

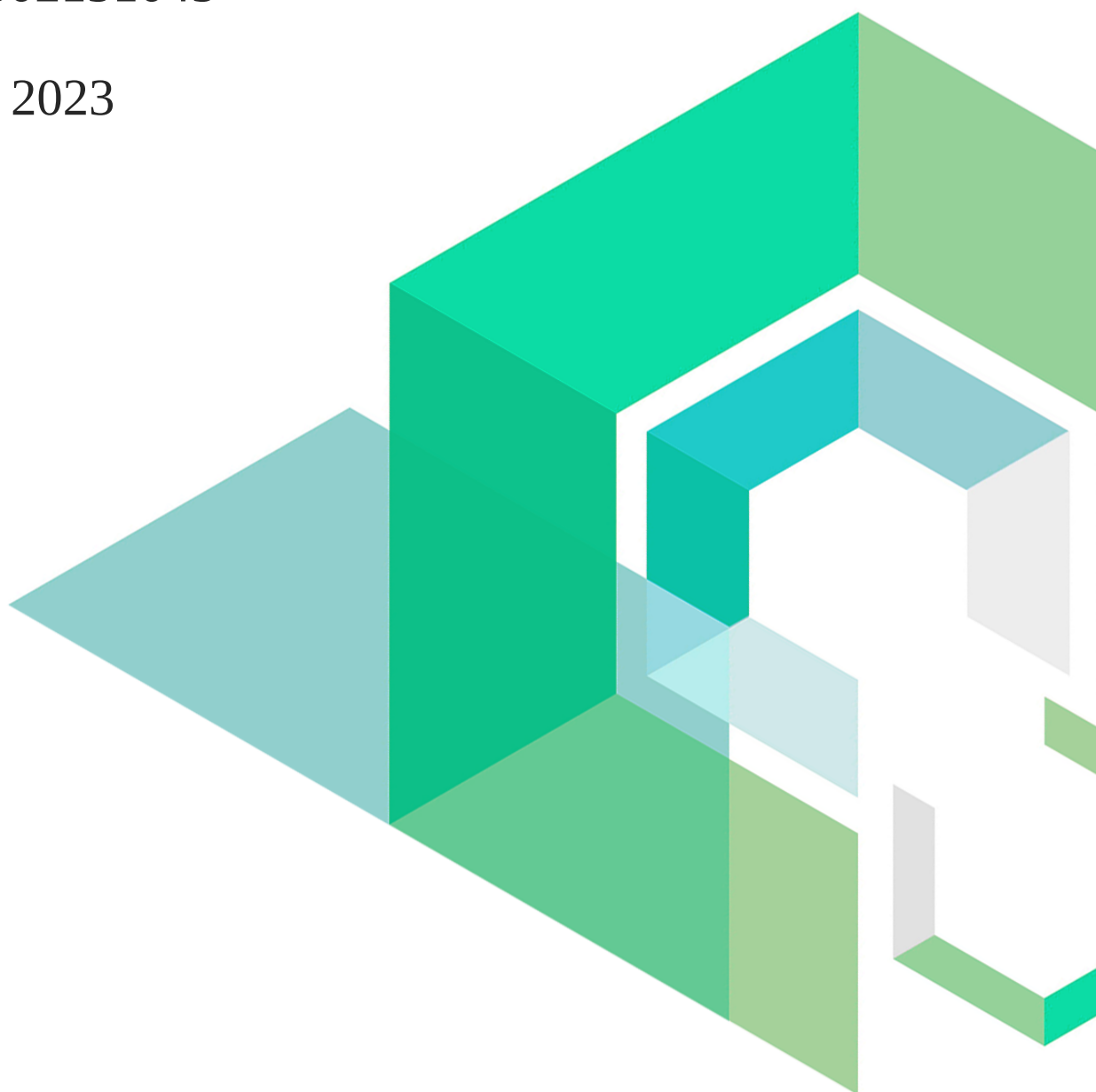
HIESWAP

Smart Contract Security Audit

V1.0

No. 202302151045

Feb 15th, 2023

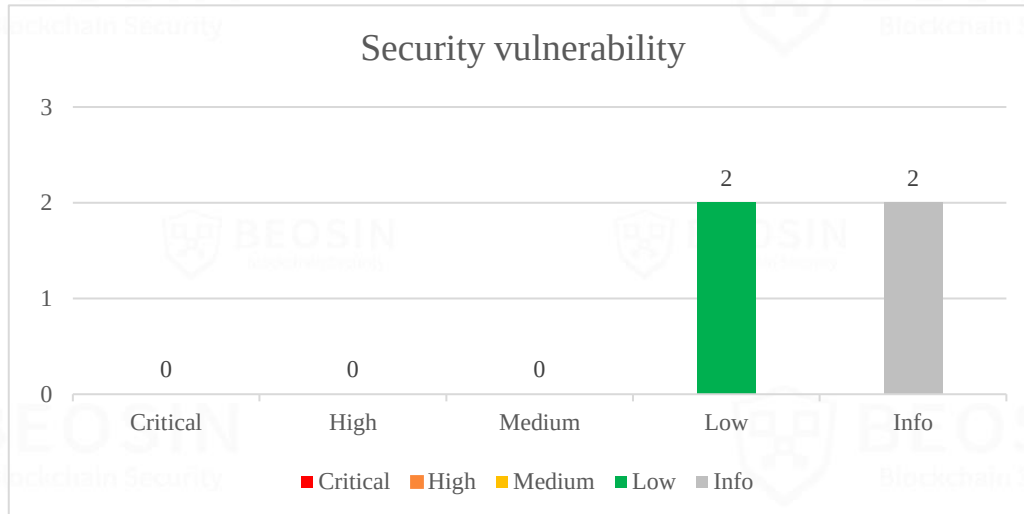


Contents

Summary of Audit Results.....	1
1 Overview.....	3
1.1 Project Overview	3
1.2 Audit Overview	3
2 Findings	4
[HIESWAP-1] Use <i>safeTransfer</i> consistently instead of <i>transfer</i>	5
[HIESWAP-2] The <i>setToken</i> should limit 0 address to add	6
[HIESWAP-3] Should check that the lengths of the incoming arrays are the same.....	7
[HIESWAP-4] The <i>transferOwner</i> lacks check for 0 address.....	8
3 Appendix	9
3.1 Vulnerability Assessment Metrics and Status in Smart Contracts	9
3.2 Audit Categories	11
3.3 Disclaimer.....	13
3.4 About Beosin	14

Summary of Audit Results

After auditing, 2 Low-risk and 2 Info items were identified in the HIESWAP project. Specific audit details will be presented in the **Findings** section. Users should pay attention to the following aspects when interacting with this project:



- **Project Description:**

1. **Business overview**

Hieswap is a cross-chain bridge contract based on the EVM chain. In this contract, the administrator can call the *setToken* function to add a single tokens (tokens related information includes Rate, Fee, maximum handling fee, minimum transfer number, maximum transfer number), or you can call the *setTokens* function for multiple generations for multiple generations add currency information. Users can conduct cross-chain transfer and will charge a certain amount of handling fees. In the Hieswap contract, administrators can also call *take* to transfer tokens in the contract and change the owner address. Similarly, administrators can also use *crossTokenWithNative* to transfer native currencies.

