BEOSIN
Blockchain Security

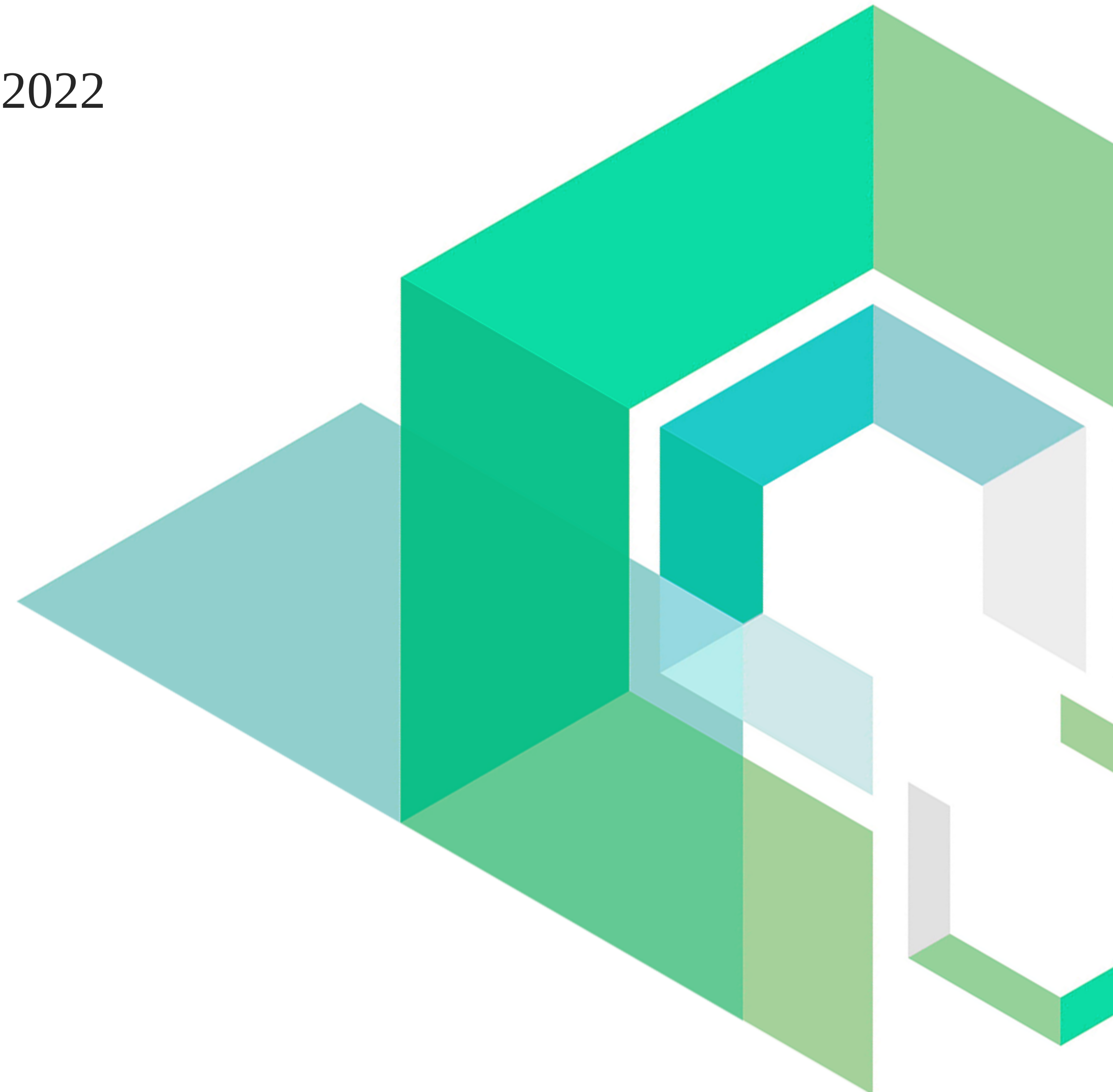
SwapFinder

Smart Contract Security Audit

V1.0

No. 202209021725

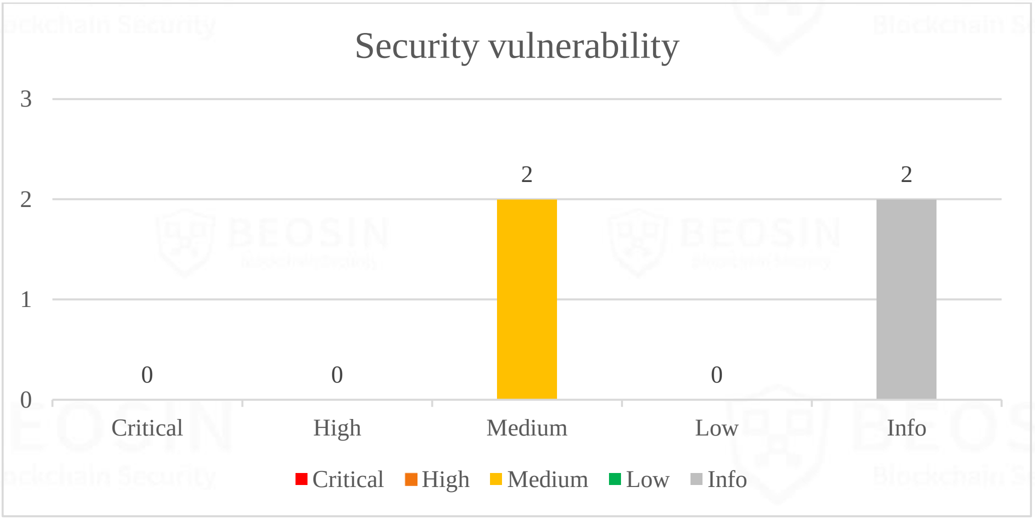
Sep 2nd, 2022



Contents

Summary of audit results

After auditing, 2 Medium-risk and 2 Info-risk items were identified in the SwapFinder 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:**

1. For the project party, it is necessary to pay attention to the possible loss of contract funds, phishing attacks and fee-free swap risks in SwapFinder-2; For the user, it is necessary to pay attention to check whether the address of the pair contract(srcReceiver) and the payment address(dstReceiver) in the swap parameter are correct when interacting.

- **Project Description:**

1. Business overview

The SwapFinder includes a contract that provides the function of exchanging tokens. Users can exchange various tokens through the contract, and support direct swap through BNB to tokens or swap tokens to BNB. Through contract exchange, users need to be charged a certain amount of handling fees and commission as a reward for referrers.

