SOLIDIFIED

Audit Report for 1inch Money Market - March 28, 2022

Summary

Audit Report prepared by Solidified covering the 1inch Money Market smart contracts.

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the code below. The final debrief took place on March 28, 2022, and the results are presented here.

UPDATE: Team comments were received on April 5, 2022, and the report was updated accordingly.

Audited Files

The source code has been supplied in a private source code repository:

https://github.com/1inch/money-market-protocol

Commit number: `b14086086ec2d5c253eb1933eae371c9797b5960`

File List:

```
./contracts
├── ChainlinkOracleAdapter.sol
├── Formula.sol
├── MoneyMarket.sol
├── NaiveFeedRegistry.sol
├── UniswapV3OracleAdapter.sol
├── interfaces
│   ├── IFormula.sol
│   ├── IMoneyMarket.sol
│   ├── IPriceOracle.sol
│   ├── deployers
│   │   ├── IDebtokenDeployer.sol
│   │   ├── ILentokenDeployer.sol
│   │   ├── ISuptokenDeployer.sol
│   │   └── IWethLentokenDeployer.sol
│   └── tokens
│       ├── IDebtoken.sol
│       ├── ILentoken.sol
│       ├── IMoneyMarketToken.sol
│       ├── ISuptoken.sol
│       ├── IWETH.sol
│       └── IWethLentoken.sol
├── libs
│   ├── Constants.sol
│   ├── Flagged.sol
│   ├── MovingPrices.sol
│   └── Underlying.sol
└── tokens
    ├── Debtoken.sol
    ├── Lentoken.sol
    ├── MoneyMarketToken.sol
    ├── Suptoken.sol
    └── WethLentoken.sol
```

Intended Behavior

1inch Money Market is a lending pool-based protocol where anyone can add new tokens without requiring owner permission.

Findings

Smart contract audits are an important step to improve the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of a smart contract system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**.

Note, that high complexity or lower test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than a security audit and vice versa.

Criteria	Status	Comment
Code complexity	Medium	-
Code readability and clarity	High	-
Level of Documentation	High	-
Test Coverage	High	-

Issues Found

Solidified found that the 1inch Money Market contracts contain no critical issues, 1 major issue, 0 minor issues, and 2 informational notes.

We recommend issues are amended, while informational notes are up to the team's discretion, as they refer to best practices.

Issue #	Description	Severity	Status
1	MoneyMarket.sol: Contract owner can drain the entire contract by assigning a malicious oracle	Major	Acknowledged
2	Lentoken.sol: Token withdraw failures revert without meaningful error messages	Note	Acknowledged
3	Suptoken.sol: Gas could be saved in function startWithdrawal() by declaring variable withdrawal as storage instead of memory	Note	Acknowledged

Critical Issues

No critical issues have been found.

Major Issues

1. `MoneyMarket.sol`: Contract owner can drain the entire contract by assigning a malicious oracle

Function `setPriceOracle()` allows the contract owner to potentially assign a malicious oracle that would allow them (or an attacker) to drain the entire contract.

Recommendation

`setPriceOracle()` should not be able to immediately reassign the oracle, but rather give market participants adequate time to close their positions (in case they wish to) before a new oracle is assigned.

Note

We also highly recommend giving market participants ample time to settle their positions before any of the other market parameters are modified.

Status

Acknowledged. Team's response: "*Noted. Contract owner will be multisig with timelock*".

Minor Issues

No minor issues have been found.

Informational Notes

2. Lentoken.sol: Token withdraw failures revert without meaningful error messages

All token withdraw failures do not give any meaningful error messages back to the user. In case they have insufficient balances for instance, they will only get the obscure “burn amount exceeds balance” error, which might not make much sense to them in the current context.

Recommendation

Check if the account does not have enough tokens to withdraw, then revert if true with a meaningful error message.

Status

Acknowledged. Team’s response: “*Won’t fix. We think that revert reason is meaningful enough to understand that there is not enough balance*”.

3. Suptoken.sol: Gas could be saved in function startWithdrawal() by declaring variable withdrawal as storage instead of memory

Declaring the variable `withdrawal` as `storage` would prevent the compiler from having to copy the contents of `PendingWithdrawal` from storage to memory each time the variable is assigned, hence saving the users on gas.

Recommendation

Declare `withdrawal` as a `storage` variable in function `startWithdrawal()`.

Status

Acknowledged. Team's response: "*Won't fix. In fact it increases gas usage by 1 gas*".

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of 1inch or its products. This audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

The individual audit reports are anonymized and combined during a debrief process, in order to provide an unbiased delivery and protect the auditors of Solidified platform from legal and financial liability.

Oak Security GmbH