StarlinkMinerV2 Audit Report

Version 1.0.0

Serial No. 2021022200012014

Presented by Fairyproof

Feburary 22, 2021

FAIRYPROOF

01. Introduction

This document includes the results of the audit performed by the Fairyproof team on the StarlinkMinerV2 project, at the request of the Starlink team.

The audited code includes four contract files SLNToken.sol which has been deployed at https://scan.hecochain.com/address/0x4e252342cf35Ff02c4CCA9bc655129f5b4a2f901#contracts, StarPools.sol which has been deployed at https://scan.hecochain.com/address/0x503E606b016bA52de770201a8cD5D70Ad67fb00A#contracts, WordFundAsset.sol and WordFund.sol. All the four files can be found in the public StarlinkMinerV2 Github repository, and the version used for this report is commit

53e135bedc75d2bd0f6e429ba6308c8a33fef722

The goal of this audit is to review StarlinkMinerV2's solidity implementation for a decentralized search engine, 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

StarlinkMinerV2's codebase 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 four contract files SLNToken.sol, StarPools.sol, WordFundAsset.sol and WordFund.sol under the directory https://github.com/starlink-so/starlinkminerv2/tree/main/contracts. 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 StarlinkMinerV2 system should work:

https://pool.starlink.so/ and

whitepaper

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

— Comments from Auditor

No vulnerabilities with critical, high or low severities were found in the StarlinkMinerV2's codebase. Two vulnerabilities with medium severity were acknowledged by the team, and the team will manually and periodically perform necessary operations to avoid triggering potential issues or risks and may make changes in future upgrades.

The StarlinkMinerV2's codebase passed the audit performed by the Fairyproof team.

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. 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.

04. List of issues by severity

A. Critical

- N/A

B. High

- N/A

C. Medium

- StarPools.sol

Uncovered Conditions

Logic Vulnerability

D. Low

- N/A

05. List of issues by contract file

- StarPools.sol

Uncovered Conditions: Medium

06. Issue descriptions and recommendations by contract file

- StarPools.sol

Uncovered Conditions: Medium

Source and Description:

Line 154: the function `getBlocksReward` doesn't cover all necessary conditions.

Recommendation:

Since the contract has been deployed and may not be updated in the near term, consider manually and periodically calling the function `massUpdatePools` to avoid triggering uncovered conditions.

Update: Acknowledged by the depth team. The team prefers to keep it for now and will manually call the function `massUpdatePools` at least once per week to avoid triggering uncovered conditions..

Logic Vulnerability: Medium

Source and Description:

Line 215: the function `updatePool` doesn't handle the case in which the value of the variable `block.number` is far greater than `nextReductionBlock`, thus causing the gap between `block.number` and `nextReductionBlock` possibly containing multiple `reductionBlockPeriod`.

Recommendation:

Since the contract has been deployed and may not be updated in the near term, consider manually and periodically calling the function `massUpdatePools` to avoid triggering this issue.

Update: Acknowledged by the depth team. The team prefers to keep it for now and will manually call the function `massUpdatePools` at least once per week to avoid triggering this issue.