1inch Aggregation Router V4 Security Analysis

by Pessimistic

This report is public

16 October, 2021

Abstract

In this report, we consider the security of smarts contracts of 1inch Aggregation Router V4. 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 Aggregation Router V4 smart contracts. We performed our audit according to the procedure described below.

The audit discovered only a few issues in the code, all of low severity. They do not endanger project security. However, the code is very complex due to radical gas optimizations and lacks proper documentation and comments. This poses a risk for future development or any possible integrations.

After the initial audit, the code base was updated. The code was refactored and simplified, most issues were fixed. However, a logical issue of medium severity was introduced with this update.

General recommendations

We recommend providing additional documentation to the code and addressing the other mentioned issues.

Project overview

Project description

For the audit, we were provided with 1inch Aggregation Router V4 project on a public GitHub repository, commit 93868c483180cf74fc2551568f0396938b3eeaa8.

The scope of the audit included four files:

- AggregationRouterV4.sol
- UnoswapV3Router.sol
- ClipperRouter.sol
- LimitOrderProtocolRFQ.sol

and their dependencies.

The project has no description file, the code lacks detailed NatSpec comments, and there is no public documentation available.

All tests pass, the code coverage wasn't measured due to code complexity.

The total LOC of audited sources is 479.

Code base update

After the initial audit, the code base was updated. For the recheck, we were provided with commit af3e7e23b080d92655c1d0f0e9aa473c0da11cb4. The code was refactored and significantly simplifyed. Most 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 tools: Slither and SmartCheck.
  - 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 often lead to the loss of funds or other catastrophic failures. The contracts should not be deployed before these issues are fixed. We highly recommend fixing them.

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.

Lack of Documentation

The code base of the project is heavily optimized, which increases its complexity. It includes multiple assembly blocks and other low-level tricks, but very few comments in any form. This poses a risk for future development or any possible integrations.

Considering the importance of the contracts to the project, the documentation is required to streamline both development and audit processes. It should explicitly explain the purpose and behavior of the contracts, their interactions, and main design choices. The code should be extensively covered by NatSpec comments.

Order manipulation (new)

Partially filled orders are invalidated in the updated version of the **LimitOrderProtocolRFQ** contract. One can intentionally fill only a small fraction of the whole order to make it unusable. One can manipulate the market by mass-invalidating orders.

Low severity issues

Low severity issues don't directly affect project's operations. However, they might lead to various problems in the future versions of the code. We recommend taking them into account.

Possibility of locking money (fixed)

Payable functions `uniswapV3SwapTo` and `clipperSwapTo` transfer ether only in case of wrapping to WETH token, thus making any calls with value, but without wrapping to lock money on contract's balance. Note, that there's no function to rescue such funds.

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

Possible Bug (fixed)

Utilizing `msg.sender` as the parameter to `.transferFrom` (inside **LimitOrderProtocolRFQ** at line 186) is not recommended, since it's not verified with signature, and might not be the actual spender in the case of permitted transaction.

The code has been updated, this issue is not present in the latest version of the code.

Events logging

There are no `indexed` fields in the `Swapped` event of **AggregationRouterV4**, however it might be useful for off-chain integration.

> *We only use these events in Dune Analytics, and marking them as `indexed` unnecessarily increases gas consumption.*

Code quality (fixed)

- Redundant imports in **AggregationRouterV4** at line 4, and in **LimitOrderProtocolRFQ** at lines 13 and 14.

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

- Consider moving `OrderRFQ` manipulation logic from **LimitOrderProtocolRFQ** into a separate library to improve code modularity.

  The code has been updated, this issue is not present in the latest version of the code.

- Unnecessary ternary operator at line 117 of **AggregationRouterV4**, since the variable is only used once and under the same condition.

  The code has been updated, this issue is not present in the latest version of the code.

This analysis was performed by Pessimistic:

Vladimir Tarasov, Security Engineer
Evgeny Marchenko, Senior Security Engineer
Irina Vikhareva, Project Manager

16 October, 2021