1 Overview

1.1 Project Overview

Project Name	SwapFinder
Platform	BNB Chain
Audit scope	https://github.com/SwapFinder/SwapFinderContract

Commit Hash	405caf18a80471058e238800081f3667b46d0cf3(inital)
	b2a2150b8f82487857db61e670058e4f9a62dbab(final)

1.2 Audit Overview

Audit work duration: August 30, 2022 – September 02, 2022

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

Audit team: Beosin Security Team

2 Findings

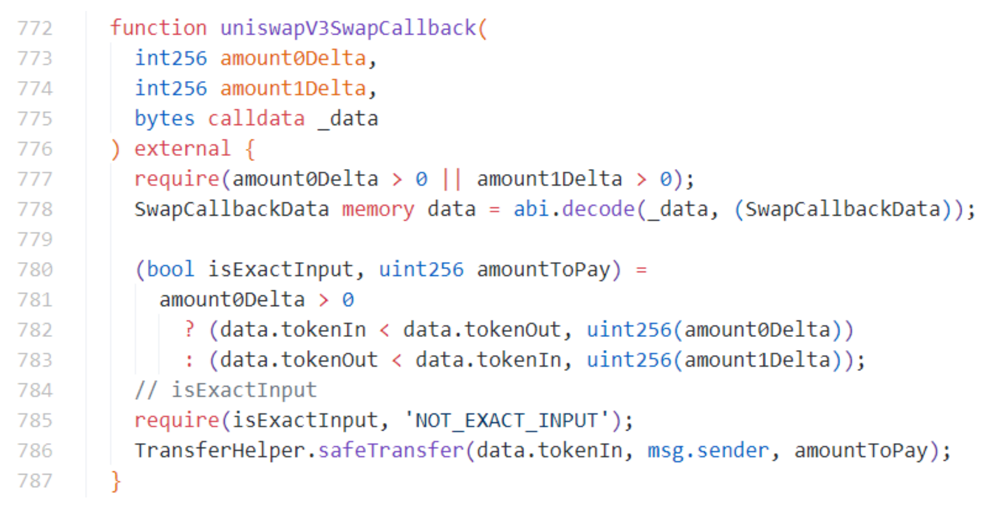
Index	Risk description	Severity level	Status
SwapFinder-1	The *uniswapV3SwapCallback* function lacks access control	Medium	Fixed
SwapFinder-2	The parameters of *swapPlus* function can be arbitrarily specified	Medium	Acknowledged
SwapFinder-3	Missing event trigger	Info	Fixed
SwapFinder-4	Redundant code	Info	Partially Fixed

Status Notes:

- SwapFinder-2 is unfixed and will cause loss of contract funds, free swap fee, Phishing attack, etc.
- SwapFinder-4 is partially fixed and will not cause any issues.

