Kika Audit Report

Version 1.0.0

Serial No. 2021052800022024

Presented by Fairyproof

May 28, 2021

FAIRYPROOF

01. Introduction

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

Project Token's Name:

KIKA

Project Token's HECO Onchain Address:

https://hecoinfo.com/address/0x14fadc277f18f848b10d120ba723a3f0f4189a05

Audited Code's Github Repository:

N/A

Audited Code's Github Commit Number:

N/A

Audited Contract Files' HECO Onchain Address:

MiningContract:

https://hecoinfo.com/address/0x06B2291907cA84d09074378D518CaED92EeBc7a5

Comptroller:

https://hecoinfo.com/address/0x66e4a0f1738F418e809C0a4414A084e1E557D141

whitePaperInterestRateModel_2_15:

https://hecoinfo.com/address/0xE76584400E7D99A227d010e501807D7134b19DC3

whitePaperInterestRateModel_2_30:

https://hecoinfo.com/address/0x9d83163e288EDBAF8372A1225B30D5536AFf822D

whitePaperInterestRateModel_2_60:

https://hecoinfo.com/address/0x56cc606457c465AdF8A8b5e2ceb6f416bD7E7131

UniswapPriceOracleV2:

https://hecoinfo.com/address/0x5fc48Dc30D75D8f1c8d7ba2fF265D85eed9A57ef

KErc20:

https://hecoinfo.com/address/0x52A4234D7D4B06739F9C13B5D27163c7BE02af58

KikaToken:

https://hecoinfo.com/address/0x14fadc277f18f848b10d120ba723a3f0f4189a05

ERC20Template:

https://hecoinfo.com/address/0xef3cebd77e0c52cb6f60875d9306397b5caca375

Other related contract files' HECO onchain addresses: https://docs.kika.vc/info/contract-addresses

The goal of this audit is to review Kika's solidity implementation for its decentralized lending application, 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 Kika system should work:

https://kika.vc

project document

These were considered the specification, and when discrepancies arose with the actual code behavior, we consulted with the Kika 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 Kika

Kika is a decentralized lending application.

04. Major functions of audited code

The audited code implements the following functions:

- Issurance of the KIKA token, max supply : 1 billion
- Issurance of 15 KErc20 tokens
- Issurance of 15 ERC20Template tokens
- Staking and reward
- Lending and borrowing
- Liquidity mining

05. Key points in audit

During the audit Fairyproof worked closely with the Kika team and reviewed possible vulnerabilities in token issurance, decentralized lending and liquidity mining.

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.

- Integer Overflow/Underflow

We checked all the code sections, which have arithmetic operations and might introduce integer overflow or underflow if no safe libraries are used. All of them use safe libraries.

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

- Setting of Transaction Fees

We checked whether or not the transaction fees were set properly.

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

- Staking and Reward

We checked whether or not the reward for staking was calculated correctly and whether or not users could withdraw their rewards.

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

- Access Control

We checked each of the functions that can modify a state, especially those functions that can only be accessed by "owner".

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

- Token Issurance

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

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

- State Update

We checked some key state variables which should only be set at initialization.

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

- Asset Security

We checked whether or not all the functions that transfer assets are safely hanlded.

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

- Miscellaneous

The Fairyproof team 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

- N/A

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.

- N/A