SOLIDIFIED

Audit Report for Betoken. May 20, 2019.

Summary

Audit Report prepared by Solidified for Betoken, their fund manager and token smart contracts (and their associated components).

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the below smart contracts. The debrief took place on May 20, 2019, and the final results are presented here.

Audited Files

```
BetokenFund.sol
BetokenLogic.sol
BetokenProxy.sol
Migrations.sol
Utils.sol
CompoundOrder.sol
CompoundOrderFactory.sol
CompoundOrderLogic.sol
LongCERC20OrderLogic.sol
LongCEtherOrderLogic.sol
ShortCERC20OrderLogic.sol
ShortCEtherOrderLogic.sol
CERC20.sol
CEther.sol
Comptroller.sol
IMiniMeToken.sol
KyberNetwork.sol
PositionToken.sol
PriceOracle.sol
MiniMeToken.sol
TokenController.sol
```

The audit was based on commit `8878b16dda3761d79bf3181fb80971c6e48cfbad` and Solidity compiler version `0.5.0`.

SOLIDIFIED

Audit Report for Betoken. May 20, 2019.

Intended Behavior

The purpose of these contracts is to implement a decentralized fund management system

Issues Found

Critical

1. Fund shareholders can potentially lose a significant amount of their assets to the developer if managers don't sell

---

Functions `sellLeftoverToken()` and `sellLeftoverCompoundOrder()` allow anyone to send the developer all unsold investments during the previous Manage cycle. If for any reason (malicious or otherwise) a manager does not sell their investments, the Betoken fund shareholders can potentially lose a significant amount of their assets to the developer.

Recommendation

Sell all unsold assets and compound orders at the end of each Manage cycle, then add the amount to `totalFundsInDAI`. This way it gets distributed proportionally to all shareholders once they withdraw their shares.

After the Manage cycle has ended and investments have been sold, `sellLeftoverToken()` should now sell any given token (if supported by Kyber) and add the amount to `totalFundsInDAI`. This is intended as a way for shareholders to make use of any token sent directly to the contract (either by mistake or otherwise) instead of the funds being lost.

Amended [28.05.2019]

Issue was fixed by Betoken and is no longer present in commit

`b3ee70c8d4f7daa64ffe80ea1469a76236475380`

Major

2. `__register()` specs mismatch

During the intermission cycle, the `__register()` function transfers the received DAI to developer's wallet instead of the fund. There is no reference in the white paper to support this, thus shareholders will unknowingly be losing out on all registration DAI received during the intermission phase.

Recommendation

Have `__register()` always add received DAI to `totalFundsInDAI`

Amended [28.05.2019]

Issue was fixed by Betoken and is no longer present in commit

`b3ee70c8d4f7daa64ffe80ea1469a76236475380`

3. Constructor accesses CDAI before initialization

The constructor compares `_compoundTokenAddr` to address(CDAI) before CDAI has been initialized.

Recommendation

Initialize CDAI first before executing this instruction.

Amended [28.05.2019]

Issue was fixed by Betoken and is no longer present in commit

`b3ee70c8d4f7daa64ffe80ea1469a76236475380`

Minor

4. External calls should be wrapped in require statements

All calls to external contracts, specifically ERC20 contracts, should be wrapped in require statements because token contracts can return `false` instead of reverting.

Recommendation

Wrap external calls in require statements.

Amended [28.05.2019]

Issue was fixed by Betoken and is no longer present in commit

`b3ee70c8d4f7daa64ffe80ea1469a76236475380`

5. Fallback function should call `depositEther()`

The contract's payable fallback function should call `depositEther()` so that users don't lose their ETH in case they send it directly to `BetokenFund`.

Betoken's Response:
This is tricky to fix, because BetokenFund receives Ether from KyberNetwork when it makes a trade that converts an asset into Ether.It could be fixed by checking if (msg.sender == address(kyberNetwork)), but KyberNetworkProxyInterface does not provide access to the address of the current KyberNetwork. In addition, selling LongCEtherOrder also sends Ether to BetokenFund. The cost of adding the fix outweighs the potential benefit. Decision: no change.

Solidified agrees with Betoken team decision

6. Contract specs mismatch

The white paper specifies a maximum of 6 manager-initiated votes, while the contract only allows for 5, as arrays proposers and candidates have a maximum length of 5 elements.

Recommendation

Fix mismatch by updating: `proposers`, `candidates`, `forVotes`, `againstVotes`, `managerVotes`, and `__isValidChunk()` in both `BetokenLogic` and `BetokenFund`.

