1inch NETWORK

1inch FixedRateSwap Security Analysis

by Pessimistic

This report is public

December 1, 2021

Abstract

In this report, we consider the security of FixedRateSwap smart contracts of 1inch 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. One 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, security audit is not an investment advice.

Summary

In this report, we considered the security of 1inch FixedRateSwap smart contracts. We performed our audit according to the procedure described below.

The audit showed only a few issues of low severity. They do not endanger project security. Overall, the source code of the contracts is of very high quality.

After the initial audit, the code base was updated. Min return feature was introduced and all issues from initial audit were fixed.

General recommendations

We have no further recommendations for the code base improvement.

Project overview

Project description

For the audit, we were provided with 1inch FixedRateSwap project on a public GitHub repository, commit 0b5a75e9f56e7d21c290dd28c59dc140dcbcc1d5.

The project has minimal README.md file, the code has detailed NatSpec comments. We consider this documentation adequate to the project.

All 34 tests pass, the code coverage is 91.06%. CI runs tests and measures coverage regularly.

The total LOC of audited sources is 268.

Code base update

After the initial audit, the code base was updated. For the recheck, we were provided with commit 712fced5e8e6bb1606e060e8ce0c43fadd578890. The updated has added min return checks to all the interaction functions. All previosly discovered issues have been fixed.

Procedure

In our audit, we consider the following crucial features of the code:

1. Whether the code is secure.
2. Whether the code corresponds to the documentation (including whitepaper).
3. Whether the code meets best practices.

We perform our audit according to the following procedure:

- Automated analysis
    - We scan project's code base with automated tool Slither.
    - We manually verify (reject or confirm) all the issues found by tools.
- Manual audit
    - We manually analyze code base for security vulnerabilities.
    - We assess overall project structure and quality.
- Report
    - We reflect all the gathered information in 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 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.

The audit showed no issues of medium severity.

Low severity issues

Low severity issues do not directly affect project operation. However, they might lead to various problems in the future versions of the code. We recommend taking them into account.

Code quality (fixed)

- One can slightly improve `_getVirtualAmountsForDepositImpl` and `_getRealAmountsForWithdrawImpl` functions.
  We recommend calculating `dy` and `shift` values as the first actions within the `for` loops at lines 343-358 and 384-400 of `FixedRateSwap` contract. It slightly simplifies code and reduces gas consumption.
- Consider declaring `withdrawWithRatio` function as `external` instead of `public` to improve code readability and optimize gas consumption.
- We recommend following CEI (checks-effects-interactions) pattern in `depositFor` function since it prevents some types of re-entrancy attacks.

These issues have been fixed and are not present in the latest version of the code.

This analysis was performed by Pessimistic:
Evgeny Marchenko, Senior Security Engineer
Vladimir Tarasov, Security Engineer
Irina Vikhareva, Project Manager

December 1, 2021