1 Overview

1.1 Project Overview

Project Name	HIESWAP
Platform	Polygon, Ethereum, Heco, OKExChain, BNB Chain, Fon Smart Chain, TRON
Audit Scope	Hieswap.sol
File Hash	5491B23CD8F95FFA3A4A14F804082127F4610E1496EBDBFFEDACA5D7F397D921

1.2 Audit Overview

Audit work duration: Feb 14, 2023 – Feb 15, 2023

Audit methods: Formal Verification, Static Analysis, Typical Case Testing and Manual Review.

Audit team: Beosin Security Team.

2 Findings

Index	Risk description	Severity level	Status
HIESWAP-1	<i>Use <code>safeTransfer</code> consistently instead of <code>transfer</code></i>	Low	Acknowledged
HIESWAP-2	<i>The <code>setToken</code> should limit 0 address to add</i>	Low	Acknowledged
HIESWAP-3	<i>Should check that the lengths of the incoming arrays are the same</i>	Info	Acknowledged
HIESWAP-4	<i>The <code>transferOwner</code> lacks check for 0 address</i>	Info	Fixed

Status Notes:

1. HIESWAP-1 is not fixed and will not cause any issues.
2. HIESWAP-2 is not fixed and will not cause any issues.
3. HIESWAP-3 is not fixed and will not cause any issues.

Finding Details:

[HIESWAP-1] Use *safeTransfer* consistently instead of *transfer*

Severity Level	Low
Type	Business Security
Lines	Hieswap.sol #L79
Description	Use <i>safeTransfer</i> consistently instead of <i>transfer</i>. <pre> 76 function take(address token, uint256 amount) external onlyOwner { 77 if(token == address(0)) 78 return payable(owner).transfer(amount); 79 IERC20(token).transfer(owner, amount); 80 } 81 </pre> Figure 1 Source code of <i>take</i> function(unfixed)
Recommendations	It is recommended to use <i>safeTransfer</i> to transfer tokens.
Status	Acknowledged.

