BEOSIN
Blockchain Security

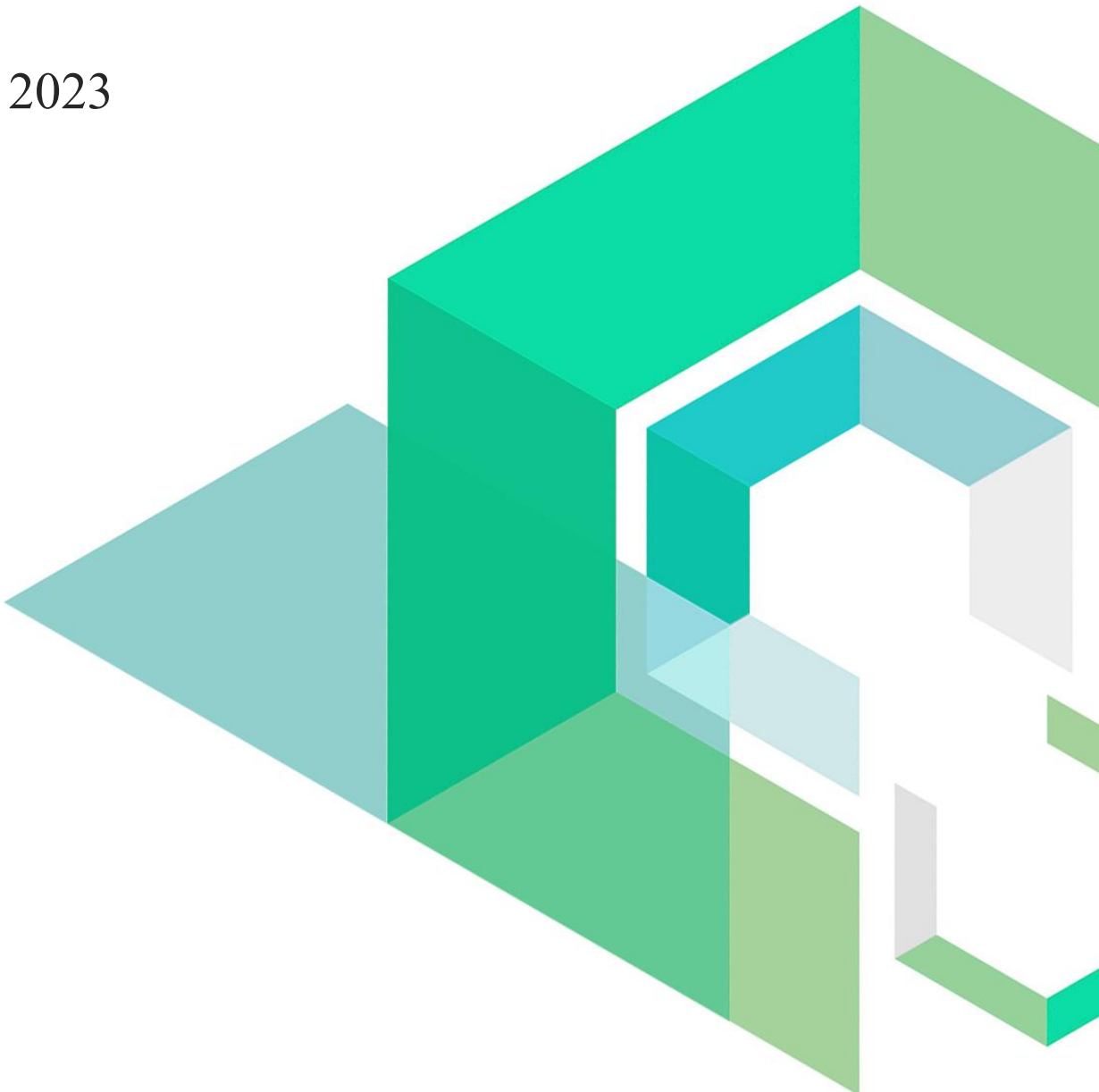
BtokSwap

Smart Contract Security Audit

V1.0

No. 202304071650

Apr 07th, 2023



Contents

Summary of Audit Results

After auditing, 3 Low-risks and 1 Info -risk were identified in the BtokSwap project. Specific audit details will be presented in the **Findings** section. Users should pay attention to the following aspects when interacting with this project:

*Notes:

- **Risk Description:**

  For the project side, it is recommended to use the multi-signature wallet to manage the permissions of the executer and owner address. For users, they should pay attention to checking whether the parameters such as the rate and address are correct when interacting to avoid losses.

Business overview

BtokSwap is a swap aggregator that allows users to combine multiple transaction orders into a single transaction. The contract provides three trading functions, one is to normally trade with other Uniswap-based DEX, the other is to support tokens that take fees for swapping, and the remaining one is to interact with the cross contract in the project, the cross function is not included in the scope of this audit. The contract has a pause function, and the user cannot swap when the contract is paused. When using the swap aggregation function, a certain handling fee will be charged according to the rate set by the project party. By default, it will be charged before swap, and users can also choose to be charged after swap.

