CERTIK

CertiK Audit Report for CoFiX

CertiK Audit Report for

CoFiX

Request Date: 2020-09-12

Revision Date: 2020-09-23

Platform Name: EVM

Contents

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

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	Sep 11, 2020
DELIVERY DATE	Sep 23, 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 6b7b9585368ce51b08013c239a1080689ccf1179

  Source Code SHA-256 Checksum

  CoFiXController.sol hash
  65c1e7a0d1c7c8672d11c73ac9bb8d635d910bfd7a110bc3173d27769e41ac14

  CoFiXERC20.sol hash
  E5ccb2a46e910dda11dfef2e35c0fe0dfbfe47b5c2fe6e95c4118df65e9acc98

  CoFiXFactory.sol hash
  328dab48947245e9ba31098a73cbb42b93d3499f29d6a16734f52ffb8b2a9055

  CoFiXKTable.sol hash
  E4aead1f099565a98a2330742150218529bdeee38e7daefb2e1df4a789080a16

  CoFiXPair.sol hash
  8038498497ef0a69b11139bc996ac49555e97cecc16a6ab6c20dc8af413401f2

  CoFiXRouter.sol hash
  b2a34ca7113ec89a5007ba8959bf1a7a4ad753aebc5602c70e89f2f9b5bb037a

  ICoFiXController.sol hash
  3f8cd6b7f264ec95cc32d2933901a36b2636899b05fe11a2c11356782cab925e

ICoFiXERC20.sol hash
02dee3bc2241e000957547d2b5f773cbc779b3dfde13cc8f640dccd01bc346f9

ICoFiXFactory.sol hash
ea566743310d3d45b64ec7b7d99d5aa6d091fdbca8cc34cc0f165e3ff4d192d6

ICoFiXKTable.sol hash
f2ce95efcce554ccada105979af5c9264e6286f06b14cc024b7d661154f54fa1

ICoFiXPair.sol hash
363d5d2582989de75784fc67ca2d8bdd707c9388bd5d599ac521479ff7d4c6b9

ICoFiXRouter.sol hash
59305c20461f24c80cf2db88f84deb3840c6dacef61cfa8b7b09e945084cf28b

IERC20.sol hash
394ea8a43e5557b3cd3e3c8fd94157eef2e82b85ed3f67da10e8472b3822cbd0

INest_3_OfferPrice.sol hash
0c6c2034f211343f2548302464ba982a8fcc66bfc508e9c4a178bda1d347a71f

INest_3_VoteFactory.sol hash
5db97686b23d1e41676677c2018b62d420bd75297ea5aa0f1e59c30f8b7b8dff

IWETH.sol hash
a827f0aea23e710c6ccab2f87ff19e46832f7e4d32b1165b3655cfdc709fc29d

ABDKMath64x64.sol hash
ed8d35d2204b33d401531b326d53ec57bf2dd67b10ae7d5bd35eb39e76a0fcae

SafeMath.sol hash
721aa0bd0b67c797b6c46dc2a042c98d42bdb577d75e9944da772d860839f1e0

TransferHelper.sol hash
ebbd04fbcefb537b3202ba2bb5d94dcfb444632616fce8d05b021e81ac70037c

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 codebase of the project is a typical ERC implementation and the locking mechanism of the token is derived from an officially recognized library, specifically from OpenZeppelin.

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
Unlocked Compiler Version Declaration	Language Specific Issue	Informational	CoFiXController.sol, CoFiXERC20.sol, CoFiXFactory.sol, CoFiXKTable.sol, CoFiXPair.sol, CoFiXRouter.sol, ICoFiXController.sol, ICoFiXERC20.sol, ICoFiXFactory.sol, ICoFiXKTable.sol, ICoFiXPair.sol, ICoFiXRouter.sol, IERC20.sol, INest_3_OfferPrice.sol, INest_3_VoteFactory.sol, IWETH.sol, ABDKMath64x64.sol, SafeMath.sol, TransferHelper.sol

[INFORMATIONAL] Description:

The compiler version utilized throughout the project uses the "^" prefix specifier, denoting that a compiler version which is greater than the version will be used to compile the contracts. Recommend the compiler version should be consistent throughout the codebase.

Recommendations:

It is a general practice to instead lock the compiler at a specific version rather than allow a range of compiler versions to be utilized to avoid compiler-specific bugs and be able to identify

ones more easily. We recommend locking the compiler at the lowest possible version that supports all the capabilities wished by the codebase. This will ensure that the project utilizes a compiler version that has been in use for the longest time and as such is less likely to contain yet-undiscovered bugs.

