Roco Audit Report

Version 1.0.0

Serial No.: 20220117000220019

Presented by Fairyproof

January 17, 2022

FAIRYPROOF

01. Introduction

This document includes the results of the audit performed by the Fairyproof team on the Roco project.

Audit Start Time:

Jan 14, 2022

Audit End Time:

Jan 17, 2022

Audited Source Files:

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

```
RocoToken.sol     :
0x862858cd27fe08bf0ded4ca24d23bde7be096ad17ebde6ac598540c8b5bb7a2d
RocoMultiStake.sol:
0xfd4d5a17ca72220e81468a5465b7cdf8ff5a3ea265350ad02cb3c7618643edeb
```

The goal of this audit is to review Roco's solidity implementation for its token issurance and staking functions, 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.

This audit only applies to the specified code, software or any materials supplied by the Roco team for specified versions. Whenever the code, software, materials, settings, enviroment etc is changed, the comments of this audit will no longer apply.

— 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.

— The Roco Team's Consent/Acknowledgement:

The audited materials of the project including but not limited to the documents, home site, source code, etc are all developed, deployed, managed, and maintained outside Mainland CHINA.

The members of the team, the foundation, and all the organizations that participate in the audited project are not Mainland Chinese residents.

The audited project doesn’t provide services or products for Mainland Chinese residents.

— 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 token issurance and staking functions should work:

https://roco.finance/

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

— Comments from Auditor

Serial Number	Auditor	Audit Time	Result
20220117000220 19	Fairyproof Security Team	January 14, 2022 - January 17, 2022	Low Risk

Summary:

The Fairyproof security team used its auto analysis tools and manual work to audit the project. During the audit, 3 risks of medium-severity and 2 risk of low-severity were discovered, and 1 neutral suggestion was raised. The Roco team confirmed 2 risks of low-severity, fixed 3 risks of medium-severity, and ignored 1 neutral suggestion.

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. Major functions of audited code

The audited code mainly implements the following functions:

- Issurance of ROCO

ROCO is an ERC-20 token

- Blockchain: Avalanche
- Token Address: 0xb2a85C5ECea99187A977aC34303b80AcbDdFa208
- Token Name: ROCO
- Token Symbol: ROCO
- Token Precision: 18
- Max Supply: 100,000,000
- No transaction charge: transaction charge is zero and cannot be modified
- No pyramid mechanism

- Staking

`RocoMultiStake.sol` is a staking contract which has the following functions:

- Single token staking
- Fees will be charged when users stake or withdraw tokens
- For a user who stakes for less than 1 year, the quantity of the reward token he/she gets cannot exceed two times of the quantity of his/her staking token. For a user who stakes for more than 1 year, the quantity of the reward token he/she gets cannot exceed three times of the quantity of his/her staking token.
- Pausing staking, pausing withdrawal of deposited tokens and pausing withdrawal of reward tokens

Note:

- Tokens that have fee charges in transactions and tokens that are scalable shouldn't be allowed for staking

- Admin Rights

There are two kinds of admins: admin in charge of token issurance and admin in charge of staking

Admin Rights for Token Issurance:

- Locking/Unlocking token distribution

Admin Rights for Staking:

- Pausing/Resuming token staking
- Withdrawing rewards
- Setting reward parameters
- Setting a staking operation's min/max quantity, and the max quantity of a liquidity pool
- Setting fees for staking and withdrawing staking fees

04. 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
- Replay Attack
- Reordering Attack
- Miner's Advantage
- Rollback Attack
- DDos Attack
- Transaction Ordering Attack
- Race Condition
- Access Control
- Integer Overflow/Underflow
- Timestamp Attack
- Gas Consumption
- Inappropriate Callback Function
- Function Visibility
- Implementation Vulnerability
- Uninitialized Storage Pointer
- Arithmetic Precision
- Tx.origin
- Fake Deposit
- Shadow Variable
- Design Vulnerability
- Token Issurance
- Admin Rights
- Inappropriate Proxy Design
- Inappropriate Use of Slots

- Asset Security
- Contract Upgrade/Migration
- Code Improvement
- Misc

05. 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.

Neutral is not an issue or risk but a suggestion for code improvement.