1 Overview

1.1 Project Overview

Project Name	BtokSwap
Platform	Ethereum
File Hash(SHA256)	52ad3404213478571946399f6e0d0288a02e6fb37824ec4c350f7150b905084c (TransitAllowed.sol) c7f22930d47be576b4fe45e74f3a6b1670ff52eeb6408647830b14787c62f4cd (BtokSwap.sol) 3033089b6a084924b32215b97eb4736664513d0dd550b84627164a9a6ccb9b95 (BtokSwapFees.sol) b273464b6359f79ade4fae2c9814345ad09c6a6ec202be5e59b3869fa5b3f5b3 (BtokSwapRouterV4.sol)

1.2 Audit Overview

Audit work duration: Apr 03, 2023 – Apr 07, 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
BtokSwap-1	Fees can be set to any value	Low	Acknowledged
BtokSwap-2	Risk of centralization	Low	Acknowledged
BtokSwap-3	Function design flaw	Low	Acknowledged
BtokSwap-4	Token decimals may not be considered	Info	Acknowledged

Status Notes:

1. BtokSwap-1 is not fixed and may cause the user's assets to suffer losses if the fees are set incorrectly.

2. BtokSwap-2 is not fixed and may be centralization risks that may lead to loss of user funds.

3. BtokSwap-3 is not fixed and may cause user’s token dust stuck in the contract in some cases.

4. BtokSwap-4 is not fixed and may affect the result of the discount calculation.

Finding Details:

[BtokSwap-1] Fees can be set to any value

Severity Level	Low
Type	Business Security
Lines	BtokSwapFees.sol#L51-58
Description	In the *setupFees* function, there is no limit to the value range of fees, and wrong settings may cause the user's assets to suffer losses.

```
51    function setupFees(uint8[] memory swapType, uint256[] memory feeRate, string[] memory channel) public onlyExecutor {
52        require(swapType.length == feeRate.length, "TransitSwap: invalid data");
53        require(swapType.length == channel.length, "TransitSwap: invalid data");
54        for(uint256 index; index < swapType.length; index++) {
55            _fees[swapType[index]][channel[index]] = feeRate[index];
56        }
57        emit SetupFees(swapType, feeRate, channel);
58    }
```

Figure 1 Source code of *setupFees* function

Recommendations	It is recommended to limit the value of fees within a reasonable range according to business needs.
Status	Acknowledged.

[BtokSwap-2] Risk of centralization

Severity Level	**Low**
Type	Business Security
Lines	BtokSwapRouterV4.sol#L66-76
Description	In the router contract, the executor role can modify the relevant contract addresses of the swap and cross, and there may be centralization risks that may lead to loss of user funds.

```
66      function changeTransitSwap(address newTransit) external onlyExecutor {
67          address oldTransit = _transit_swap;
68          _transit_swap = newTransit;
69          emit ChangeTransitSwap(oldTransit, newTransit);
70      }
71
72      function changeTransitCross(address newTransit) external onlyExecutor {
73          address oldTransit = _transit_cross;
74          _transit_cross = newTransit;
75          emit ChangeTransitCross(oldTransit, newTransit);
76      }
```

Figure 2 Source code of BtokSwapRouterV4 contract

Recommendations	It is recommended to use the multi-signature wallet to manage the permissions of the executer and owner address.
Status	Acknowledged.

[BtokSwap-3] Function design flaw

Severity Level	Low
Type	Business Security
Lines	BtokSwap.sol #L111-116
Description	The *swap* function lacks the balance check and refund function for srctoken. When the user's funds are not all used for swap, the remaining token dust will stuck in the contract.
Recommendations	It is recommended to add srctoken related refund logic.
Status	Acknowledged.

```
111             if (desc.needTransfer[index] == 1) {
112                 uint afterBalance = IERC20(desc.dstToken).balanceOf(address(this));
113                 TransferHelper.safeTransfer(desc.dstToken, desc.receiver, afterBalance.sub(beforeBalance));
114             } else if (desc.needTransfer[index] == 2) {
115                 TransferHelper.safeTransferETH(desc.receiver, address(this).balance.sub(beforeBalance));
116             }
```

Figure 3 Source code of related code

[BtokSwap-4] Token decimals may not be considered

Severity Level	Info
Type	Business Security
Lines	BtokSwapFees.sol #L69-86
Description	In the calculation of the rate discount, if the decimals of the supported tokens are different, it may affect the result of the discount calculation.

```
66    /**
67     * @dev Returns the swap of the current fees.
68     */
69    function getFeeRate(address trader, uint256 tradeAmount, uint8 swapType, string memory channel) public view r
70        uint256 feeRate = _fees[swapType][channel];
71        if (feeRate == 0) {
72            feeRate = _fees[swapType]["default"];
73            require(feeRate > 0, "TransitSwap: invalid swapType");
74        }
75        if(feeRate == 1) {
76            payFees = 0;
77        } else {
78            uint256 normalPayFees = tradeAmount.mul(feeRate).div(10000);
79            payFees = normalPayFees;
80            if (_support_discount) { //折扣计算逻辑
81                uint256 sumTokenBalance;
82                for (uint256 index; index < _support_tokens.length; index++) {
83                    if (_support_tokens[index] != address(0)) {
84                        sumTokenBalance = sumTokenBalance.add(IERC20(_support_tokens[index]).balanceOf(trader));
85                    }
86                }
```

Figure 4 Source code of *getFeeRate* function

Recommendations	It is recommended to ensure that the supported tokens have the same decimals depending on the business, or to supplement the corresponding calculation logic for different decimals.
Status	Acknowledged.

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.

Likelihood \ Impact	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
		Canister upgrade risk
		Third-party Protocol Interface Consistency
3	Business Security	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.

BEOSIN
Blockchain Security

Official Website

https://www.beosin.com

Telegram

https://t.me/+dD8Bnqd133RmNWNl

Twitter

https://twitter.com/Beosin_com

Email

Contact@beosin.com