Recommend to Deploy with any of the following Solidity versions:0.6.12

(CoFiX - resolved) The issue is addressed in commit ed20d4720160d04bcc1ae14e482ae1b9bde27419.

Exhibit 2

TITLE	TYPE	SEVERITY	LOCATION
Proper Usage of “public” and “external” type	Coding Style	Informational	CoFiXController.sol: L190, CoFiXPair.sol: L363,L369

[INFORMATIONAL] Description:

“public” functions that are never called by the contract should be declared “external” . When the inputs are arrays “external” functions are more efficient than “public” functions.

Recommendations:

Use the “external” attribute for functions never called from the contract.
(CoFiX - resolved) The issue is addressed in commit ed20d4720160d04bcc1ae14e482ae1b9bde27419.

Exhibit 3

TITLE	TYPE	SEVERITY	LOCATION
Missing checks of zero address	Optimization	Informational	CoFiXERC20.sol: L42, L48, L54, L59 CoFiXPair.sol: L50

[INFORMATIONAL] Description:

Missing checks of zero address in these functions: “_mint()”, ”_transfer()”, “_burn()”, ”_approve()” in CoFiXERC20.sol and ”_safeTransfer()” in CoFiXPair.sol.

Recommendations:

Consider checking the address ‘from’ and address ‘to’, such as:

```
function _transfer(address from, address to, uint value) private {
    require(from != address(0), "transfer from the zero address");
    require(to != address(0), "transfer to the zero address");
    balanceOf[from] = balanceOf[from].sub(value);
    balanceOf[to] = balanceOf[to].add(value);
    emit Transfer(from, to, value);
}
```

Exhibit 4

TITLE	TYPE	SEVERITY	LOCATION
Boolean Equality	Optimization	Informational	CoFiXController.sol: L123

[INFORMATIONAL] Description:

Boolean constants can be used directly and do not need to be compared to true or false.

Recommendations:

Consider changing it as following:
require(callerAllowed[msg.sender], "CoFiXCtrl: caller not allowed");
(CoFiX - resolved) The issue is addressed in commit
ed20d4720160d04bcc1ae14e482ae1b9bde27419.

Exhibit 5

TITLE	TYPE	SEVERITY	LOCATION
Missing security checks and events	Optimization	Minor	CoFiXFactory.sol: L58, L63

[MINOR] Description:

It is better to ensure the sensitive access control actions as setGovernance, setController.

Recommendations:

We recommend adding more security checks, and declaring events.

```
event SetGovernance(address indexed  _new);
event SetController(address indexed  _new);
function setGovernance(address _new) external override {
    require(_new != address(0), "CFactory: governance cannot be zero address");
    require(_new !=  governance, "CFactory: same address of the old governance");
    require(msg.sender == governance, "CFactory: !governance");
    governance = _new;
    emit SetGovernance(_new);
}

function setController(address _new) external override {
    require(msg.sender == governance, "CFactory: !governance");
    require(_new !=  address(0), "CFactory: controller cannot be zero address");
    require(_new !=  controller, "CFactory: same address of the old controller");
    controller = _new;
    emit SetController(_new);
}
```

(CoFiX - resolved) The issue is partially addressed in commit ed20d4720160d04bcc1ae14e482ae1b9bde27419. The events are declared but the checks are still not added.

Exhibit 6

TITLE	TYPE	SEVERITY	LOCATION
Simplifying Existing Code	Optimization	Informational	CoFiXFactory.sol: L59,L64,L69 CoFiXKTable.sol: L20,L26 CoFiXController: L52,L57,L62,L67,L72,L77,L82,L87,L92,L97,L102,L108

[INFORMATIONAL] Description:

Consider using a modifier to replace the below same codes existing in many functions:
require(msg.sender == governance, "CFactory: !governance");

Recommendations:

Consider changing it as following example:

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

(CoFiX - resolved) The issue is addressed in commit ed20d4720160d04bcc1ae14e482ae1b9bde27419.

Exhibit 7

TITLE	TYPE	SEVERITY	LOCATION
Unused Variable	Ineffectual Code	Informational	CoFiXFactory.sol: L81,L85 CoFiXController.sol: L20, L122,L198

[INFORMATIONAL] Description:

Some return variables are declared in functions “append()” and “uint2str()” of CoFiXFactory, and in function “calcVariance()” of CoFiXController.sol, but they are never used.

