iSwap V5 Audit Report

Version 1.0.0

Serial No.: 20220125000420 20

Presented by Fairyproof

January 25, 2022

FAIRYPROOF

01. Introduction

This document includes the results of the audit performed by the Fairyproof team on the iSwap V5 project.

Audit Start Time:

January 20, 2022

Audit End Time:

January 21, 2022

Audited Source Files:

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

```
ISwapArbitrumBridge.sol:
0x057056bffc36a69e98b273f1d2af9f3135d12d367a32edcf2e1c836848cc687f

ISwapAvaxBridge.sol:
0x483e5e2ababe105afafa756d8cb306ab976e6b7b2f3e68abf79ef19ab8e528c0

ISwapBaseBridge.sol:
0x2a8baf4f56e376962e6489cbf826aafb380ef564e88e1216082ff30887f2357b

ISwapBaseBridgeOptimization.sol:
0x66ed5f54f229f0935c419092eca756ec9d6e2d8b43cc993e5b7ef15b39e4c356

ISwapBaseStorage.sol:
0x80444eae611c88fe770f654d635b23f66cf71da8630862bd3aa6f69dfbec0562

ISwapBscBridge.sol:
0xfd9ad309b1dd277d20f390e62e74316a73e9af7827e4e9dcdeabad6e7c2ad6ec

ISwapEthBridge.sol:
0x07823469defeecbf5d1f68d74192ccde3f14d074ec5f89c29a9d1d8b7c5afa66

ISwapFtmBridge.sol:
0xda9946749bb67932990ead349eba65fed024086b84ee824d8aa90cd5c54bcb05

ISwapHecoBridge.sol:
0x0696dd77e54eb4d862b2ac52b42224d4afbcae124cbed54ca941a02438b78849

ISwapOecBridge.sol:
0xadad2c13d4ead6b99913f11e415a9e942dde9a168362b3a7970895fe9376b8f5

ISwapPolygonBridge.sol:
0x1a0c35db1d6e900602497cd9fa7eaef22bbdb14fc949a9577cff377d920c0c25

IHRC20.sol:
0x8f909a8cc5f325334aa4c767554ce7cf2d8faa7ce6cf37dc8461e959c1d0f8f0
```

```
IRouter.sol:
0x8d3d1f32a4632fe68614cc775e5c1fde5147b08abdc2647ff4df4ac07115170f

IWETH.sol:
0xf79a254f47708877e0ff8f94fcf6a9fd6668c58e89a0e89b8537af9b3905f09f

ParamsParser.sol:
0x12db2fc15f3ea649eb490f8e613e90f81e69bb2ff68494183d92cbdcc38d365c

RevertReasonParser.sol:
0xf4cdf713b9f1ae39a549d414e6e6c86b72a148da6d01df793ca02e8d820200e1

TransferHelper.sol:
0xb9774be2bc8df05e45e941137ad5aeff1485e8d27be0023aaa2894f52fc31a12
```

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

```
contracts/
├── ISwapArbitrumBridge.sol
├── ISwapAvaxBridge.sol
├── ISwapBaseBridge.sol
├── ISwapBaseBridgeOptimization.sol
├── ISwapBaseStorage.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
```

The goal of this audit is to review iSwap V5's solidity implementation for its cross-chain exchange 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 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.

— The iSwap Team's Consent/Acknowledgement:

The audited materials of the project including but not limited to the documents, home site, source code, etc are all developed, deployed, managed, and maintained outside Mainland CHINA.

The members of the team, the foundation, and all the organizations that participate in the audited project are not Mainland Chinese residents.

The audited project doesn’t provide services or products for Mainland Chinese residents.

— 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 cross-chain exchange 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 Auditor

Serial Number	Auditor	Audit Time	Result
2022012500042020	Fairyproof Security Team	January 20, 2022 - January 21, 2022	Low Risk

Summary:

The Fairyproof security team used its auto analysis tools and manual work to audit the project. During the audit, 1 risk of low-severity was identified. The iSwap team confirmed the risk.

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 mainly implements a cross-chain exchange where users can do crypto exchanges using stable coins as a medium of exchange across multiple blockchains including ETH, BSC, HECO, OKEX, Polygon, Arbitrum, Avalanche and Fantom.

Note:

Validation for cross-chain transactions is done by relayers whose implementation was not covered by this audit.

