SOLIDIFIED

Audit Report for Alongside - May 12, 2022

Summary

Audit Report prepared by Solidified covering the Alongside smart contracts.

Process and Delivery

Independent Solidified experts performed an unbiased and isolated audit of the code below. The final debrief took place on April 18, 2022, and the results are presented here.

Audited Files

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

https://github.com/Alongside-Finance/index-contracts/tree/main/src/contracts

Commit number: `5ec49fe6f6e1920b085e89c054bd54e418c9430f`

Audited file list:

```
./contracts
├── controller
│   ├── Controller.sol
│   └── ControllerInterface.sol
├── factory
│   ├── Factory.sol
│   ├── FactoryInterface.sol
│   ├── FactoryV2.sol
│   ├── Members.sol
│   ├── MembersInterface.sol
├── token
│   └── IndexToken.sol
└── utils
    └── IndexedMapping.sol
```

Intended Behavior

Alongside aims to create an on-chain single asset index (index token) that tracks and is fully backed by a basket of crypto assets.

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 Alongside contracts contain no critical issues, no major issues, 2 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	Migrate.sol: Contract deployment could potentially fail when there are too many merchants	Minor	Resolved
2	Upgraded factories can potentially corrupt state	Minor	Resolved
3	IndexedMapping.sol: Contract could save on gas by removing the valueExists mapping	Note	-
4	Factory.sol / FactoryV2.sol: Argument depositAddress is redundant in function addMintRequest()	Note	-
5	Incorrect Comment	Note	-

Critical Issues

No critical issues have been found.

Major Issues

No major issues have been found.

Minor Issues

1. Migrate.sol: Contract deployment could potentially fail when there are too many merchants

The `MigrateFactory` constructor iterates over every single element in the `merchants` array. If the array gets large enough, this operation will exceed the block gas limit and eventually fail.

Recommendation
Consider adding a limit on the number of merchants that will not exceed the block gas limit.

2. Upgraded factories can potentially corrupt state

When the protocol upgrades its factory, there is nothing on-chain that guarantees that the new factory has an identical state to the old factory. This can cause issues such as mint/burn requests, custodian deposit addresses and merchant deposit addresses potentially getting lost or corrupt.

Recommendation
Consider storing all the factory's state in a separate contract, then use the *proxy pattern* to `delegatecall` all factory functions from within this contract. This way the factory code could be upgraded without having to risk introducing state migration discrepancies.

Informational Notes

3. IndexedMapping.sol: Contract could save on gas by removing the valueExists mapping

---

The `IndexedMapping` library could save on gas by removing the `valueExists` mapping and using a 1-based index for the `valueIndex` mapping. This would allow the library to determine if a value exists simply by it having an index that is greater than zero.

It's worth noting that we recommend implementing this in a way that is completely transparent to the other contracts. This means that contracts using the library should still be using a zero-based index, and the conversion to a 1-based index would only happen internally in the library's own functions.

4. Factory.sol / FactoryV2.sol: Argument depositAddress is redundant in function addMintRequest()

The `depositAddress` argument can always be retrieved by calling `custodianDepositAddress[msg.sender]`, and is therefore not required to be provided by the caller.

Recommendation
Consider removing the `depositAddress` argument.

5. Factory.sol: Incorrect Comment

In function `addMintRequest()` there is a mismatch between actual return type (`uint256`) and its NatSpec documentation (`bool`)

Recommendation

Consider correcting the comment.

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of Alongside 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