Fixed Rate Swap Smart Contract Audit

1inch Network

15 November 2021

iosiro

1. Introduction

iosiro was commissioned by 1inch Network to conduct an audit of their updated Fixed Rate Swap smart contracts. The audit was performed by 2 auditors between 25 October and 02 November 2021, consuming a total of 10 person-days. An audit was conducted on a prior version of the contracts and is available here.

This report is organized into the following sections.

- **Section 2 - Executive summary:** A high-level description of the findings of the audit.
- **Section 3 - Audit details:** A description of the scope and methodology of the audit.
- **Section 4 - Design specification:** An outline of the intended functionality of the smart contracts.
- **Section 5 - Detailed findings:** Detailed descriptions of the findings of the audit.

The information in this report should be used to better understand the risk exposure of the smart contracts, and as a guide to improving the security posture of the smart contracts by remediating issues identified. The results of this audit are only a reflection of the source code reviewed at the time of the audit and of the source code that was determined to be in-scope.

The purpose of this audit was to achieve the following:

- Identify potential security flaws.
- Ensure that the smart contracts functioned according to the documentation provided.

Assessing the economics, game theory, or underlying business model of the platform was strictly beyond the scope of this audit.

Due to the unregulated nature and ease of transfer of cryptocurrencies, operations that store or interact with these assets are considered very high risk with regard to cyber-attacks. As such, the highest level of security should be observed when interacting with these assets. This requires a forward-thinking approach, which takes into account the new and experimental nature of blockchain technologies. Strategies that should be used to encourage secure code development include:

- Security should be integrated into the development lifecycle and the level of perceived security should not be limited to a single code audit.
- Defensive programming should be employed to account for unforeseen circumstances.
- Current best practices should be followed where possible.

2. Executive summary

This report presents the findings of an audit performed by iosiro of 1inch Network's updated Fixed Rate Swap smart contracts. The updated code implemented functionality to handle deposits and withdrawals of amounts in any ratio of the asset pair.

Several informational issues were identified during the audit. Negligible risk was posed by the issues. The code was found to be of a high standard and made use of best practices.

3. Audit details

3.1 Scope

The source code considered in-scope for the assessment is described below. Code from all other files is considered to be out-of-scope. Out-of-scope code that interacts with in-scope code is assumed to function as intended and introduce no functional or security vulnerabilities for the purposes of this audit.

3.1.1 1inch smart contracts

Commit: 0b5a75e

File: FixedRateSwap.sol

3.2 Methodology

A variety of techniques were used while conducting the audit. These techniques are briefly described below.

3.2.1 Code review

The source code was manually inspected to identify potential security flaws. Code review is a useful approach for detecting security flaws, discrepancies between the specification and implementation, design improvements, and high-risk areas of the system.

3.2.2 Dynamic analysis

The contracts were compiled, deployed, and tested in a test environment. Manual analysis was used to confirm that the code operated at a functional level, and to verify the exploitability of any potential security issues identified. This included fuzzing different transactions to assess the system's robustness to random combinations of transactions and values.

3.2.3 Automated analysis

Tools were used to automatically detect the presence of several types of security vulnerabilities, including reentrancy, timestamp dependency bugs, and transaction-ordering dependency bugs. The static analysis results were manually analyzed to remove false-positive results. True positive results would be indicated in this report. Static analysis tools commonly used include Slither, MythX, as well as Securify. Furthermore, the Remix IDE, compilation output, and linters are also used to identify potential areas of concern.

3.3 Risk ratings

Each issue identified during the audit has been assigned a risk rating. The rating is determined based on the criteria outlined below.

- **High risk** - The issue could result in a loss of funds for the contract owner or system users.
- **Medium risk** - The issue resulted in the code specification being implemented incorrectly.
- **Low risk** - A best practice or design issue that could affect the security of the contract.
- **Informational** - A lapse in best practice or a suboptimal design pattern that has a minimal risk of affecting the security of the contract.

- **Closed** - The issue was identified during the audit and has since been addressed to a satisfactory level to remove the risk that it posed.

4. Design specification

The following section outlines the intended functionality of the system at a high level.

4.1 Fixed rate swap contract

The fixed rate swap contract is an Automated Market Maker (AMM) that allows swapping tokens with a near 1:1 rate subject to a variable fee. The fee in this kind of AMM is lower than that of typical constant product AMMs. The system is specifically designed to allow for swaps between similar asset types, e.g. USDC and USDT.

