Booster Audit Report

Version 1.0.2

Serial No. 20210315000220018

Presented by Fairyproof

March 15, 2021

FAIRYPROOF

01. Introduction

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

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

52c2a63ee4ec2b499217f022b4bca3d1d24a735f

Audited contract files' onchain address: N/A (not deployed yet at the time of writing)

The contract files audited include all the files with the extension "sol" under both the `contracts` directory and the `interfaces` directory except the `contracts/pools/ActionPool.sol` file. They are as follows:

```
contracts/
├── BOOToken.sol
└── pools
    └── BOOPools.sol

interfaces/
├── IActionPools.sol
└── IActionTrigger.sol
```

The goal of this audit is to review booster's solidity implementation for its token issurance and liquidity mining 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.

— 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 booster system should work:

https://booster.farm/

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

— Comments from Auditor

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

One vulnerability with medium severity was acknowledged by the team, and the team will make changes in
future upgrades.

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 Booster

Booster is a cross-chain DeFi aggregator.

04. Major functions of audited code

The audited contract files implement the project's token issurance and liquidity mining functions. The total supply of the tokens is fixed and the tokens are distributed through liquidity minging.
When a user provides liquidity he/she will be rewarded with the project's token and an additional amount will be minted and distributed to the booster team. Users need to manually claim their staked tokens and rewards.
Attention: the audited contracts have a blacklist and can suspend claim of rewards. This is explained by the booster team as preventing malicious mining. Addresses in the blacklist can claim only part of their rewarded tokens. Editting of the blacklist and setting of the ratio of what percentage of rewarded tokens can be claimed are managed by the booster team.

Note: when the Fairyproof team conducted the audit, some contract files were still under development and were not covered by this audit, therefore the variable `extendPool` in line 65 in the `contracts\pools\BOOPool.sol` contract file must be set with a zero address. The booster team should in no way set this variable with a non-zero address before all the contract files are done and a new audit is conducted, otherwise some functions of the contract files may not work properly.

05. Key points in audit

During the audit we worked closely with the booster team and focused on checking the access control to the token issurance function, helped the team fix a bug in calculating mining rewards and presented an enhancement proposal.

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

- Replay Attack
- Reordering Attack
- Injection Attack
- DDos Attack
- Transaction Order Dependence
- Race Condition
- Integer Overflow/Underflow
- Gas Usage and Gas Limit
- Callback Function
- Timestamp Attack
- Function Visibility
- Logic Vulnerability
- Uninitialized Storage Pointer
- Arithmetic Precision
- Tx.origin
- Fake Deposit
- Shadow Variable
- Design Vulnerability
- Token Issurance
- Asset Security
- Access Control
- Proxy Contract
- Delegatecall
- Contract Migration/Upgrade

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

The implemented token issurance function gradually issues its token. The total supply of the tokens is fixed and defined by the contracts' deployer. However the owner can edit the minter's address, thus allowing any address to mint or destroy the tokens. But no matter what addresses are allowed to mint the tokens the total supply of the tokens is always fixed.

We found a vulnerability with medium severity. For more details please refer to section 11 "Issue descriptions and recommendations by contract file".

- 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

- BOOToken.sol

Inappropriate Access Control

D. Low

- N/A

10. List of issues by contract file

- BOOToken.sol

Inappropriate Access Control: Medium

11. Issue descriptions and recommendations by contract file

- BOOToken.sol

Inappropriate Access Control: Medium

Source and Description:

The owner can edit the minter's address, thus allowing any address to mint or destroy the tokens. But no matter what addresses are allowed to mint the tokens the total supply of the tokens is always fixed.

Recommendation:

Consider transferring the owner's access control to a multi-sig wallet or a DAO to mitigate the risk of minting tokens at will.

Update: Acknowledged by the booster team. The team will use the timelock to manage the owner's access control in future upgrades.

12. Recommendations to enhance the overall security

We list some recommendations in this section. They can enhance the overall security of the system if they are adopted.

- Changing Variable Visibility

The `BOOToken.sol` contract file has three variables `mintWhitelist`, `depositBlacklist` and `rewardRestricted`. Their visibility is set to the default value and therefore they cannot be accessed externally. Consider changing their visibility to `public`. The booster team made the change accordingly.

- Transferring Access Control

Consider transferring the access control to the liquidity pools to a multi-sig wallet or a DAO.