SOLIDIFIED

Audit Report for ICHI oneToken - May 3, 2021

Summary

Audit Report prepared by Solidified covering the ICHI oneToken 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 May 3, 2021, and the results are presented here.

Audited Files

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

https://github.com/ichifarm/ichi-oneToken/

Commit number: `09fca7431a18028779d36427ab3ccdf947e43d40`

Intended Behavior

ICHI is a DAO (Decentralized Autonomous Organization) governed by the community of ICHI token holders. ICHI provides a stablecoin service to other crypto communities through its oneToken Factory.

The oneToken Factory enables the creation of oneTokens (ERC20 tokens collateralized by a blend of stablecoins and member coins, designed to always maintain a peg to one USD). It is a generalized contract designed to produce upgradeable oneToken instances each with distinct governance and internal logic within boundaries defined by ICHI.

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	Low	-
Code readability and clarity	High	-
Level of Documentation	High	-
Test Coverage	High	-

Issues Found

Solidified found that the ICHI oneToken contracts contain no critical issues, 1 major issue, 6 minor issues, and 5 informational notes.

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

#	Description	Severity	Status
1	ICHICompositeOracle.sol: Incorrect implementation for oracle function amountRequired()	Major	Pending
2	StrategyCommon.sol: Function closeAllPositions() will always fail after a large enough number of assets have been added	Minor	Pending
3	Arbitrary.sol: Function executeTransaction() does not refund extra ETH sent	Minor	Pending
4	UniswapOracleSimple.sol: Oracle does not enforce that indexToken is a USD stablecoin	Minor	Pending
5	OneTokenV1Base.sol: Function removeStrategy() does not close the strategy	Minor	Pending
6	Incremental.sol: Zero value logic mismatch for stepSize	Minor	Pending
7	OneTokenV1.sol: Function redeem() does not check if the liabilities of the collateral token exceed the contrat's balance, which could prevent the user from withdrawing the collateral token afterwards	Minor	Pending
8	OneTokenV1.sol: Two-step withdrawal process is redundant	Note	-
9	OneTokenV1Base.sol: Redundant call to _transferOwnership() in init()	Note	-
10	OneTokenV1.sol: Function redeem() assumes one to one ratio between collateral token and oneToken	Note	-

11	OneTokenV1.sol: Collateral tokens which charge transfer fees are not compatible with the accounting of collateral balances	Note	-
12	OneTokenV1Base.sol: setStrategyAllowance() amount parameter documentation is incorrect	Note	-

Critical Issues

No critical issues have been found.

Major Issues

1. ICHICompositeOracle.sol: Incorrect implementation for oracle function amountRequired()

Function `amountRequired()` incorrectly returns a value that is 1:1 pegged to the given `amountUsd`. This results in `ICHICompositeOracle` always returning the wrong result for any given `amountUsd`.

Recommendation
Have the function fetch all the pairs from the `oracleContracts` oracle chain (in their respective order), then use these values to correctly compute `amountUsd`.

Minor Issues

2. StrategyCommon.sol: Function closeAllPositions() will always fail after a large enough number of assets have been added

Since function `closeAllPositions()` iterates over all `OneTokenV1Base` assets, its gas consumption will exceed the block gas limit after a large enough number of assets have been added. This will cause the function to always fail when called.

Recommendation

Implement a cap on the number of assets that can be added to `OneTokenV1Base` that won't exceed the block gas limit.

3. Arbitrary.sol: Function executeTransaction() does not refund extra ETH sent

Function `executeTransaction()` does not refund any extra ETH sent that exceeds the `value` parameter.

Recommendation
Either refund ETH that exceeds `value` or eliminate `value` altogether and use `msg.value` instead.

4. UniswapOracleSimple.sol: Oracle does not enforce that indexToken is a USD stablecoin

The documentation states that `indexToken` must be a USD token, however, the contract's constructor implementation does not enforce that.

Recommendation
Create a whitelist of ICHI supported USD tokens and have the constructor verify that the given `indexToken` address is in the whitelist.

Note
Same issue is also present in `ICHICompositeOracle`.

5. OneTokenV1Base.sol: Function removeStrategy() does not close the strategy

The documentation of function `removeStrategy()` states that it should close the strategy, however this is not implemented.

Recommendation
Have the function call `closeStrategy()`.

6. Incremental.sol: Zero value logic mismatch for stepSize

The value of `stepSize` can be assigned to zero in `setParams()`, but is not allowed to be so in `setStepSize()`.

Recommendation
Resolve mismatch between the two functions.

7. OneTokenV1.sol: Function redeem() does not check if the liabilities of the collateral token exceed the contrat's balance, which could prevent the user from withdrawing the collateral token afterwards

Function `redeem()` does not check if the liabilities of the collateral tokens exceed contrat's balance. For instance, in case several collateral tokens are used, the user calling `redeem()` for certain collateral will not be informed if the collateral is not available to withdraw.

Informational Notes

8. OneTokenV1.sol: Two-step withdrawal process is redundant

Although the current two-step withdrawal process does indeed protect from flash loan attacks, there is virtually very little that can be done with a flash loan that cannot be done with an attacker that has a large amount of liquidity (e.g. a whale). In that case, the two-step withdrawal process will be of little help.

Furthermore, the current implementation of `UniswapOracleSimple` should not suffer from these kinds of attacks. `ICHICompositeOracle` should be safe as well, provided that all `oracleContracts` are either `UniswapOracleSimple` or `ICHIPeggedOracle` contracts.

Recommendation
Eliminate two-step withdrawal to improve user experience.

9. OneTokenV1Base.sol: Redundant call to _transferOwnership() in init()

Function `init()` calls both `initOwnable()` and `_transferOwnership()`, which is redundant.

Recommendation
Remove the call to `_transferOwnership()`.

10. OneTokenV1.sol: Function redeem() assumes one to one ratio between collateral token and oneToken

Arbitrage opportunities might exist when oneToken is not on an exact 1:1 peg with its collateral.

11. OneTokenV1.sol: Collateral tokens which charge transfer fee are not compatible with the accounting of collateral balances

The actual collateral balances in contract will be lower than accounted for in case a collateral token charges a transfer fee.

12. OneTokenV1Base.sol: setStrategyAllowance() amount parameter documentation is incorrect

The `amount` parameter is incorrectly documented.

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of ICHI oneToken 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.

Solidified Technologies Inc.