Deposits

Any user can deposit any ratio of assets into the pool. If the ratio between the deposit amounts differs from that of the pool amounts, a fee is calculated by determining the virtual swap amount needed to balance the amounts. The LP tokens sent to the depositor equate to the sum of the deposit amounts minus the fee.

Withdrawal

LP tokens can be burned to retrieve a proportionate amount of the underlying assets held by the system. Users can indicate the preferred ratio of assets returned when withdrawing. In such an instance, a swap is performed and its fee deducted to obtain the correct output amounts.

Fee calculation

Fees are charged on each swap and stored directly into the pool, proportionally distributing them between all LPs. The fee amount is calculated as follows:

`getReturn(x0, x1) = (0.9999 * (x1 - x0) + 3.3827123349983306 * ((x0 - 0.4568073509746632) ** 18 - (x1 - 0.4568073509746632) ** 18)) / (x1 - x0)` where `x0` and `x1` are the proportion of the `from` asset of the total balance before and after the swap, respectively.

This function is the integral of the fee curve over the interval x0 -> x1, which calculates the average value of the fee curve over that interval. Proportional values are used as normalized inputs to the function such that the fee curve can be designed over one fixed range, [0, 1].

Virtual swap calculation

To perform a virtual swap, the contract calculates the optimal amount to swap of a given asset to balance the amounts with the expected ratio. An initial approximation of the swap amount is determined using the assumption that the tokens swap at a 1:1 ratio. In reality, the tokens do not swap at a 1:1 ratio due to fees; hence the approximated amount may require adjustment to find a better approximation. The contract identifies the optimal value iteratively using a binary search approach.

Swap types

- `swap0To1(uint256 inputAmount)` - swaps from asset 0 to asset 1 and sends the swapped assets to `msg.sender`
- `swap0To1For(uint256 inputAmount, address to)` - swaps from asset 0 to asset 1 and sends the swapped assets to `to`

- `swap1To0(uint256 inputAmount)` - swaps from asset 1 to asset 0 and sends the swapped assets to `msg.sender`
- `swap1To0For(uint256 inputAmount, address to)` - swaps from asset 1 to asset 0 and sends the swapped assets to `to`

Helper functions

A `getReturn(IERC20 tokenFrom, IERC20 tokenTo, uint256 inputAmount)` function is exposed to allow users to determine the amount of tokens they will receive for a given swap.

5. Detailed findings

The following section includes in-depth descriptions of the findings of the audit.

5.1 High risk

No identified high-risk issues were present at the conclusion of the audit.

5.2 Medium risk

No identified medium-risk issues were present at the conclusion of the audit.

5.3 Low risk

No identified low-risk issues were present at the conclusion of the audit.

5.4 Informational

5.4.1 Token pairs must have the same number of decimals

Require statements were added to the constructor to revert if the asset types provided had a different number of decimals. This approach would prevent the creation of pools for some asset pairs such as DAI/USDT. While this prevents the risk of miscalculations between tokens with different decimals, a more flexible approach would be to implement logic that correctly scales between the assets based on their number of decimals at all entry and exits points in the contract.

5.4.2 Token transfer fees could lead to incorrect system states

The mainnet contract in scope for the audit used the USDC-USDT pair. USDT has an optional transfer fee that can optionally be enabled by Tether. If this fee were to be enabled, it would lead to inconsistencies, specifically when exchanging from USDT to USDC, as the transfer fee would not be accounted for in the exchanged amount.

The likelihood of the fee being enabled is relatively low, given the far-reaching effect it would have on the ecosystem.

5.4.3 Binary search inflates gas price

The binary search method used to calculate the optimal virtual swap amounts scales the transaction gas price linearly as the virtual swap amount increases. In search of a more efficient approach, the function used to determine the virtual swap amount (let us call this amount `delta`) required to balance the ratios was combined with the `_getReturn(x0, x1)` function to calculate `delta` directly. This produced the following formula:

`[x - delta][yBalance - _getReturn(x0,x1)] - [y + _getReturn(x0,x1)][xBalance + delta] = 0` where `x0` and `x1` are the proportion of the `from` asset of the total balance before and after the swap of `delta` amount, respectively. Since `x`, `y`, `xBalance`, and `yBalance` are all known values, the function could be expanded and solved for `delta`. However, this expansion identified that the `delta` value was one of the intercepts between a quadratic function and an 18th order function that is not trivially solvable without numeric approximation methods, which the binary search solution performs already.

The lack of a direct solution combined with the gas cost being relatively insignificant when compared to the asset amounts required to inflate the gas price meant that the current solution was sufficient and presented a negligible risk to the user.

5.5 Closed

No issues identified during the audit were closed.