[HIESWAP-2] The *setToken* should limit 0 address to add

Severity Level	Low
Type	Business Security
Lines	Hieswap.sol #L54-L59
Description	The <i>setToken</i> function does not limit whether to add address(0) as a token address. If address(0) is added successfully, the code for transferring token in <i>crossChain</i> will be invalid.

```
function setToken(
    address token,
    uint256 toChainId,
    uint256 rate,
    uint256 min,
    uint256 max,
    uint256 fee,
    uint256 maxFee
) public onlyOwner {
    supportTokens[token][toChainId] = rate;
    feeRates[token][toChainId] = fee;
    maxFees[token][toChainId] = maxFee;
    mins[token][toChainId] = min;
    maxes[token][toChainId] = max;
}
```

Figure 2 Source code of *setToken* function(unfixed)

Recommendations	It is recommended to add the judgment of whether the token is address(0) in <i>setToken</i> , and judge whether the token has been added to prevent overwriting.
Status	Acknowledged.

[HIESWAP-3] Should check that the lengths of the incoming arrays are the same

Severity Level	Info
Type	Business Security
Lines	Hieswap.sol #L60-73,L104-L112
Description	Both functions in <i>setTokens</i> and <i>multiOut</i> accept multiple array parameters. If the length of one array is inconsistent with other arrays, the execution will fail.

```
function setTokens(
    address[] memory token,
    uint256[] memory toChainId,
    uint256[] memory rate,
    uint256[] memory min,
    uint256[] memory max,
    uint256[] memory fee,
    uint256[] memory maxFee
) external onlyOwner {
    uint256 count = token.length;
    for(uint256 i = 0; i < count; i++){
        setToken(token[i], toChainId[i], rate[i], min[i], max[i], fee[i], maxFee[i]);
    }
}
```

Figure 3 Source code of *setTokens* functions(unfixed)

```
function multiOut(
    uint256[] memory fromChainId,
    bytes32[] memory txHashe,
    address[] memory token,
    address[] memory account,
    address[] memory to,
    uint256[] memory amount,
    uint256[] memory nativeAmount
) onlyOwner external {
    uint256 count = fromChainId.length;
    for(uint256 i = 0; i < count; i++){
        crossTokenWithNative(fromChainId[i], txHashe[i], token[i], account[i], to[i], amount[i], nativeAmount[i]);
    }
}
```

Figure 4 Source code of *multiOut* functions(unfixed)

Recommendations	It is recommended to check whether the length of the incoming array is consistent.
Status	Acknowledged.

[HIESWAP-4] The *transferOwner* lacks check for 0 address

Severity Level	Info
Type	Business Security
Lines	Hieswap.sol #L86-L88
Description	If newOwner is address(0), it is equivalent to giving up ownership, and all functions that require administrator operations will not be executed.

```

86     function transferOwner(address newOwner) external onlyOwner {
87         owner = newOwner;
88     }

```

Figure 5 Source code of *transferOwner* functions(unfixed)

Recommendations	It is recommended to judge whether the newOwner address is 0 address. And add an event for key parameter changes.
-----------------	---

Status	Fixed.
--------	--------

```

86     function transferOwner(address newOwner) external onlyOwner {
87         require(newOwner != address(0));
88         owner = newOwner;
89     }
90

```

Figure 6 Source code of *transferOwner* functions(fixed)

3 Appendix

3.1 Vulnerability Assessment Metrics and Status in Smart Contracts

3.1.1 Metrics

In order to objectively assess the severity level of vulnerabilities in blockchain systems, this report provides detailed assessment metrics for security vulnerabilities in smart contracts with reference to CVSS 3.1 (Common Vulnerability Scoring System Ver 3.1).

According to the severity level of vulnerability, the vulnerabilities are classified into four levels: "critical", "high", "medium" and "low". It mainly relies on the degree of impact and likelihood of exploitation of the vulnerability, supplemented by other comprehensive factors to determine of the severity level.

Impact Likelihood	Severe	High	Medium	Low
Probable	Critical	High	Medium	Low
Possible	High	High	Medium	Low
Unlikely	Medium	Medium	Low	Info
Rare	Low	Low	Info	Info

3.1.2 Degree of impact

- **Severe**

Severe impact generally refers to the vulnerability can have a serious impact on the confidentiality, integrity, availability of smart contracts or their economic model, which can cause substantial economic losses to the contract business system, large-scale data disruption, loss of authority management, failure of key functions, loss of credibility, or indirectly affect the operation of other smart contracts associated with it and cause substantial losses, as well as other severe and mostly irreversible harm.

- **High**

High impact generally refers to the vulnerability can have a relatively serious impact on the confidentiality, integrity, availability of the smart contract or its economic model, which can cause a greater economic loss, local functional unavailability, loss of credibility and other impact to the contract business system.

- **Medium**

Medium impact generally refers to the vulnerability can have a relatively minor impact on the confidentiality, integrity, availability of the smart contract or its economic model, which can cause a small amount of economic loss to the contract business system, individual business unavailability and other impact.

