Newland Audit Report

Version 1.0.0

Serial No. 2021080300022020

Presented by Fairyproof

August 3, 2021

01. Introduction

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

Audit Start Time:

July 29, 2021

Audit End Time:

July 30, 2021

Audited Code's Github Repository:

https://github.com/huobipool-community/newland-chef/tree/feature_booster/

Audited Code's Github Commit Number When Audit Started:

beb8b96113d852c86a1fa09fa5f79471d4dc4a60

Audited Code's Github Commit Number When Audit Ended:

6b16d11b13b168e7636f20f6748f70c3829e72b0

Audited Source Files:

The calculated SHA-256 values for the audited files when the audit was done are as follows:

```
BoosterStakingChef.sol:
0x0039a8e9690938fd4839fbf89a8cda1eadfb14c816b877c6aa58e0a6f90b4c7f

Treasury.sol:
0x991657e3ab365eab06aeaedd02f2fa41c81fcc6b969059e33237769da5cf5154

interface/IActionPools.sol:
0x3da385fd98df87bd2f72a9a0b26a5965718ffe14fb2343a0f615f9e920d8eb21

interface/ILavaChef.sol:
0x6abdd0ae6d2d7a2cc4d9d3b8cabb1588355fbda2d5435c5611f7b300c2ba5027

interface/ILavaFactory.sol:
0xcbdf5f0fc67ade3a1713f468d7c931a6ef0abfc84f2bae238691497bce684efd

interface/ILavaPair.sol:
0xd949cd39043b1d9fe770e1f1241a0c3cc711f931940c4a7c04e25708b61fa6fe

interface/IMdexChef.sol:
0x0b8fc9055470ae367cb2609a7b9e25ceb7a1403e17d8df3b2cc5c433e4e9e405
```

```
interface/IMdexFactory.sol:
0x6c7f8c969f8102a47adac8d38d50ae8dd35572a4cc12cd0247422269a24d22cc

interface/IMdexPair.sol:
0xf8ff935bffc75c91a6a3e33bf28d0ce9ea3ef1fe2830252d3cfb9fe3db477fb0

interface/IStrategyConfig.sol:
0xcfef691dcfb506c582961e1788fdaf6122cfc8f70bad8a863d44fd52e0be3336

interface/IStrategyLink.sol:
0x0debb702d11c1b7731dc728ca8e98cd806cc23e63309f3ca56da456202314944

interface/ISushiChef.sol:
0x8cdfb29b7d6df559cc6b0f3d9b1e57fa9944edda1c8863d5055afda855d98d4a

interface/ISushiFactory.sol:
0x6cb1d5862b8259b9a30932c75828ee5dde89feb212cbd5216a53c43d751ec244

interface/ISushiPair.sol:
0x0fc3b3ba1a0bb18ba6bd79231f44ad27ec51fd60eac3703fbb29c3e93698a844

interface/ITenBankHall.sol:
0x03403467479c37aba5abd75cc404d513600a65b428aff509c880989ecae4e513

interface/IWETH.sol:
0xed7fcba27321727dc41a6ae7d84157e5d687f1934dd3445fa352967ab8a37ba2

interface/IWHT.sol:
0xdf2cf22a530cfd2e550ef16cb6c1c49dab83d868262481672 27c91e8fe55836

library/MdxLib.sol:
0x9fba93dfc818ec6062b5f7a585af4f62f869a19e942c5d7bd6d4b4bac6763b02

library/TenMath.sol:
0x5e5871185220025e529a56d0889b5822e375ba4a80d0e708b1bc43a2299330cf

library/TransferHelper.sol:
0xabc92b1643747ea3fee75e169113092862a7fe28bdd4f1b9b902c68ea5ac5eaa
```