The iSwap team is able to deposit or withdraw onchain assets, therefore the iSwap team should make sure there are sufficient assets in target chains for users to complete cross-chain exchanges.

When a user initiates a cross-chain exchange, a transaction fee will be charged. Whitelisted users will get rewards for transactions.

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
- Replay Attack

- Reordering Attack
- Miner's Advantage
- Rollback Attack
- DDos Attack
- Transaction Ordering Attack
- Race Condition
- Access Control
- Integer Overflow/Underflow
- Timestamp Attack
- Gas Consumption
- Inappropriate Callback Function
- Function Visibility
- Implementation Vulnerability
- Uninitialized Storage Pointer
- Arithmetic Precision
- Tx.origin
- Fake Deposit
- Shadow Variable
- Design Vulnerability
- Token Issurance
- Admin Rights
- Inappropriate Proxy Design
- Inappropriate Use of Slots
- Asset Security
- Contract Upgrade/Migration
- Code Improvement
- Misc

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.

Informational is not an issue or risk but a suggestion for code improvement.

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 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.

- Variable Setting

We checked whether or not the variable settings were proper.

In our initial review we found the following issues:

The `ISwapBaseBridgeOptimization.sol` file missed zero-address checks for some variables.

Recommendation:

Consider adding a zero-address check for the `to` variable in lines 62, 94 and 127:

```
require(to != address(0), 'BaseBridge: INVALID_TO');
```

The `_getSwapInfoSrcToken` function defined in the `ISwapBaseBridgeOptimization.sol` file missed a check for `path.length`.

Recommendation:

Consider adding a check for `path.length` as follows:

```
function _getSwapInfoSrcToken(SwapInfo[] calldata swapInfo, uint256 index)
internal pure returns(address) {
    //adding the following require statement.
    require(swapInfo[index].itemPath[0].path.length > uint256(1), "BaseBridge:
ITEM_PATH_LESS_THEN_2");
    address srcToken = swapInfo[index].itemPath[0].path[0];
    return srcToken;
}
```

The `_getSwapInfoDstToken` function missed a check for `itemPathLen` and `pathLen` respectively.

Recommendation:

Consider adding a check for both `itemPathLen` and `pathLen` respectively as follows:

```
function _getSwapInfoDstToken(SwapInfo[] calldata swapInfo, uint256 index)
internal pure returns(address) {
    uint256 itemPathLen = swapInfo[index].itemPath.length;
    //adding the following require statement
    require(itemPathLen > uint256(0), "BaseBridge: ITEM_LESS_THEN_1");
    uint256 pathLen = swapInfo[index].itemPath[itemPathLen - 1].path.length;
    //adding the following require statement
    require(pathLen > uint256(1), "BaseBridge: ITEM_PATH_LESS_THEN_2");
    address dstToken = swapInfo[index].itemPath[itemPathLen - 1].path[pathLen -
1];
    return dstToken;
}
```

Status:

It has been fixed by the iSwap team. We didn't find issues or risks in these functions or areas at the time of writing after the team fixed it.

- 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.

- Cross-chain Exchange

We checked whether or not there were issues or risks in the cross-chain exchange's implementation.

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

- 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.

- Code Improvement

We checked whether or not the code could be improved to enchance its efficiency and readability.

In our initial review, we found the following issues:

Line 93 in the `ISwapBaseBridgeOptimization.sol` file could be deleted. Here was the code:

```
require(swapInfo[0].itemPath[0].path[0] == chainToken, 'BaseBridge:
INVALID_PATH');
```

And the following line could be inserted in line 99:

```
require(dstToken == chainToken, 'BaseBridge: INVALID_PATH');
```

Recommendation:

Consider making changes as suggested.

Status:

It has been fixed by the iSwap team. We didn't find issues or risks in these functions or areas at the time of writing after the team fixed it.

- Miscellaneous

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

07. List of issues by severity

Index	Title	Issue/Risk	Severity	Status
FP-1	Front-run Issue	Misc	Low	Confirmed

08. Issue descriptions

[FP-1] [Low] Front-run Issue

Risk Severity: Low

Issue/Risk: Misc

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

Status:

It has been confirmed by the iSwap team. And the team will use random numbers, enforce a max amount for a traction or alternative strategies to improve its validation of cross-chain transactions to mitigate front-running risks.

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.

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

Consider transferring 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.