- **Low**

Low impact generally refers to the vulnerability can have a minor impact on the smart contract, which can pose certain security threat to the contract business system and needs to be improved.

3.1.4 Likelihood of Exploitation

- **Probable**

Probable likelihood generally means that the cost required to exploit the vulnerability is low, with no special exploitation threshold, and the vulnerability can be triggered consistently.

- **Possible**

Possible likelihood generally means that exploiting such vulnerability requires a certain cost, or there are certain conditions for exploitation, and the vulnerability is not easily and consistently triggered.

- **Unlikely**

Unlikely likelihood generally means that the vulnerability requires a high cost, or the exploitation conditions are very demanding and the vulnerability is highly difficult to trigger.

- **Rare**

Rare likelihood generally means that the vulnerability requires an extremely high cost or the conditions for exploitation are extremely difficult to achieve.

3.1.5 Fix Results Status

Status	Description
Fixed	The project party fully fixes a vulnerability.
Partially Fixed	The project party did not fully fix the issue, but only mitigated the issue.
Acknowledged	The project party confirms and chooses to ignore the issue.

3.2 Audit Categories

No.	Categories	Subitems
1	Coding Conventions	Redundant Code
		require/assert Usage
		Cycles Consumption
2	General Vulnerability	Integer Overflow/Underflow
		Reentrancy
		Pseudo-random Number Generator (PRNG)
		Transaction-Ordering Dependence
		DoS (Denial of Service)
		Function Call Permissions
		Returned Value Security
		Rollback Risk
		Replay Attack
		Overriding Variables
		Call Canister controllable
3	Business Security	Canister upgrade risk
		Third-party Protocol Interface Consistency
		Business Logics
		Business Implementations
		Manipulable Token Price
		Centralized Asset Control
		Asset Tradability
		Arbitrage Attack

Beosin classified the security issues of smart contracts into three categories: Coding Conventions, General Vulnerability, Business Security. Their specific definitions are as follows:

- **Coding Conventions**

Audit whether smart contracts follow recommended language security coding practices. For example, smart contracts developed in Solidity language should fix the compiler version and do not use deprecated keywords.

- **General Vulnerability**

General Vulnerability include some common vulnerabilities that may appear in smart contract projects. These vulnerabilities are mainly related to the characteristics of the smart contract itself, such as integer overflow/underflow and denial of service attacks.

- **Business Security**

Business security is mainly related to some issues related to the business realized by each project, and has a relatively strong pertinence. For example, whether the lock-up plan in the code match the white paper, or the flash loan attack caused by the incorrect setting of the price acquisition oracle.

*Note that the project may suffer stake losses due to the integrated third-party protocol. This is not something Beosin can control. Business security requires the participation of the project party. The project party and users need to stay vigilant at all times.

3.3 Disclaimer

The Audit Report issued by Beosin is related to the services agreed in the relevant service agreement. The Project Party or the Served Party (hereinafter referred to as the "Served Party") can only be used within the conditions and scope agreed in the service agreement. Other third parties shall not transmit, disclose, quote, rely on or tamper with the Audit Report issued for any purpose.

The Audit Report issued by Beosin is made solely for the code, and any description, expression or wording contained therein shall not be interpreted as affirmation or confirmation of the project, nor shall any warranty or guarantee be given as to the absolute flawlessness of the code analyzed, the code team, the business model or legal compliance.

The Audit Report issued by Beosin is only based on the code provided by the Served Party and the technology currently available to Beosin. However, due to the technical limitations of any organization, and in the event that the code provided by the Served Party is missing information, tampered with, deleted, hidden or subsequently altered, the audit report may still fail to fully enumerate all the risks.

The Audit Report issued by Beosin in no way provides investment advice on any project, nor should it be utilized as investment suggestions of any type. This report represents an extensive evaluation process designed to help our customers improve code quality while mitigating the high risks in blockchain.

3.4 About Beosin

Beosin is the first institution in the world specializing in the construction of blockchain security ecosystem. The core team members are all professors, postdocs, PhDs, and Internet elites from world-renowned academic institutions. Beosin has more than 20 years of research in formal verification technology, trusted computing, mobile security and kernel security, with overseas experience in studying and collaborating in project research at well-known universities. Through the security audit and defense deployment of more than 2,000 smart contracts, over 50 public blockchains and wallets, and nearly 100 exchanges worldwide, Beosin has accumulated rich experience in security attack and defense of the blockchain field, and has developed several security products specifically for blockchain.



Official Website

<https://www.beosin.com>

Telegram

<https://t.me/+dD8Bnqd133RmNWNl>

Twitter

https://twitter.com/Beosin_com

Email

Contact@beosin.com