The source files audited include all the files with the extension "sol" as follows:

```
contracts/
├── BoosterStakingChef.sol
├── Treasury.sol
├── interface
|   ├── IActionPools.sol
|   ├── ILavaChef.sol
|   ├── ILavaFactory.sol
|   ├── ILavaPair.sol
|   ├── IMdexChef.sol
|   ├── IMdexFactory.sol
|   ├── IMdexPair.sol
|   ├── IStrategyConfig.sol
|   ├── IStrategyLink.sol
|   ├── ISushiChef.sol
|   ├── ISushiFactory.sol
|   ├── ISushiPair.sol
```

```
|   ├── ITenBankHall.sol
|   ├── IWETH.sol
|   └── IWHT.sol
└── library
    ├── MdxLib.sol
    ├── TenMath.sol
    └── TransferHelper.sol
```

The goal of this audit is to review Newland's solidity implementation for its aggregator application for Booster's leveraged mining, 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 system 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. If the audited source files are smart contract files, 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

The above files' code 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 source code.
   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 source code 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 the above source files. 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 Newland system should work:

https://newland.finance/

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

— Comments from Auditee

No vulnerabilities with critical, high, medium or low-severity were found in the above source code.

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 blockchain applications.

03. Introduction to Newland

Newland is a multi-chain based one-stand DeFi platform. It currently supports Ethereum, Huobi Ecological Chain (HECO), Binance Smart Chain (BSC), OKExChain (OKT). Newland offers lending, staking, cross-chain mining, multi-ecosystem(DOT, SOL) service etc.

Notes:

- **The application will invest users' assets to third-party platforms to earn profits. Those third-party platforms were not covered by this audit.**
- **The application may use some third-party libraries. These libraries were not covered by this audit.**

04. Major functions of audited code

The audited code implements aggregator functions which mainly include:

- token-pair mining: user-provided tokens will be paired to trading pairs and supplied to Mdex to get LP tokens
- The LP tokens will be supplied to Booster's TenBankHall contract for mining to earn the BOO token
- Users participate in mining to earn rewards in ERC-20 tokens

05. Key points in audit

During the audit we worked closely with the Newland team, helped fix some issues and refine the code implementation. Here is what we did:

- BoosterStakingChef.sol

Helped fix an issue in integer precision

The function `miningRewardPerBlock()` in line 157 had this issue

```
rewardPerBlock = rewardPerBlock.mul(pool.lpBalance).div(totalLPAmount);
uint baseVal = 1e18;
rewardPerBlock = rewardPerBlock.mul(baseVal.sub(miningProfitRate)).div(baseVal);
```

Recommendation:

Consider putting the division of `totalLPAmount` after all the multiplications.

Update: the Newland team submitted a fix with commit 6b16d11b13b168e7636f20f6748f70c3829e72b0.

Transferring of ERC-20 tokens didn't use SafeTransfer

The following functions didn't user `SafeTransfer`:

The `revoke()` function in line 181, the `updatePool()` function in line 372, the `remainTransfer()` function in line 572, the `safeHptTransfer()` function in line 585 and the `safeMiningTransfer()` function in line 599

Recommendation:

Consider using the `TransferHelper` or the interfaces in the `SafeERC20` library to handling token transfers. This would prevent issues with transferring of non-standard ERC-20 tokens.

Update: the Newland team submitted a fix with commit 6b16d11b13b168e7636f20f6748f70c3829e72b0.

06. Coverage of issues

The issues that the Fairyproof team covered when conducting the audit include but are not limited to the following ones:

- Re-entrancy Attack
- DDos Attack
- Integer Overflow
- Function Visibility
- Logic Vulnerability
- Uninitialized Storage Pointer
- Arithmetic Precision
- Tx.origin
- Shadow Variable

- Design Vulnerability
- Token Issurance
- Asset Security
- Access Control

07. 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.

08. List of issues by severity

A. Critical

- N/A

B. High

- N/A

C. Medium

- N/A

D. Low

- N/A

9. List of issues by source file

- N/A

10. Issue descriptions

- N/A

11. Recommendations to enhance the overall security

We list some recommendations in this section. They are not mandatory but will enhance the overall security of the system if they are adopted.

- N/A