06. List of issues by severity

Index	Title	Issue/Risk	Severity	Status
FP-1	Inappropriate Handling of Admin Rights	Implementation Vulnerability	Low	Confirmed
FP-2	Missing Constraints for BalanceTemp	Admin Rights	Low	Confirmed
FP-3	Missing Constraints for Parameter Settings	Admin Rights	Medium	Fixed
FP-4	Improper Design of Emergency Withdrawal	Admin Rights	Medium	Fixed
FP-5	Missing Constraint for Parameter Setting	Implementation Vulnerability	Medium	Fixed
FP-6	Misleading Information in Require	Code Improvement	Neutral	Ignored

07. Issue descriptions

[FP-1] [Low] Inappropriate Handling of Admin Rights

Risk Severity: Low

Issue/Risk: Implementation Vulnerability

Description:

With regard to admin rights, in the `RocoToken.sol` and `RocoMultiStake.sol` files, the `lock` and `unlock` functions didn't handle the admin rights correctly..

Recommendation:

Consider removing the `lock` and `unlock` functions.

Status:

This bug in the `RocoMultiStake.sol` file was fixed but the one in the `RocoToken.sol` file cannot be fixed since the contract has been deployed.

[FP-2] [Low] Missing Constraints for `BalanceTemp`

Risk Severity: Low

Issue/Risk: Admin Rights

Description:

The `setBalanceTemp` function defined in the `RocoMultiStake.sol` file was used to set `BalanceTemp`. If the variable was improperly set, it would cause issues in staking, unstaking, calculation of APR etc.

Recommendation:

Consider calling the function to set `BalanceTemp` with great care.

Status:

It has been confirmed by the Roco team.

[FP-3] [Medium] Missing Constraints for Parameter Settings

Risk Severity: Medium

Issue/Risk: Admin Rights

Description:

The `startTime`, `endTime`, `FeeRate` defined in the `RocoMultiStake.sol` file all needed constraints. `startTime` should be less than `endTime` and `FeeRate` should be in 0~10.

Recommendation:

Consider adding constraints for these variables.

Status:

It has been fixed by the Roco team.

[FP-4] [Medium] Improper Design of Emergency Withdrawal

Risk Severity: Medium

Issue/Risk: Admin Rights

Description:

The `WithdrawEmergencyUser` function defined in the `RocoMultiStake.sol` file was used to withdraw staked tokens by users in emergency. This function had a variable to enable/disable this withdrawal operation. If the variable wasn't correctly set in emergent conditions, users' staked tokens would never be withdrawn.

Recommendation:

Consider removing the variable.

Status:

It has been fixed by the Roco team.

[FP-5] [Medium] Missing Constraint for Parameter Setting

Risk Severity: Medium

Issue/Risk: Implementation Vulnerability

Description:

The `setPerRocoSecond` function defined in `RocoMultiStake.sol` was used to set the value of `RocoPerSecond`. If a user hadn't claimed his/her reward and `RocoPerSecond` was set to a lower value, the number of reward tokens would be smaller than what the user should be able to claim.

Recommendation:

Consider adding a require to ensure every time when `setPerRocoSecond` is called the new value of `RocoPerSecond` is greater than the old value.

Status:

It has been fixed by the Roco team.

[FP-6] [Neutral] Misleading Information in Require

Risk Severity: Neutral

Issue/Risk: Code Improvement

Description:

The information defined in the `require` statement in line 410 of the `RocoToken.sol` file was misleading.

Recommendation:

Consider changing the information to "Account is already included".

Status:

It has been confirmed by the Roco team.

08. 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.

- Improvement on GAS Consumption

The variable `_tTotal` stands for the max supply and it is a constant. Using `constant` to define a variable saves gas consumption.

Recommendation:

Consider using `constant` to define `_tTotal`.

Status:

It has been confirmed by the Roco team.

- Adding Events

When a liquidity pool reaches its max capacity or its status has any change, no events are emitted to announce this change

Recommendation:

Consider adding events for status change.

Status:

It has been confirmed by the Roco team.

- Using Multi-sig Wallet to Set Reward Parameters

The reward parameters in the `RocoMultiStake.sol` contract were set by `owner`. This right is centralized.

Recommendation:

Consider using a multi-sig wallet to set reward parameters.

Status:

It has been confirmed by the Roco team.

- Don't Stake Or Reward Tokens That Have Callback Functions

This is to prevent re-entrancy attacks.