1Inch FixedRateSwap Audit

November 2021

By CoinFabrik

Introduction

CoinFabrik was asked to audit the contracts for the 1inch Fixed Rate Swap project.

Summary

The contracts audited are from the GitHub repository at `https://github.com/1inch/fixed-rate-swap/`. The audit is based on the commit `b1600f61b77b6051388e6fb2cb0be776c5bcf2d1`. Fixes were made and the commit 52552c22f88ba3e628479bcb34bca541e9812ab6 was reviewed;

Contracts

The audited contracts are:

- `contracts/FixedRateSwap.sol`: Swap, deposit and withdraw functionalities.

Analyses

The following analyses were performed:

- Misuse of the different call methods
- Integer overflow errors
- Division by zero errors
- Outdated version of Solidity compiler
- Front running attacks
- Reentrancy attacks
- Misuse of block timestamps
- Softlock denial of service attacks
- Functions with excessive gas cost
- Missing or misused function qualifiers
- Needlessly complex code and contract interactions
- Poor or nonexistent error handling
- Failure to use a withdrawal pattern

- Insufficient validation of the input parameters
- Incorrect handling of cryptographic signatures

Severity Classification

Security risks are classified as follows:

- **Critical:** These are issues that we manage to exploit. They compromise the system seriously. They must be fixed **immediately**.
- **Medium:** These are potentially exploitable issues. Even though we did not manage to exploit them or their impact is not clear, they might represent a security risk in the near future. We suggest fixing them **as soon as possible**.
- **Minor:** These issues represent problems that are relatively small or difficult to take advantage of but can be exploited in combination with other issues. These kinds of issues do not block deployments in production environments. They should be taken into account and be fixed **when possible**.
- **Enhancement:** These kinds of findings do not represent a security risk. They are best practices that we suggest to implement.

This classification is summarized in the following table:

SEVERITY	EXPLOITABLE	ROADBLOCK	TO BE FIXED
Critical	Yes	Yes	Immediately
Medium	In the near future	Yes	As soon as possible
Minor	Unlikely	No	Eventually
Enhancement	No	No	Eventually

Issues Found by Severity

Critical Severity Issues

No issues found.

Medium Severity Issues

No issues found.

Minor Severity Issues

No issues found.

Enhancements

EN-01 Division by Zero in withdrawFor()

Require that `totalSupply() > 0` when calling `withdrawFor()`. Although the function should not be called in this case, the division by 0 error warning could be misleading and may be avoided with the requirement.

EN-02 Documentation Enhancements and Errors

- The shares produced in `deposit()/depositFor()` and the amounts returned via `withdrawForWithRatio()` are incompletely documented in the comments. It is briefly described that amounts are adjusted to make fair deposits and capture fees or that proportional withdrawals include extra swap afterwards to get to the specified ratio with no additional information. Including equations would help users in understanding the contract, including, for example, the possible maximum fee.
- The `getReturn()` comment/documentation reads:

```
getReturn(x) = 0.9999 + (0.5817091329374359 - x *
      1.2734233188154198)^17
```

However this is erroneous, the correct statement is

```
getReturn(x) = int_[x_0,x_1] f(x)
f(x) =(0.9999 + (0.5817091329374359 - x * 1.2734233188154198)^17)
x0 = _ONE * fromBalance / (fromBalance + toBalance);
x1 = _ONE * (fromBalance + inputAmount) / (fromBalance + toBalance);
```

- Also, update the value of the constants C2 and C3 in the comments for `getReturn()` to reflect their actual values.

Other Considerations

Front-running attack in Interaction Functions

During the audit the development team found a front running issue in interaction functions (deposit(), depositFor(), withdrawWithRatio(), withdrawForWithRatio()) that could allow an attacker to modify supply and therefore reduce the tokens deposited/withdrawn by the front-runned user.

The issue was fixed by adding minima to the interaction functions thus allowing users to establish the minimum number of tokens they accept.

Conclusion

We found the contracts to be simple and straightforward. A security vulnerability was found and fixed. The proposed enhancements were taken into account and documentation has been improved.

Disclaimer: This audit report is not a security warranty, investment advice, or an approval of the 1inch Fixed Rate Swap project since CoinFabrik has not reviewed its platform. Moreover, it does not provide a smart contract code faultlessness guarantee.