Coinwind Audit Report

Version 1.0.0

Serial No. 2021040600022019

Presented by Fairyproof

April 6, 2021

01. Introduction

This document includes the results of the audit performed by the Fairyproof team on the coinwind project, at the request of the coinwind team.

The audited code at the time of writing has not been open source yet. The initial code received by the Fairyproof team before the audit was started contained three contract files: ControllerHub.sol, HubPool.sol and StrategyMdex.sol. The calculated sha256 values for these three initial files were as follows:

```
contracts/ControllerHub.sol:
0x7a3432c1c53ee27428861ac3deec878b02c1448a32d63693422916d7ce90cd99
contracts/HubPool.sol:
0x16d63acc8e864a8b941d07d6d45201b8bd9429c9e7886a7abea533161e7a166b
contracts/StrategyMdex.sol:
0x2f7b47df74b7340a27c14f365b4ffcd944411bbf90a4af0ca5a4623c86fd7e5b
```

After the audit is done the calculated sha256 values for the modified files are as follows:

```
contracts/ControllerHub.sol:
0x45be44da466646c1880e72ea81af8d434bfa5e350244e8012c9d4dafd26cf7c9
contracts/HubPool.sol:
0x6f8be0b23fd0ec4984234de07b27d9d711859ef3e1dd04b8d4b88ca1c115d833
contracts/StrategyMdex.sol:
0x10eb1a3a442095a6a678433ecca63ed61b417068eb55497113a0620bb45f02df
```

The goal of this audit is to review coinwind's solidity implementation for an aggregator service for DeFi investors, study potential security vulnerabilities, its general design and architecture, and uncover bugs that could compromise the software in production.

We make observations on specific areas of the code that present concrete problems, as well as general observations that traverse the entire codebase horizontally, which could improve its quality as a whole.

— Disclaimer

Note that as of the date of publishing, the contents of this report reflect the current understanding of known security patterns and state of the art regarding smart contract security. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk.

The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. Risks or issues introduced by using data feeds from offchain sources are not extended by this review either.

Given the size of the project, the findings detailed here are not to be considered exhaustive, and further testing and audit is recommended after the issues covered are fixed.

To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services.

FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

— Methodology

Coinwind's codebase was studied in detail in order to acquire a clear impression of how the its specifications were implemented. The codebase was then subject to deep analysis and scrutiny, resulting in a series of observations. The problems and their potential solutions are discussed in this document and, whenever possible, we identify common sources for such problems and comment on them as well.

The Fairyproof auditing process follows a routine series of steps:

1. Code review that includes the following
   i. Review of the specifications, sources, and instructions provided to Fairyproof to make sure we understand the size, scope, and functionality of the project's smart contracts.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Fairyproof describe.
2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run the test cases.
   ii. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

— Structure of the document

This report contains a list of issues and comments on all three contract files initially received by the Fairyproof team, which are as follows:

```
contracts/ControllerHub.sol:
0x7a3432c1c53ee27428861ac3deec878b02c1448a32d63693422916d7ce90cd99
contracts/HubPool.sol:
0x16d63acc8e864a8b941d07d6d45201b8bd9429c9e7886a7abea533161e7a166b
contracts/StrategyMdex.sol:
0x2f7b47df74b7340a27c14f365b4ffcd944411bbf90a4af0ca5a4623c86fd7e5b
```

Each issue is assigned a severity level based on the potential impact of the issue and recommendations to fix it, if applicable. For ease of navigation, an index by topic and another by severity are both provided at the beginning of the report.

— Documentation

For this audit, we used the following sources of truth about how the coinwind system should work:


```
contracts/ControllerHub.sol:
0x7a3432c1c53ee27428861ac3deec878b02c1448a32d63693422916d7ce90cd99
contracts/HubPool.sol:
0x16d63acc8e864a8b941d07d6d45201b8bd9429c9e7886a7abea533161e7a166b
contracts/StrategyMdex.sol:
0x2f7b47df74b7340a27c14f365b4ffcd944411bbf90a4af0ca5a4623c86fd7e5b
```


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

— Comments from Auditee

No vulnerabilities with medium severity were found in the coinwind's codebase. Two vulnerabilities with critical severity, one vulnerability with high severity and two vulnerabilities with low severity were fixed by the team. Two vulnerabilities with low severity were acknowledged by the team, and the team doesn't think they will trigger issues or risks and may make changes in future upgrades.

