inSure DeFi Audit Report

Version 1.0.0

Serial No. 2021102100012012

Presented by Fairyproof

October 21, 2021

FAIRYPROOF

01. Introduction

This document includes the results of the audit performed by the Fairyproof team on the inSure DeFi project.

Audit Start Time:

October 19, 2021

Audit End Time:

October 20, 2021

Token's Name:

inSure

Token's Symbol:

SURE

Token's Precisions:

18

Audited Code's Github Repository:

https://github.com/inSureToken/SmartContract

Audited Code's Github Commit Number When Audit Started:

474f85ce42afaaf31370476420cbcafbf78382fd

Audited Source File's Address:

https://etherscan.io/address/0xcb86c6a22cb56b6cf40cafedb06ba0df188a416e

Audited Source Files:

The calculated SHA-256 values for the audited files when the audit was done are as follows:

```
inSure.sol: 0xc8d1ed2ecb6cdc015b5e922e7ba31e7f1628f07c94be90d006ea71c82ac565df
inSureBSCAnySwap.sol:
0x087f90e15e5b22aa2f11b4f598ed68614f95b5480cf904be1169c220831c0b75
```

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

```
contracts/
├── inSure.sol
└── inSureBSCAnySwap.sol
```

The goal of this audit is to review inSure DeFi's token issurance function, 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.

This audit only applies to the specified code, software or any materials supplied by the inSure DeFi team for specified versions. Whenever the code, software, materials, settings, enviroment etc is changed, the comments of this audit will no longer apply.

— 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 system 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. If the audited source files are smart contract files, 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 source code.
   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 source code 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 source 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 token issurance should work:

https://insuretoken.net/

whitepaper

These were considered the specification.

— Comments from Auditee

No vulnerabilities with critical, high or medium-severity were found in the above source code.

One vulnerability with low-severity was found in the above source code.

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 blockchain applications.

03. Major functions of audited code

The audited code implements a token issurance function.

Name: inSure

Symbol: SURE

Precisions: 18

Max Supply: 88,000,000,000

04. 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

05. 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.

06. Major areas that need attention

Based on the provided souce code 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 had arithmetic operations and might introduce integer overflow or underflow if no safe libraries were used. All of them used safe libraries.

We found one issue. For more details please refer to "08. Issue descriptions".

- 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.

- Access Control

We checked each of the functions that could modify a state, especially those functions that could 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 could 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 were safely hanlded.

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

- Inefficient Code

We checked whether or not there was inefficient code which could impact the code's readability and maintainability.

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

- Contract Migration/Upgrade

We checked whether or not the contract files introduce issues or risks associated with contract migration/upgrade.

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.

07. List of issues by severity

A. Critical

- N/A

B. High

- N/A

C. Medium

- N/A

D. Low

- Integer Overflow

08. Issue descriptions

- Integer Overflow: Low

The following code section doesn't use safe math libraries to do arithmetic operations and may have integer overflow:

```
//line 83
allowance[_from][msg.sender] -= _value;

//line 130
balanceOf[msg.sender] -= _value;            // Subtract from the sender
totalSupply -= _value;                      // Updates totalSupply

//line 147
balanceOf[_from] -= _value;                         // Subtract from the
targeted balance
allowance[_from][msg.sender] -= _value;             // Subtract from the
sender's allowance
totalSupply -= _value;
```

Recommendation:

Consider using safe math libraries to do arithmetic operations.

09. 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