1inch Fusion Settlement Extension Security Analysis

by Pessimistic

This report is public

December 13, 2023

Abstract

In this report, we consider the security of smart contracts of 1inch Fusion Settlement Extension 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 Fusion Settlement Extension smart contracts. We described the audit process in the section below.

The audit showed only one issue of low severity. It does not endanger project security.

The overall code quality is high.

General recommendations

We recommend fixing the mentioned issue.

Project overview

Project description

For the audit, we were provided with 1inch Fusion Settlement Extension project on a public GitHub repository, commit ff7909c4f32bee8879211607275dc30d788afee8.

The scope of the audit included:

- **SettlementExtension.sol**,
- **FeeBankCharger.sol**,
- **interfaces/IFeeBankCharger.sol**.

The documentation for the project included NatSpec comments.

All 81 tests pass successfully. The code coverage is 93.65%.

The total LOC of audited sources is 205.

Audit process

We started the audit on November 22, 2023, and finished on November 24, 2023.

We inspected the materials provided for the audit. Then, we contacted the developers for an introduction to the project.

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:

- Possible gas optimizations;
- Calculations do not overflow.

We scanned the project with the following tool:

- Semgrep rules for smart contracts.

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.

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 severity 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.

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 future versions of the code. We recommend fixing them or explaining why the team has chosen a particular option.

L01. Redundant code (fixed)

Manually applying mask `_RESOLVER_ADDRESS_MASK` in this expression `uint80(uint160(resolver) & _RESOLVER_ADDRESS_MASK)` is unnecessary, as the compiler itself will apply this mask when the `resolver` is cast to `uint80` at line 173 in the **SettlementExtension** contract.

The issue has been fixed at commit 3017c3b260f9193d68460a10219df19dca7d69de.

Notes

N01. Custom dispatcher (commented)

Solidity compiler adds checks on calldata size, and that non-32 byte type values do not exceed the type range in `public`/`external` functions. Since only the Limit order protocol will be calling the **SettlementExtension** contract and all size checks are conducted within the Limit order protocol contract itself, consider writing a custom dispatcher in the `fallback` function to eliminate these redundant checks and save gas in the **SettlementExtension** contract.

> *Comment from the developers:* *Won't fix. We will try this approach next time.*

N02. Validation of the order data (commented)

In the current system, it is the resolver's responsibility to validate user-signed order data. Otherwise, the user might grieve the resolvers by putting extremely high resolver/integration fees, which are charged from them.

> *Comment from the developers:* *Won't fix. As resolvers are professional traders they expect to properly validate orders including the fee part as well.*

This analysis was performed by Pessimistic:

Pavel Kondratenkov, Senior Security Engineer
Yhtyyar Sahatov, Security Engineer
Irina Vikhareva, Project Manager

December 13, 2023