The coinwind's codebase passed the audit performed by the Fairyproof team.

02. About Fairyproof

Fairyproof is a leading technology firm in the blockchain industry, providing consulting and security audits for organizations. Fairyproof has developed industry security standards for designing and deploying smart contract systems.

03. Severity level reference

Every issue in this report was assigned a severity level from the following:

Critical severity issues need to be fixed as soon as possible.

High severity issues will probably bring problems and should be fixed.

Medium severity issues could potentially bring problems and should eventually be fixed.

Low severity issues are minor details and warnings that can remain unfixed but would be better fixed at some point in the future.

04. List of issues by severity

A. Critical

- ControllerHub.sol

Typo

- StrategyMdex.sol

Shadowed Variable

B. High

- HubPool.sol

Missing Validity Check

C. Medium

- N/A

D. Low

- ControllerHub.sol

Unused Interface

- StrategyMdex.sol

Redundant Code

Unused Interface

- General Issue

Duplicate Contract Names

05. List of issues by contract file

- HubPool.sol

Missing Validity Check: High

- ControllerHub.sol

Typo: Critial

Unused Interface: Low

- StrategyMdex.sol

Shadowed Variable: Critial

Redundant Code: Low

Unused Interface: Low

- General Issue

06. Issue descriptions and recommendations by contract file

- HubPool.sol

Missing Validity Check: High

Source and Description:

Lines 108, 115, 370 and 383: the function `setMin` in line 108, `setEarnLowerlimit` in 115, `earn` in line 370 and `available` in 383 have no validity check for the parameter `token` . If `token` is not an element of the array `TokenOfPid`, the variable `_pid` will be assigned 0, thus causing the following statements to read the first pool's state which is not expected behavior.

Recommendation:

Consider adding a validity check in each fo these functions for the parameter `token` before the statement that updates a pool's state. Here is a recommended change:

```
require(token != address(0) && address(pool.token) == token ,"invalid token");
```

Update: Fixed by the team adopting the recommended change.

- ControllerHub.sol

Typo: Critial

Source and Description:

Line 344: the statement `IStrategy _strategy = IStrategy(strategieList[0]);` should be `IStrategy _strategy = IStrategy(strategieList[i]);` ?

Recommendation:

Consider changing `IStrategy _strategy = IStrategy(strategieList[0]);` to

`IStrategy _strategy = IStrategy(strategieList[i]);`

Update: Fixed by the team adopting the recommended change.

Unused Interface: Low

Source and Description:

Line 130: the interface `IController` is defined but never used.

Recommendation:

Consider changing the statement `contract ControllerHub` in line 148 to

`contract ControllerHub is IController`

to explicitly force the `ControllerHub` contract to implement the interface `IController`.

Update: Fixed by the team removing the definition for the interface.

- StrategyMdex.sol

Shadowed Variable: Critial

Source and Description:

Line 451: the variable `liquidity` is redefined in line 451 which shadows the definition in line 445, thus causing unexpected errors.

Recommendation:

Considering removing the definition in line 451.

Update: Fixed by the team adopting the recommended change.

Redundant Code: Low

Source and Description:

Line 69: the implementation for the function modifier `onlyOwner` defined in line 69 is the same as the implementaion for the function modifer `onlyOwnerOrCtl` defined in line 74. Both of the two modifiers behave the same.

Recommendation:

Consider removing either of them.

Update: Fixed by the team adopting the recommended change.

Unused Interface: Low

Source and Description:

Line 45: the interface `Strategy` is defined but never used.

Recommendation:

Consider changing the statement `contract StrategyMdex` in line 55 to

`contract StrategyMdex is Strategy`

to explicitly force the `StrategyMdex` contract to implement the interface `Strategy`.

Update: Acknowledged by the team. The team doesn't think this will cause potential issues or risks and therefore prefers to keep it for now, and may make a change in a future upgrade.

- General Issue

Duplicate Contract Names：Low

Source and Description:

Some contracts have duplicate names such as `IController`, `IERC20`, `IHecoPool`, `SafeMath`, `SafeERC20` and `Address`. These duplicate names will trigger compiler warnings and unexpected issues or risks.

Recommendation:

Consider moving all contracts whose names are duplicate to separate contract files and importing these files when needed.

Update: Acknowledged by the team. The team doesn't think this will cause potential issues or risks and therefore prefers to keep it for now, and may make a change in a future upgrade.