CertiK Audit Report for CoFiX

CertiK Audit Report for

CoFiX

Request Date: 2020-10-09

Revision Date: 2020-10-16

Platform Name: EVM

Contents

CertiK Audit Report for CoFiX **1**

Contents **2**

Disclaimer **3**

About CertiK **3**

Executive Summary **4**

Testing Summary **5**

Review Notes **6**

- Introduction 6
- Documentation 7
- Summary 7
- Recommendations 8

Findings **9**

Exhibit 1 **9**

Exhibit 2 **10**

Exhibit 3 **12**

Exhibit 4 **13**

Exhibit 5 **14**

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Verification Services Agreement between CertiK and CoFiX(the “Company”), or the scope of services/verification, and terms and conditions provided to the Company in connection with the verification (collectively, the “Agreement”). This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK’s prior written consent.

About CertiK

CertiK is a technology-led blockchain security company founded by Computer Science professors from Yale University and Columbia University built to prove the security and correctness of smart contracts and blockchain protocols.

CertiK, in partnership with grants from IBM and the Ethereum Foundation, CertiK’s mission of every audit is to apply different approaches and detection methods, ranging from manual, static, and dynamic analysis, to ensure that projects are checked against known attacks and potential vulnerabilities. CertiK leverages a team of seasoned engineers and security auditors to apply testing methodologies and assessments to each project, in turn creating a more secure and robust software system.

CertiK has served more than 100 clients with high quality auditing and consulting services, ranging from stablecoins such as Binance’s BGBP and Paxos Gold to decentralized oracles such as Band Protocol and Tellor. CertiK customizes its engineering tool kits, while applying cutting-edge research on smart contracts, for each client on its project to offer a high quality deliverable. For more information: https://certik.io.

Executive Summary

This report has been prepared for **CoFiX** to discover issues and vulnerabilities in the source code of their **Smart Contract** as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Dynamic Analysis, Static Analysis, and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Testing Summary

SECURITY LEVEL

File	Vulnerabilities
CoFiToken.sol	1 Minor
CoFiXVaultForCNode.sol	1 Minor 3 Informational
CoFiXVaultForLP.sol	0
CoFiXVaultForTrader.sol	1 Informational
CoFiStakingRewards.sol	1 Informational
CoFiXStakingRewards.sol	0
CNodeStakingRewards.sol	0
CoFiXController.sol	1 Informational
CoFiXRouter.sol	0
CoFiXPair.sol	0
CoFiXFactory.sol	0

Smart Contract Audit
This report has been prepared as a product of the Smart Contract Audit request by **CoFiX**.
This audit was conducted to discover issues and vulnerabilities in the source code of **CoFiX**'s smart contract.

TYPE	Smart Contract
SOURCE CODE	https://github.com/Computable-Finance/CoFiX
PLATFORM	EVM
LANGUAGE	Solidity
REQUEST DATE	Oct 09, 2020
DELIVERY DATE	Oct 16, 2020
METHODS	A comprehensive examination has been performed using Dynamic Analysis, Static Analysis, and Manual Review.

Review Notes

Introduction

CertiK team was contracted by the **CoFiX** team to audit the design and implementation of their smart contract.

The audited source code link is:

- Source Code:

  https://github.com/Computable-Finance/CoFiX

  commit da1f3690d714c83055f96940a4f4f10c7edbc520

  Source Code SHA-256 Checksum

  CoFiToken.sol hash

  ffcfca80d659070801099b3e53e3e93b030fd904af478f2b67a79cc2fd781e78

  CoFiXVaultForCNode.sol hash

  9a2c2aa728b8b4297f6da35f793e056ad6c7a944ed7ebad15e2a9038e24768b8

  CoFiXVaultForLP.sol hash

  b2b7996b8af78a521cbf505be0b5ee2a5450d262d0416be33ed95e87321edc7a

  CoFiXVaultForTrader.sol hash

  cc6fb225a8d400d60103c55a64e22c851142a2ed34a72d218121ec87f4a769e1

  CoFiStakingRewards.sol hash

  d83769ca7d40f2b8796a86d0f25deed2873b2c2750f0eb02c0d937ae347d7275

  CoFiXStakingRewards.sol hash

  b61e32421b47cce32e644de6301cb94b1e23091bca69077a851c31d1d2067d0c

  CNodeStakingRewards.sol hash

  da4a54591f9eb8c4e16632b74d3bf5d547fc9e9bbf2ecc8ba6d1f722d4f935bf

  CoFiXController.sol hash
  4f619422b556edf97193ace28116e1a61c15e6b0cb7ec46b0529f0489c0f22c0
  CoFiXRouter.sol hash
  96c7cf8cd3826ee4c1e783241ecfef925bfbcab5f16bc082aed2a78ece17c05c

CoFiXPair.sol hash
5a16f16ee387b5da781a56e1996665ae3ed88715fefc43036d4967acaf035e47
CoFiXFactory.sol hash
3bc7b0f7b34209c6088aff9b699bcf317a7786ce69069d9d389f3dfad763369e

The goal of this audit was to review the Solidity implementation for its business model, study potential security vulnerabilities, its general design and architecture, and uncover bugs that could compromise the software in production.

The findings of the initial audit have been conveyed to the team behind the contract implementations and the source code is expected to be re-evaluated before another round of auditing has been carried out.

