LuckTogether Audit Report

Version 1.0.0

Serial No. 2021032400022018

Presented by Fairyproof

March 24, 2021

FAIRYPROOF

01. Introduction

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

The audited code can be found in the public LuckTogether Github repository, and the version used for this report is commit

88de77de959c3c23a8af15aa7fbbd60633c88354

The audited contract files' onchain address: N/A

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

```
contracts/
├── prize-pool
|    ├── compound
|         └── CompoundPrizePool.sol
|    ├── PrizePool.sol
|    ├── PrizePoolInterface.sol
|    └── YieldSource.sol
├── registry
|    ├── Registry.sol
|    └── RegistryInterface.sol
├── reserve
|    ├── Reserve.sol
|    └── ReserveInterface.sol
├── utils
|    └── MappedSinglyLinkedList.sol
└── token
     ├── ControlledToken.sol
     └── ControlledTokenInterface.sol
```

Attention: other contract files not listed here but contained in these directories are not covered by this audit. Contract files and libraries from third parties, which the audited smart contracts interact with, are not covered by this audit either.

The goal of this audit is to review LuckTogether's solidity implementation for a lossless lottery, 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

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 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 the above contract 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 LuckTogether system should work:

https://www.lucktogether.com/

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

— Comments from Auditor

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

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. Introduction to LuckTogether

LuckTogether is a decentralized lossless lottery.

04. Major functions of audited code

The audited code implements the following functions:

It allows users to deposit ERC-20 tokens into the PrizePool. These deposited tokens are lended to third-party decentralized lending applications to earn interests which will be used to reward lottery winners.

The PrizePool performs liquidation when necessary. The owner has the full access control to the deposited assets and can withdraw all the underlying tokens. Users can get back their staked tokens and winners can get their winning shares.

How the reward is distributed to winners is implemented by the `TokenListenerInteface.sol` contract.

Attention: the contracts related to how the reward is distributed are not covered by this audit.

05. Key points in audit

During the audit, we worked closely with the LuckTogether team and helped the team：

- fix some bugs in its application logic,
- review the owner's access control,
- review contract migration/upgrade risks and
- refine some code writing

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. Major areas that need attention

Based on the provided contract files the Fairyproof team focused on the possible issues and risks related to the following functions or areas.

- Liquidation

The function `liquidation` defined in the `PrizePool.sol` contract file takes an input parameter `flatAddress` . If this parameter is incorrectly set, the amount of rewarded tokens sent to a user will be greater than what he/she should get. We recommended that the parameter `flatAddress` should be set only once on initialization or be set with extreme caution. The LuckTogether team fixed this issue.

In addition the `PrizePool.sol` contract file has a redundant variable `exitFee` which can be deleted. The LuckTogether team deleted it.

- Contract Migration/Upgrade

The initial `PrizePool.sol` and `ControllerToken.sol` contract files could be migrated or upgraded and this may cause potential risks. We recommended the team to remove this feature and the team removed it.

- Miscellaneous

We didn't find issues or risks in other functions or areas at the time of writing.

09. List of issues by severity

A. Critical

- N/A

B. High

- N/A

C. Medium

- N/A

D. Low

- N/A

10. List of issues by contract file

- N/A

11. Issue descriptions and recommendations by contract file

- N/A

12. Recommendations to enhance the overall security

- N/A