iSwap V4 Audit Report

Version 1.0.0

Serial No. 2021120800022020

Presented by Fairyproof

December 8, 2021

灵踪安全

FAIRYPROOF

01. Introduction

This document includes the results of the audit performed by the Fairyproof team on the iSwap V4 project, at the request of the iSwap team.

Audit Start Time:

December 1, 2021

Audit End Time:

December 6, 2021

Audited Source Files:

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

```
ISwapArbitrumBridge.sol:
0x65d8c793bc1f8df6d8e728528dc4bffdcd727db5ceb1d850b70ba82206db9979

ISwapAvaxBridge.sol:
0x6f429a15ae0355e01cd0af8fef77b26db91028200c28defeb66f23ab053c875b

ISwapBaseBridge.sol:
0x0cb3028d055333c283a76415704c877562d4281bffb63c2d4660f0129919c233

ISwapBscBridge.sol:
0xd1ab95fcea393278e2adc9983bdfdbf8e6755b38245308bb689ef71e89888db6

ISwapEthBridge.sol:
0x3893a53ed7db19e2c6eeedee028c4d2ecf9ae1104911074fff8bd35469b59d03

ISwapFtmBridge.sol:
0x42bc9b01630c77649b26806827995f33a985c310b78e3d35748ffaf3d16c6461

ISwapHecoBridge.sol:
0x929f3032659481ccb2c6c119ace00fbbcbe592269059c00ad73ba0e16974c9f2

ISwapOecBridge.sol:
0x91f4a19b1dd9410547b37d4282c701686da76a46b8c437066586d1de6cf89413

ISwapPolygonBridge.sol:
0x932af9b9a35dd9aa785483c46fc3421103dee97dba2041dba7643d0a098606b0

IHRC20.sol:
0x8f909a8cc5f325334aa4c767554ce7cf2d8faa7ce6cf37dc8461e959c1d0f8f0

IRouter.sol:
0x8d3d1f32a4632fe68614cc775e5c1fde5147b08abdc2647ff4df4ac07115170f
```

```
IWETH.sol:
0xf79a254f47708877e0ff8f94fcf6a9fd6668c58e89a0e89b8537af9b3905f09f

ParamsParser.sol:
0x12db2fc15f3ea649eb490f8e613e90f81e69bb2ff68494183d92cbdcc38d365c

RevertReasonParser.sol:
0xf4cdf713b9f1ae39a549d414e6e6c86b72a148da6d01df793ca02e8d820200e1

TransferHelper.sol:
0xb9774be2bc8df05e45e941137ad5aeff1485e8d27be0023aaa2894f52fc31a12

IHub.sol:
0x54217004bc0486a44702687489456a1aae34a2914e4bd5c29afe2ea3c878f978

IHRC20.sol:
0x8f909a8cc5f325334aa4c767554ce7cf2d8faa7ce6cf37dc8461e959c1d0f8f0

IWETH.sol:
0xd8c65eebf30834fd5f2e50e3f85e0ddec8f3f59f1016862d02b1820917547b41

readme.md:
0x0e4d51e0078d0384c792fdc322ab32c70ac9a18a8f2b3d57504443309532a54e

TransferHelper.sol:
0xb9774be2bc8df05e45e941137ad5aeff1485e8d27be0023aaa2894f52fc31a12
```

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

```
iswap-v4/
├── ISwapArbitrumBridge.sol
├── ISwapAvaxBridge.sol
├── ISwapBaseBridge.sol
├── ISwapBscBridge.sol
├── ISwapEthBridge.sol
├── ISwapFtmBridge.sol
├── ISwapHecoBridge.sol
├── ISwapOecBridge.sol
├── ISwapPolygonBridge.sol
├── interfaces
|   ├── IHRC20.sol
|   ├── IRouter.sol
|   └── IWETH.sol
└── utils
    ├── ParamsParser.sol
    ├── RevertReasonParser.sol
    └── TransferHelper.sol

iswap-IHub/
├── IHub.sol
├── interfaces
|   ├── IHRC20.sol
|   └── IWETH.sol
├── readme.md
└── utils
    └── TransferHelper.sol
```

The goal of this audit is to review iSwap V4's code, 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 iSwap 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 iSwap V4's cross-chain exchange application should work:

http://iswap.com

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

— 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. Introduction to iSwap V4

iSwap V4 is a cross-chain crypto exchange application.

Note: it currently only supports ETH, BSC, HECO, OKEX, Polygon, Arbitrum, Avalanche and Fantom.

04. Major functions of audited code

The audited code mainly implements the following function:

It takes stablecoins as a medium of exchange to implement cross-chain crypto exchange among ETH, BSC, HECO, OKEX, Polygon, Arbitrum, Avalanche and Fantom.

Note:

The validation of cross-chain transactions, which is done by the relayer nodes, was not covered by this audit.

The admin can deposit or withdraw onchain assets. Therefore, before enabling a cross-chain transaction that will withdraw onchain assets on the target chain, the admin needs to make sure the target chain has sufficient onchain assets for the user to withdraw.

Users who do cross-chain transactions in the system will pay transacton fees. Whitelisted users who do cross-chain transactions in the system may get rewards.

05. 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

06. 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.

07. 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 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.

- 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.

- 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 transferred assets were safely hanlded.

In line 114 of the `IHub.sol` file, `order.from` could be arbitrarily set by users. When the `allowance` of `order.from` had crypto assets, these assets could be maliciously transferred to other addresses. Here is the code section:

```
TransferHelper.safeTransferFrom(order.asset, order.from, address(this),
order.amount);
```

Consider changing `order.from` to `msg.sender` to make sure the sender can only send assets from his/her own wallet:

```
TransferHelper.safeTransferFrom(order.asset, msg.sender, address(this),
order.amount);
```

It has been fixed by the team.

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 introduced 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.

- Implementation of Cross-Chain Exchange

We checked whether or not there were issues or risks in the implementation of cross-chain exchange and the validation by the relayer nodes.

We found an issue, for more details please refer to "09. Issue descriptions".

- Miscellaneous

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

08. List of issues by severity

A. Critical

- N/A

B. High

- N/A

C. Medium

- N/A

D. Low

- Front-run Risk

09. Issue descriptions

- Front-run Risk: Low

Source and Description:

There was a front-run related vulnerability in the implementation of the validation of cross-chain transactions. Based on the implementation, when a user initiated a cross-chain transaction, an arbitrager could exploit the vulnerability to front-run thus causing the user to suffer a huge slippage.

Recommendation:

Consider using a random number, enforcing a max amount for a traction or alternative strategies to improve its validation of cross-chain transactions to mitigate front-running risks.

Update: the iSwap team will make changes in future upgrades.

10. 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.

- Transferring Admin's Access Control to Multi-sig Wallet

Consider transferring the admin's access control to bridges and relayers to multi-sig wallets to mitigate risks that may be caused by the Admin's private key being compromised .

- Monitoring Onchain Transactions

Consider monitoring onchain transactions in multiple nodes such that no transactions would be ignored or delayed.