Amended [28.05.2019]

Issue was fixed by Betoken and is no longer present in commit

`b3ee70c8d4f7daa64ffe80ea1469a76236475380`

7. Token specs mismatch

The Betoken white paper specifies that Kairo is a non-transferable token, while the deployment scripts deploy it as a regular MiniMeToken and do not disable transfers.

Recommendation

Create a separate token contract for Kairo that explicitly disables transfers (can still be inherited from `MiniMeToken` with the `transfer` and `transferFrom` functions overridden).

Amended [28.05.2019]

Issue was fixed by Betoken and is no longer present in commit

`b3ee70c8d4f7daa64ffe80ea1469a76236475380`

Notes

8. Consider adding error messages in require statements

In many places require is used without suitable error strings. Current versions of Solidity support error messages and their use is highly recommended.

9. Beware of frontrunning issues

Due to the nature of the Fund, managers can be susceptible to frontrunning attacks, where third parties insert transactions to take advantage of investments opportunities. This can be done either by other managers to be rewarded with a higher commission, or traders on Kyber watching the network. Unfortunately, there aren't any solutions to this problem, but users need to be aware of it.

10. Consider unifying storage layout

A safer approach would be to unify `BetokenLogic` and `BetokenFund` storage on a third contract that both instances inherit from. It greatly diminishes the chance of storage layout errors.

Amended [28.05.2019]

Issue was fixed by Betoken and is no longer present in commit

`b3ee70c8d4f7daa64ffe80ea1469a76236475380`

11. Proxy contracts can be optimized to have only one function that delegate calls to logic contract

Instead of having individual functions making `delegatecalls` to logic contract, consider implementing the generic proxy pattern, that forwards any received call to the implementation contract.

12. Update Solidity compiler

The most current and up to date version of the Solidity compiler is 0.5.8. It is recommended to always utilize the highest possible version at the time of contract writing because known bugs are patched: https://solidity.readthedocs.io/en/develop/bugs.html. Three Solidity compiler bugs were fixed between the version used in the smart contracts and the current version, none of them are applicable to the in-scope smart contract, and therefore this issue is being reported as a Note.

Amended [28.05.2019]

Issue was fixed by Betoken and is no longer present in commit

`b3ee70c8d4f7daa64ffe80ea1469a76236475380`

13. Duplicate code

Code for the following functions duplicates existing logic found in `BetokenLogic`: `currentChunk()` , `currentSubchunk()` , `getVotingWeight()` , `getTotalVotingWeight()` , and `kairoPrice()` .

Recommendation

Consider removing code duplication if the `BetokenFund` contract is not properly using it.

Amended [28.05.2019]

Issue was fixed by Betoken and is no longer present in commit

`b3ee70c8d4f7daa64ffe80ea1469a76236475380`

14. Unused code

Internal functions `__handleInvestment()` , `__returnStake()` , and `__recordRisk()` are not being called anywhere inside the `BetokenFund` contract code.

Recommendation

Remove the aforementioned functions.

Amended [28.05.2019]

Issue was fixed by Betoken and is no longer present in commit

`b3ee70c8d4f7daa64ffe80ea1469a76236475380`

15. Replace cyclic dependencies with interfaces

It's best practice to create and import contract interfaces when two contracts are codependent instead of creating cyclic dependencies.

Recommendation

Consider creating interfaces for `BetokenProxy` and `BetokenFund` and importing them instead of importing the actual contracts.

Amended [28.05.2019]

Issue was fixed by Betoken and is no longer present in commit

`b3ee70c8d4f7daa64ffe80ea1469a76236475380`

16. No need found for storing betokenFund

There is no need to store `betokenFund` as it can always be derived from `betokenFundAddress`.

Recommendation

Remove `betokenFund` to save on gas costs.

Amended [28.05.2019]

Issue was fixed by Betoken and is no longer present in commit

`b3ee70c8d4f7daa64ffe80ea1469a76236475380`

Audit Report for Betoken. May 20, 2019.

Conclusion

One critical and two major issues were found, along with several minor issues that can affect the desired behavior of the contract. It is recommended that they're addressed before proceeding to contract deployment.

Follow up [28.05.2019]

All issues previously found were fixed and are no longer present in commit `b3ee70c8d4f7daa64ffe80ea1469a76236475380`.

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of the Betoken platform or its products. This audit does not provide a security or correctness guarantee of the audited smart contracts. 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.