Documentation

The sources of truth regarding the operation of the contracts in scope were lackluster and **are something we advise to be enriched to aid in the legibility of the codebase as well as project**. To help aid our understanding of each contract's functionality we referred to in-line comments and naming conventions.

These were considered the specification, and when discrepancies arose with the actual code behaviour, we consulted with the **CoFiX** team or reported an issue.

Summary

The scope for the CoFiX protocol contains the CoFi token, an ERC 20 token with the vault and the staking contracts, plus the swap contracts.

Certain optimization steps that we pinpointed in the source code mostly referred to coding standards and inefficiencies, however **2 minor vulnerabilities were identified during our audit that solely concerns the specification**.

Certain discrepancies between the expected specification and the implementation of it were identified and were relayed to the team, however they pose no type of vulnerability and concern an optional code path that was unaccounted for.

Recommendations

Overall, the codebase of the contracts should be refactored to assimilate the findings of this report, enforce linters and / or coding styles as well as correct any spelling errors and mistakes that appear throughout the code **to achieve a high standard of code quality and security**.

Findings

Exhibit 1

TITLE	TYPE	SEVERITY	LOCATION
DAO needed	Optimization	Minor	CoFiToken.sol, CoFiXVaultForCNode.sol: L54

[MINOR] Description:

According to the code and doc, the CoFiX DAO is not completed yet.
Governance ownership is better to be transferred to the CoFiX DAO when it is implemented.
Since the Cofi Token has no limit and the governance is able to add any contract to minters, governance can do unlimited mint and the investors will be under risk.
Secondly, in vaults the "currentCoFiRate" is calculated based on the "initCoFiRate". But the "initCoFiRate" can be set by governance for different vaults. This can impact the rewards.

Thirdly, since staking CoFi can earn ETH, it is important to control the CoFi mint and CoFi rewards.

Discussions:

We saw the records of Governance Authority Transfer to Multi-Sig Wallet.
Not sure CoFiX will use Multi-Sig Wallet as governor? Or implementing CoFiX DAO and transfer governance to DAO?

(CoFiX - response) The CoFiX DAO is still under design. Currently, we use multi-sig wallet as governor to ensure the security of user assets. Governance ownership will be transferred to the CoFiX DAO in the next stage when the CoFi token is widely distributed. At that point, we will invite CertiK to review the governance model.

Exhibit 2

TITLE	TYPE	SEVERITY	LOCATION
Simplifying Existing Code	Coding Style	Informational	CoFiStakingRewards.sol: L155,L160,L166 CoFiXVaultForCNode.sol: L43,L48,L54,L59,L65

[INFORMATIONAL] Description:

Consider using a modifier to replace the below same codes existing in many functions.

Examples:

- In **CoFiStakingRewards.sol**:

Functions "setGovernance", "setDividendShare" and "withdrawSavingByGov"

require(msg.sender == governance, "CoFiStaking: !governance");

- In **CoFiXVaultForCNode.sol:**

Functions "setGovernance","setCNodePool","setInitCoFiRate","setDecayPeriod" and "setDecayRate".

require(msg.sender == governance, "CVaultForCNode: !governance");

Recommendations:

Consider changing it as following example:

```
 modifier onlyGovernance() {
       require(msg.sender == governance, 'CoFiStaking: !governance');
       _;
   }
```

(CoFiX - response) CoFiStakingRewards and CoFiXVaultForCNode are deployed and running on the main-net now. We could improve the coding style by using onlyGovernance() modifier in the next update.

Exhibit 3

TITLE	TYPE	SEVERITY	LOCATION
Gas consumption	Optimization	Informational	CoFiXVaultForCNode.sol: L22,L23,L25

[INFORMATIONAL] Description:

Variables like “cofiToken”,”factory”,”genesisBlock” change only once, better to define it as immutable to avoid gas consumption.

Recommendations:

Consider changing as follow:

address public immutable cofiToken;

address public immutable factory;

uint256 public immutable genesisBlock;

(CoFiX - resolved) The immutable keyword has been used to reduce gas costs before the main-net release.

Exhibit 4

TITLE	TYPE	SEVERITY	LOCATION
Declare Events	Optimization	Informational	CoFiXVaultForCNode.sol: L54,L59,L65

[INFORMATIONAL] Description:

Several sensitive actions are defined without event declarations.

Examples :

- Functions “setInitCoFiRate”,”setDecayPeriod” and ”setDecayRate”.

Recommendations:

Consider adding events for sensitive actions, and emit it in the function.

(CoFiX - response) CoFiXVaultForCNode is deployed and running on the main-net now. We could add the event in the next update.

Exhibit 5

TITLE	TYPE	SEVERITY	LOCATION
Inconsistent coding style	Coding Style	Informational	CoFiXVaultForTrader.sol: L26, CoFiXController.sol: L21

[INFORMATIONAL] Description:

It’s better to stick to the same coding style.

refer to https://solidity.readthedocs.io/en/v0.6.12/style-guide.html#code-layout

Examples:

- in CoFiXVaultForTrader.sol: uint256 public constant RATE_BASE = 1e18;
- in CoFiXController.sol: uint256 constant public AONE = 1 ether;

Recommendations:

The recommendations outlined here are intended to improve the readability, and thus they are not rules, but rather guidelines to try and help convey the most information through the names of things.

(CoFiX - response) Thanks for pointing out. We could improve there in the next update.