Function “queryOracle()” in CoFiXController.sol declares an unused return variable “data”.
Not sure the usage of this “data” will be implemented later or not, but so far this is an unused return variable.

State variable ‘THETA_BASE’ is defined in CoFiXController.sol but never used.

Recommendations:

Consider removing the unused variable.
(CoFiX - resolved) The issue is addressed in commit ed20d4720160d04bcc1ae14e482ae1b9bde27419.

Exhibit 8

TITLE	TYPE	SEVERITY	LOCATION
Check zero address	Optimization	Minor	CoFiXFactory.sol: L41

[MINOR] Description:

Check whether the return value of “pair” is zero or not, refer to :

https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/Create2.sol

Recommendations:

Consider changing it as following example:

```
assembly {

    pair := create2(0, add(bytecode, 32), mload(bytecode), salt);

}
require(pair != address(0), "Create2: Failed on createPair");
```

Exhibit 9

TITLE	TYPE	SEVERITY	LOCATION
Incorrect Naming Convention Utilization	Coding Style	Informational	ICoFiXController.sol: L7

[INFORMATIONAL] Description:

Solidity defines a naming convention that should be followed. In general, the following naming conventions should be utilized in a Solidity file:

- Events should be named using the CapWords style
- Local and state variable names should use mixedCase

refer to https://solidity.readthedocs.io/en/v0.4.25/style-guide.html#naming-conventions
Examples:

- Event “newK” in ICoFiXController.sol

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 - resolved) The issue is addressed in commit ed20d4720160d04bcc1ae14e482ae1b9bde27419.

Exhibit 10

TITLE	TYPE	SEVERITY	LOCATION
Keep the interface consistent	Optimization	Informational	CoFiXController.sol: L117,L122

[INFORMATIONAL] Description:

Better to use inheritance to keep the “CoFixController.sol” consistent with its interface.

Recommendations:

Considering changing the contract “CoFiXController” as follows:

```
import “./interface/ICoFiXController.sol”;
contract CoFixController is ICoFiXController{
        event newK(...); // delete this line.
        function addCaller(address caller) external override{...}
        function queryOracle(...) external payable override returns(...){}
}
```

(CoFiX - resolved) The issue is addressed in commit ed20d4720160d04bcc1ae14e482ae1b9bde27419.

Exhibit 11

TITLE	TYPE	SEVERITY	LOCATION
Misleading Error Message	Optimization	Informational	CoFiXController.sol: L118

[INFORMATIONAL] Description:

Misleading error message in function “addCaller()” of CoFiXController.

require(msg.sender == factory || msg.sender == governance, "CoFiXCtrl: only factory");

Recommendations:

Consider changing the error message with clear reason.
require(msg.sender == factory || msg.sender == governance, "CoFiXCtrl: only factory, governance can add caller");
(CoFiX - resolved) The issue is addressed in commit ed20d4720160d04bcc1ae14e482ae1b9bde27419.

Exhibit 12

TITLE	TYPE	SEVERITY	LOCATION
Incorrect Order of Layout Utilization	Coding Style	Informational	CoFiXPair.sol: L17~L67

[INFORMATIONAL] Description:

Solidity defines an Order of Layout that should be followed. In general, inside each contract, library or interface, use the following order:

1. Type declarations
2. State variables
3. Events
4. Functions

Recommendations:

Consider to follow the rules from solidity's style guide.Refer to
https://solidity.readthedocs.io/en/v0.6.11/style-guide.html#order-of-layout
https://solidity.readthedocs.io/en/v0.6.11/style-guide.html#order-of-functions
(CoFiX - resolved) The issue is addressed in commit
ed20d4720160d04bcc1ae14e482ae1b9bde27419.

Exhibit 13

TITLE	TYPE	SEVERITY	LOCATION
Missing Definitions of Events	Optimization	Informational	CoFiXFactory.sol: L77, CoFiXController.sol: L51,L56,L61,L66,L71,L76,L81,L86,L91,L96,L101

[INFORMATIONAL] Description:

Better to add an event to track the change of sensitive information.

Recommendations:

We recommend declaring events for most of “set” functions in “CoFiXController.sol” and “CoFiXFactory.sol”, such as :
“setOracle()”, “setNestToken()” and so on.
(CoFiX - resolved) The issue is addressed in commit ed20d4720160d04bcc1ae14e482ae1b9bde27419.