1inch MoneyMarket Audit

March 2022

By CoinFabrik

Introduction

CoinFabrik was asked to audit the contracts for the MoneyMarket project. First we will provide a summary of our discoveries and then we will show the details of our findings.

Money Market is a lending pool-based protocol allowing users to borrow assets from a liquidity pool without requiring the matching of individual lenders and borrowers, and further providing interests on assets and debt through ERC20 tokens.

Scope

The contracts audited are from the https://github.com/1inch/money-market-protocol git repository. The audit is based on the commit `98f1553d0acf419a4b09d5b492c59b4ccdcf4af5`.

The audited contracts are:

- `contracts/interfaces/deployers/IDebtokenDeployer.sol`
- `contracts/interfaces/deployers/ILentokenDeployer.sol`
- `contracts/interfaces/deployers/ISupTokenDeployer.sol`
- `contracts/interfaces/deployers/IWethLentokenDeployer.sol`
- `contracts/interfaces/tokens/IDebtoken.sol`
- `contracts/interfaces/tokens/ILentoken.sol`
- `contracts/interfaces/tokens/IMoneyMarketToken.sol`
- `contracts/interfaces/tokens/ISuptoken.sol`
- `contracts/interfaces/tokens/IWETH.sol`
- `contracts/interfaces/IFormula.sol`
- `contracts/interfaces/IMoneyMarket.sol`
- `contracts/interfaces/IPriceOracle.sol`
- `contracts/libs/Constants.sol`
- `contracts/libs/MovingPrices.sol`
- `contracts/libs/Underlying.sol`
- `contracts/tokens/Debtoken.sol`
- `contracts/tokens/Lentoken.sol`
- `contracts/tokens/MoneyMarketToken.sol`
- `contracts/tokens/Suptoken.sol`
- `contracts/tokens/WethLentoken.sol`
- `contracts/ChainlikOracleAdapter.sol`

- `contracts/Formula.sol`
- `contracts/MoneyMarket.sol`
- `contracts/NaiveFeedRegistriy.sol`
- `contracts/UniswapV3OracleAdalter.sol`

The scope of the audit is limited to those files. No other files in this repository were audited. Its dependencies are assumed to work according to their documentation. Also, no tests were reviewed for this audit.

Analyses

Without being limited to them, the audit process included the following analyses:

- Arithmetic errors
- Outdated version of Solidity compiler
- Race conditions
- Reentrancy attacks
- Misuse of block timestamps
- Denial of service attacks
- Excessive gas usage
- Missing or misused function qualifiers
- Needlessly complex code and contract interactions
- Poor or nonexistent error handling
- Insufficient validation of the input parameters
- Incorrect handling of cryptographic signatures
- Centralization and upgradeability

Summary of Findings

We found no security issues. Some enhancements were proposed.

The documentation which was made available for this audit does not cover the MoneyMarket protocol operations (borrow, withdraw, etc) and is lacking in technical detail. Consequently, this audit has covered security issues but has not considered attacks circumventing (undocumented) restrictions for these operations.

Security Issues Found

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**.

Issues Status

An issue detected by this audit can have four distinct statuses:

- **Unresolved**: The issue has not been resolved.
- **Acknowledged**: The issue remains in the code but is a result of an intentional decision.
- **Resolved**: Adjusted program implementation to eliminate the risk.
- **Partially resolved**: Adjusted program implementation to eliminate part of the risk. The other part remains in the code but is a result of an intentional decision.
- **Mitigated**: Implemented actions to minimize the impact or likelihood of the risk

Critical Severity Issues

No issues found.

Medium Severity Issues

No issues found.

Minor Severity Issues

No issues found.

Enhancements

These items do not represent a security risk. They are best practices that we suggest implementing.

ID	Title	Status
EN-01	Banned Token Creation Is Possible	Not implemented
EN-02	Empty Call to finishWithdawalTo()	Not implemented
EN-03	Insufficient Documentation	Not implemented
EN-04	TODOs Best Practice	Not implemented
EN-04	Old Solidity Compiler Version	Not implemented

Enhancements Details

EN-01 Banned Token Creation Is Possible

Location:

- contracts/tokens/LenToken.sol:45-68

In the LenToken.sol constructor it is possible to have an input with a `tokenFactor` with `reserveRatio` equal to `1e9` which is equivalent to this token being banned.

Recommendation

Consider requiring the `reserveRatio` to be different from `1e9`.

Status

Not implemented.

EN-02 Empty Call to finishWithdawalTo()

The public function `finishWithdrawalTo()` can be called by an arbitrary user when the time is right, or may be called from `startWithdrawal()`. In either case, the result of `_shareToAmount(withdrawal.share)` could be 0 and the function would call for transferring 0 1inch tokens.

Recommendation
Check that the amount to be transferred is nonzero before calling the function. In particular, note that the check amount>0 in startWithdrawal() could be insufficient to determine whether `_shareToAmount(withdrawal.share) >0`.

Status
Not implemented.

EN-03 Insufficient Documentation
The documentation does not cover most use cases and is lacking in detail.

Recommendation
Add documentation describing the public/external operations including use cases and requirements. In particular, this applies to lending, borrowing, liquidation, money market creation and other operations.

Status
Not implemented.

EN-04 TODOs Best Practice
Remove 'todo's from the production version. These might be read as poor quality by the unskilled reader.

Recommendation
Remove ‘todo’s from the repository.

Status
Not implemented.

EN-05 Old Solidity Compiler Version
The contract `UniswapV3OracleAdapter.sol` uses pragma solidity 0.7.6 while most contracts use solidity 0.8.12.

Recommendation
Ensure all code runs with the same compiler version.

Status
Not implemented.

Changelog

- 2022-03-16 – Initial report based on commit `98f1553d0acf419a4b09d5b492c59b4ccdcf4af5`.

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