OptionRoom Audit Report

Version 1.0.0

Serial No. 2021061300022019

Presented by Fairyproof

June 13, 2021

灵踪安全

FAIRYPROOF

01. Introduction

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

Project Token's Name:

ROOM

Project Token's Ethereum Onchain Address:

https://etherscan.io/address/0xad4f86a25bbc20ffb751f2fac312a0b4d8f88c64

Audited Code's Github Repository:

N/A

Audited Code's Github Commit Number:

N/A

Audited Contract Files' Ethereum Onchain Address:

ROOM.sol: https://etherscan.io/address/0xad4f86a25bbc20ffb751f2fac312a0b4d8f88c64

Audited Contract Files:

The calculated SHA-256 values for the initial files are as follows:

```
COURT.sol:
0x50f4d7a100be5e1d9feaf1ec1bb8f08011e28a85aabeb89478cd7242f2f1a7f9

CourtStakeFlat.d.sol:
0x9aeea1530a06264c5c436ef9e3004393bc14adedccf6e8a23673d01be54290fb

ORConditionalTokens.f.sol:
0xe4c754b9152feab614097354564dcd2f3bfc1ceb7bccc24c6f3acd225d914e1a

ORFPMarket.f.sol:
0xac7a2985d98560f3284bb71d820436f6ed264b412d2a6be32873290ee7bbae0b

ORGovernor.f.sol:
0xc94a26cbcfcecb9abcd7391d7bf7b0770c488f651c5a877c90910ed5e3eb6e25

ORMarketController.f.sol:
0xbd285a304305302e0d965cd73534e4f712dcd8ead7fad1028d68f6c13622d112

ORMarketsQuery.f.sol:
0xd7d8984c4e3d90617b384c4cffe49d4680b81c2d47e070e98a8a06c4daca6647

ROOM.sol:
0x7e76aa5b794922ebe5f1d9b3eef0480dad7ee5cbbe2d80fafa95fe60877b92c3
```

```
RewardCenter.f.sol:
0x40da6e8dd62d63206e7b0bdadaf1b679f710b3de2c42e774533599f65397efb9

RewardProgram.f.sol:
0x4b1c95516c7e226b035e6264c05e23eb787b6562463959cee8e4e2df1f716205

RoomOraclePrice.f.sol:
0x32928038c4c78fb9ed9795be1d7f4743f62068948bc2cabce03b2f8f1b358280
```

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

```
/
├── COURT.sol
├── CourtStakeFlat.d.sol
├── ORConditionalTokens.f.sol
├── ORFPMarket.f.sol
├── ORGovernor.f.sol
├── ORMarketController.f.sol
├── ORMarketsQuery.f.sol
├── ROOM.sol
├── RewardCenter.f.sol
├── RewardProgram.f.sol
└── RoomOraclePrice.f.sol
```

The goal of this audit is to review OptionRoom’s solidity implementation for its oracle and forecast protocol, 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 OptionRoom system should work:

https://www.optionroom.finance/

Whitepaper

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

— Comments from Auditee

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

One vulnerability with medium-severity was 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 OptionRoom

OptionRoom is a user governed oracle and forecast protocol. OptionRoom has the ability to serve as an OaaS — Oracle as a Service where oracle requests are solved by governance. Oracle requests cost a fee and a solution incentive is paid in ROOM, rewarded to request solvers. OptionRoom allows users to create and participate in event derivatives that are pegged to real world outcomes by governance consensus.

04. Major functions of audited code

The audited code implements the following functions:

- Governance Token: COURT

    - Max supply: 40001
    - Token holders stake the COURT token to participate in governance
    - Token holders stake the COURT token to participate in gov voting in Forecast Market activities to earn ROOM
- Utility Token: ROOM
    - Max supply: 100 million
    - Used to pay Forecast Market fees and transaction fees
- Staking
    - Users can stake specified ERC-20 tokens to earn COURT
- Participation in Forecast Activities
    - Anyone can create a Forecast Market, set the market's parameters, boundaries and rewards in ROOM
    - Validate Forecast Market's: COURT holders can participate in validating a Forecast Market's outcomes
    - Participate in Voting in Forecast Market's Outcomes: users can stake specified ERC-20 tokens to participate in voting on Forecast Market's outcomes. Winners will get rewards in Room and losers lose nothing though.
    - When a Forecast Market expires, anyone can submit a dispute questioning the outcome by staking COURT. If the submitted dispute is eventually verified as being invalid, voters who voted for the outcome will get rewards in ROOM.
    - Jurisdiction: all COURT holders can vote on the dispute. The final result should win at least two thirds of the votes. If the dispute wins two thirds of the votes, the address that submitted the dispute will win 50% of the total staked COURTs in this Forecast Market, otherwise the address who submitted the dispute will lose his/her staked COURTs.

05. Key points in audit

During the audit, the audit team worked closely with the OptionRoom team, helped:

- fix a bug in price feed,
- fix a bug in the initialize function and
- present some enhancement recommendations for updating token price

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.

- Token Issurance

We checked whether or not the contract files could mint tokens at will.

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

- Asset Security in Staking and Forecast

We checked whether or not there were potential issues in asset security.

We found one issue. For more details please refer to "11 Issue descriptions".

- Forecast Implementation

We checked whether or not there were potential issues in the forecast implementation.

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

- 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

- CourtStakeFlat.d.sol

Inappropriate Access Control

D. Low

- N/A

10. List of issues by contract file

- CourtStakeFlat.d.sol

Inappropriate Access Control: Medium

11. Issue descriptions

- Inappropriate Access Control: Medium

Source and Description:

The `CourtStakeFlat._suspendAccount(address account, uint256 numOfDays)` function in line 148 in the `CourtStakeFlat.d.sol` contract file can be called by Admin to set `numOfDays`. If `numOfDays` is mistakenly set, users' assets may be locked permanently.

Recommendation:

Consider adding an upper bound for `numOfDays` to prevent it from being inappropriately set.

Update: Acknowledged by the OptionRoom team. The contract has been deployed. The OptionRoom team ensures that when they call `_suspendAccount` they will double-check and make sure `numOfDays` will be set properly.

12. 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.

- Ensuring price value is set properly prior to manually updating oracle's price

The `RoomOraclePrice.updateRoomPrice()` function in line 348 in the `RoomOraclePrice.f.sol` contract file can be used to set the Room price. If the price is not properly set it may cause loss of assets.

Recommendation:

Consider double-checking whether or not the price is set properly.

Update: the OptionRoom team will double check and make sure the price value is verified and set properly.