Smart Contract Security Audit Report

1inch SettlementExtension

1. Contents

2. General Information

This report contains information about the results of the security audit of the 1inch (hereafter referred to as "Customer") SettlementExtension smart contract, conducted by Decurity in the period from 11/30/2023 to 11/27/2023.

2.1. Introduction

Tasks solved during the work are:

- Review the protocol design and the usage of 3rd party dependencies,
- Audit the contracts implementation,
- Develop the recommendations and suggestions to improve the security of the contracts.

2.2. Scope of Work

The audit scope included the contracts in the following repository: https://github.com/1inch/limit-order-settlement. Initial review was done for the commit ff7909c4f32bee8879211607275dc30d788afee8.

The following contracts have been tested:

- contracts/SettlementExtension.sol

2.3. Threat Model

The assessment presumes the actions of an intruder who might have the capabilities of any role (an external user, token owner, token service owner, or a contract).

The main possible threat actors are:

- User (Maker),
- Protocol owner,
- Resolver (Taker).

2.4. Weakness Scoring

An expert evaluation scores the findings in this report, and the impact of each vulnerability is calculated based on its ease of exploitation (based on the industry practice and our experience) and severity (for the considered threats).

2.5. Disclaimer

Due to the intrinsic nature of the software and vulnerabilities and the changing threat landscape, it cannot be generally guaranteed that a certain security property of a program holds.

Therefore, this report is provided “as is” and is not a guarantee that the analyzed system does not contain any other security weaknesses or vulnerabilities. Furthermore, this report is not an endorsement of the Customer’s project, nor is it an investment advice.

That being said, Decurity exercises the best effort to perform its contractual obligations and follow the industry methodologies to discover as many weaknesses as possible and maximize the audit coverage using limited resources.

3. Summary

As a result of this work, we haven't discovered any exploitable security issues. Apart from the manual review, we also performed differential fuzzing using the previous implementation of the same logic in Yul and Solidity.

The suggestions in the report include fixing the low-risk issues and some best practices (see Security Process Improvement).

3.1. Suggestions

The table below contains the discovered issues, their risk level, and their status as of May 3, 2023.

Table. Discovered weaknesses

Issue	Contract	Risk Level	Status
Redundant import	limit-order-settlement/contracts/SettlementExtension.sol	Info	Not fixed
Unnecessary masking	limit-order-settlement/contracts/SettlementExtension.sol	Info	Not fixed

4. General Recommendations

This section contains general recommendations on how to improve the overall security level.

The Findings section contains technical recommendations for each discovered issue.

4.1. Security Process Improvement

The following is a brief long-term action plan to mitigate further weaknesses and bring the product security to a higher level:

- Keep the whitepaper and documentation updated to make it consistent with the implementation and the intended use cases of the system,
- Perform regular audits for all the new contracts and updates,
- Ensure the secure off-chain storage and processing of the credentials (e.g. the privileged private keys),
- Launch a public bug bounty campaign for the contracts.

5. Findings

5.1. Redundant import

Risk Level: Info

Status: Not fixed

Contracts:

- limit-order-settlement/contracts/SettlementExtension.sol

Location: Lines: 5.

Description:

The IERC20 import is unnecessary because it's already imported in FeeBankCharger.sol.

```
limit-order-settlement/contracts/SettlementExtension.sol:
  5: import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
```

Remediation:

Consider removing an import.

5.2. Unnecessary masking

Risk Level: Info

Status: Not fixed

Contracts:

- limit-order-settlement/contracts/SettlementExtension.sol

Location: Lines: 173. Function: `_isWhitelisted`.

Description:

In the following code snippet casting can be done through `uint80()` without using of `_RESOLVER_ADDRESS_MASK`.

```
limit-order-settlement/contracts/SettlementExtension.sol:
  173:             uint80 maskedResolverAddress = uint80(uint160(resolver) &
_RESOLVER_ADDRESS_MASK);
```

Remediation:

Consider refactoring the code to use `uint80()` for casting directly, thereby eliminating the need for `_RESOLVER_ADDRESS_MASK`. This will simplify the code and maintain readability.

6. Appendix

6.1. About us

The Decurity team consists of experienced hackers who have been doing application security assessments and penetration testing for over a decade.

During the recent years, we've gained expertise in the blockchain field and have conducted numerous audits for both centralized and decentralized projects: exchanges, protocols, and blockchain nodes.

Our efforts have helped to protect hundreds of millions of dollars and make web3 a safer place.