1inch Fusion Atomic Swaps Security Analysis

by Pessimistic

March 25, 2024

Abstract

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

The initial audit showed one issue of medium severity: Maker address in case of cancellation. Also, one low-severity issue was found.

The overall code quality is good. The code is highly optimized, and covered with tests.

After the initial audit, the developers fixed the M01 issue and provided comments on the other issues. Further details are available in the respective issue sections.

General recommendations

We do not have any additional recommendations.

Project overview

Project description

For the audit, we were provided with 1inch Fusion Atomic Swaps project on a private GitHub repository, commit 2e0fafdddb59ba40d8d346b51765910abd3474ed.

The scope of the audit included the entire repository, excluding mock contracts.

The documentation for the project included a link to a Google document.

All 49 tests pass successfully. The code coverage is 90.12%.

The total LOC of audited sources is 387.

Audit process

We started the audit on March 5, 2024 and finished on March 11, 2024.

We inspected the materials provided for the audit. After that, we performed preliminary research and specified the parts of the code and logic that require additional attention during an audit.

During the work, we stayed in touch with the developers and discussed confusing or suspicious parts of the 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:

- Whether the code corresponds to the documentation;
- Potential issues with the integration of L2 networks;
- The possibility of attacks from both the resolver and the maker;
- Possible code optimizations.

We scanned the project with the following tools:

- Static analyzer Slither;
- Our plugin Slitherin with an extended set of rules.

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 tools.

After the initial audit, the developers provided us with the commit 106eeffca3ef5df7df8449bb466591cf0d1fb84a, which contains the fix for the M01 issue. We verified the fix on March 25, 2024, and updated the report accordingly.

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.

M01. Maker address in case of cancellation (fixed)

According to **EscrowSrc.sol**, in the event of cancellation, tokens are sent to `immutables.maker.get()`. During cross-chain initialization, `immutables.maker.get()` is set to the provider's `order.receiver` address. This `order.receiver` denotes the address on the destination chain where the maker intends to receive funds. However, ownership of an address on one chain does not guarantee access to it on another chain. The problems may arise, particularly when these addresses are the smart contracts (or smart contract-based wallets) deployed using the `create` opcode.

The issue has been fixed in commit 106eeffca3ef5df7df8449bb466591cf0d1fb84a.

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. Gas optimization (commented)

The OpenZeppelin repository contains a more optimized version of the **Clones** library. We recommend reviewing and considering its use. Note that this version has not yet been included in the production release of the library.

> *Comment from the developers: Will fix. We are going to use the `Clones.cloneDeterministic` function from the OZ repository as soon as it is released.*

Notes

N01. Privileged role (commented)

The swap process requires adherence to a specific timeline. For instance, there exists a period during which only the resolver can unlock tokens on the destination chain. However, there is no check on the minimum length of this period. The project relies on the relayer role for such timeline checks.

> *Comment from the developers: Not an issue. On the one hand, on a frontend that maintains 1inch, timelocks will be taken from a reliable service and given to the user to sign. On the other hand, Resolver is also motivated to validate timelocks, as the user can sign the intervals deliberately favourable to them.*

N02. The process of escrow creation (commented)

In the current implementation, it is essential for the resolver to deposit funds into the escrow and fill the order within the same transaction. Otherwise, funds become stuck on the address of an undeployed contract. For a similar reason, the resolver should first create the escrow on the source chain.

> *Comment from the developers: Not an issue. The logic described fits within the protocol design context. Draft for a possible implementation of **Resolver** contract with sending a deposit within a single transaction is implemented in https://github.com/1inch/cross-chain-swap/pull/52. The requirement to first deploy escrow on the source chain and then on the destination chain is reflected in the documentation.*

N03. Time constraints might not work properly in case of L2 networks (commented)

Several L2 networks have mechanisms to combat censorship and resolve disputes. Note that these processes may take longer than the time constraints imposed by the cross-chain swap flow. For example, a swap could be completed on the source chain while still pending on the L2 network.

> *Comment from the developers: Not an issue. We will adjust the timelocks for all supported L2 chains to reflect possible differences in their finalisation periods.*

This analysis was performed by Pessimistic:

Evgeny Marchenko, Senior Security Engineer
Pavel Kondratenkov, Senior Security Engineer
Yhtyyar Sahatov, Security Engineer
Konstantin Zherebtsov, Business Development Lead
Irina Vikhareva, Project Manager
Alexander Seleznev, Founder

March 25, 2024