BitGuru Audit Report

Version 1.0.1

Serial No. 2021040900062026

Presented by Fairyproof

April 11, 2021

FAIRYPROOF

01. Introduction

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

Audited Code's Github Repository:

N/A

Audited Code's Github Commit Number:

N/A

Audited Contract Files' HECO Onchain Address:

- Guru
  0xF288A18b65FEe193C915158908143AbDde4D5106
- Farm
  0xa79F6aD7aDE25e1C4434Fd832430B93D2022888c
- Strat_Guru
  0xCCb7002571E3C0685F5A0086a757c18ccbB9619d

Audited Contract Files' BSC Onchain Address:

- Guru
  0xF1932eC9784B695520258F968b9575724af6eFa8
- BitGuruFarm
  0xAaD15F24f24c6a67b159e5ddb50376ce348Fb668
- Strat_Guru
  0x07cf83778024d46e20BA4161E545cc1b465A068C

Audited Contract Files:

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

```
Address.sol        :
0x11ad5e3e21434e00c4ceba1f5a977b7a68bdd7d16b849276ce4ff4495129eec7
BitGuruFarm.sol    :
0xa4c8ef1b0c9f654ae3156fde683b9e7086f542b40e8d8e99c2f1cbaa380f9ff4
Context.sol        :
0x9a3d1e5be0f0ace13e2d9aa1d0a1c3a6574983983ad5de94fc412f878bf7fe89
ERC20.sol          :
0xcc7ebd73b0865d4f5244a4015db133d57755f3327d506e336743be9c3c5d86c8
EnumerableSet.sol  :
0x1c80db0a768faf951fc3d048078f808575cc400f5b0eea5defa657257dffe66d
```

```
GuruTOKEN.sol      :
0xc0e7b05da88d7c552352446a4a639525a1a8436949777e72c65b7a1b644a2134
IERC20.sol         :
0x0573c2961569aa4906845d0cd428b5b7394956170054ceeaa8f8af96cd44875c
IMdexRouter.sol    :
0xa36af29d07f57e7f03c4a5f5952e208ac520b7f045b2253feed7b87e0fd1f4f1
ISwapMining.sol    :
0x292dc2cf0adf3cc28d0c6b72922f47310fb3d556ffe67ec1e5af0f0c047e8210
Ownable.sol        :
0xe310a50e815536ad3da96067bc32cb38382a9431a0cae4d7a887e79b5d954dc3
Pausable.sol       :
0x3efb901e506f8bb18ef283dc3721754853e57f79f9ca2056d45358716fa36643
ReentrancyGuard.sol:
0x3fc7968f4a1937caf3c96dffbac350398f86faad96288502e02c3a2b9f245e39
SafeERC20.sol      :
0x5a450b4e6ca591c82db0b530e63ff75b7c679f952329f1027510ee6a0624a02a
SafeMath.sol       :
0x38e47d1b5299ce0d5e48db837ed9248449043c50d90ffa0ee2ceb58ffde942c2
Strat_Guru.sol     :
0x79f4f0adff846adf8a5ab7a47a3344a5adddf140b767da92b55a958d63406bcd
```

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

```
HecoV2
├── Address.sol
├── BitGuruFarm.sol
├── Context.sol
├── ERC20.sol
├── EnumerableSet.sol
├── GuruTOKEN.sol
├── IERC20.sol
├── IMdexRouter.sol
├── ISwapMining.sol
├── Ownable.sol
├── Pausable.sol
├── ReentrancyGuard.sol
├── SafeERC20.sol
├── SafeMath.sol
└── Strat_Guru.sol
```

The goal of this audit is to review BitGuru's solidity implementation for its token issurance, liquidity mining and aggregator 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 BitGuru system should work:

https://www.bitguru.vip/

https://bitguru.gitbook.io/bitguru/

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

— Comments from Auditor

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

Three vulnerabilities with high-severity and one vulnerability with 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 BitGuru

BitGuru is an aggregator service for DeFi investors.

04. Major functions of audited code

The audited code implements the following functions:

- token issurance: there is no pre-mining, the max supply is 80000 and the access control to the mint function should be transferred to the `Farm` contract.
- aggregator service: users' staked assets will be used in third party applications to maximize their profits.
- liquidity mining: users can stake specific crypto assets to get rewards in another crypto asset.

Attention: users' staked assets will be used in third party applications to maximize their profits. If the third party applications are exploited, users' staked assets will be exposed to risks.

05. Key points in audit

During the audit, we worked closely with the BitGuru team:

- helped remove the inappropriate access control in the `BitGuruFarm.sol` and `Strat_Guru.sol` contract files, and
- gave some suggestions to refine its code implementation.

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 its token issurance function. There was no pre-mining and the max supply was 80000.

We didn't find issues or risks in this area at the time of writing.

- Asset Security

We checked whether or not users' staked assets were secure in its liquidity mining pools.

The staked assets' security is closely related to third-party applications where the staked assets are invested.

- Liquidity Mining

We checked whether or not there was inappropriate access control in the liquidity mining function such that the staked assets could be unexpectedly withdrawn.

We didn't find issues or risks in this area at the time of writing.

- Calculation of Rewards

We checked whether or not there were mistakes in the calculation of liquidity mining rewards.

We found some issues in this area. Please refer to section 11 for more details.

- 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

- BitGuruFarm.sol

Incorrect Algorithm for Calculation of Rewards

10. List of issues by contract file

- BitGuruFarm.sol

Incorrect Algorithm for Calculation of Rewards: Low

11. Issue descriptions and recommendations by contract file

- BitGuruFarm.sol

Incorrect Algorithm for Calculation of Rewards: Low

Source and Description:

Execution of the function `updatepool` or `withdraw` causes issues in the calculation of the rewards for liquidity mining. Execution of `updatepool` generates two types of reward: the `Bonus` reward and the `Guru` reward. This is not expected. Execution of `withdraw` should be expected to withdraw the staked assets in the `Guru` pool, however it doesn't.

Recommendation:

Consider re-implementing these two functions.

Update: the BitGuru team replied that the weight for the `Guru` token reward was set to 0, therefore this wouldn't cause issues. As for the withdraw function, the team replied that the function `leaveStakingGuru` should be used to withdraw the staked `Guru` tokens.

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.

- Removing Redundant Parameters

In the `BitGuruFarm.sol` contract file, the function `setAccBonus()` has two redundant parameters `_pa1` and `_pa2`. Consider removing them.

In the `Strat_Guru.sol` contract file, the function `deposit()` in line 164 has a redundant parameter `_userAddress`, the function `withdraw()` in line 220 has a redundant parameter `_userAddress`, and the function `distributeBonus()` in line 397 has a redundant parameter `poolid`. Consider removing all of them.