw any or all reward tokens at any moment, which will effectively disable it. This applies to **FarmingPool**, **FarmingPod**, and **MultiFarmingPod** contracts.

> *Comment from the developers:* *Will not fix.*

This analysis was performed by Pessimistic:

Evgeny Marchenko, Senior Security Engineer
Yhtyyar Sahatov, Junior Security Engineer
Irina Vikhareva, Project Manager

January 16, 2023