[SwapFinder-1] The *uniswapV3SwapCallback* function lacks access control

Severity Level	Medium
Type	General Vulnerability
Lines	AggregationExecutor.sol#L772-787
Description	The visibility of *uniswapV3SwapCallback* function is external, and there is no access control, Anyone can call this function to withdraw funds in the contract.



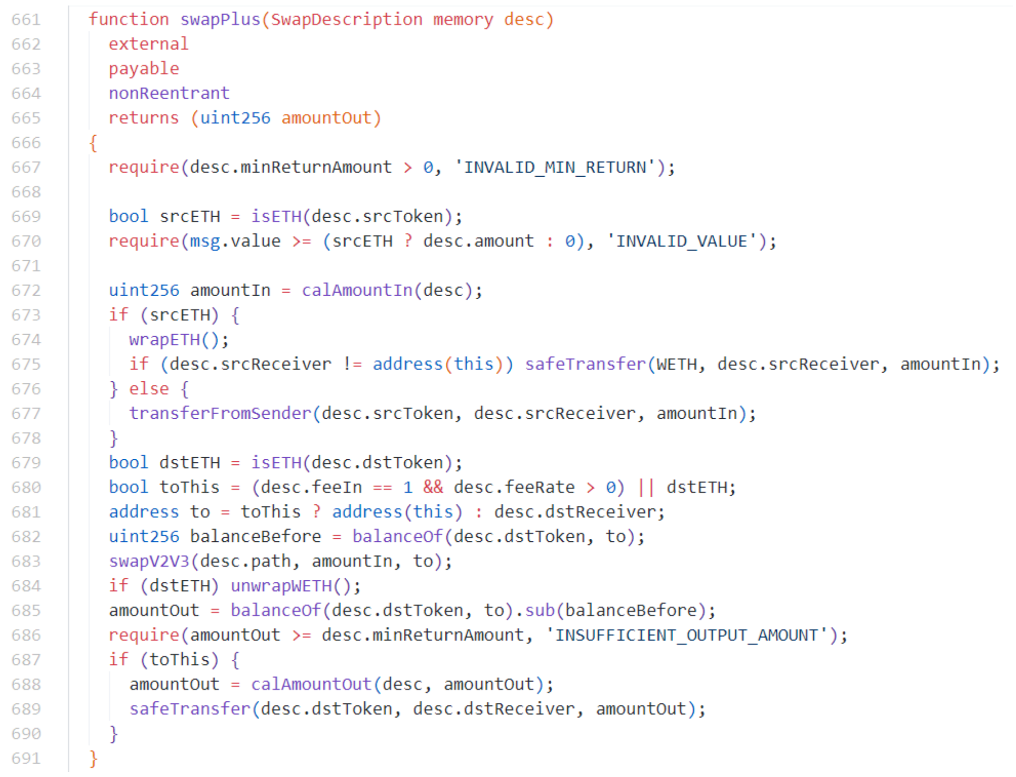
```
772  function uniswapV3SwapCallback(
773    int256 amount0Delta,
774    int256 amount1Delta,
775    bytes calldata _data
776  ) external {
777    require(amount0Delta > 0 || amount1Delta > 0);
778    SwapCallbackData memory data = abi.decode(_data, (SwapCallbackData));
779
780    (bool isExactInput, uint256 amountToPay) =
781      amount0Delta > 0
782        ? (data.tokenIn < data.tokenOut, uint256(amount0Delta))
783        : (data.tokenOut < data.tokenIn, uint256(amount1Delta));
784    // isExactInput
785    require(isExactInput, 'NOT_EXACT_INPUT');
786    TransferHelper.safeTransfer(data.tokenIn, msg.sender, amountToPay);
787  }
```

Figure 1 Source code of *uniswapV3SwapCallback* function

Recommendations	It is recommended to restrict the function to only be accessed by the pair contract of Uniswap.
Status	Fixed. The project party has deleted the function.

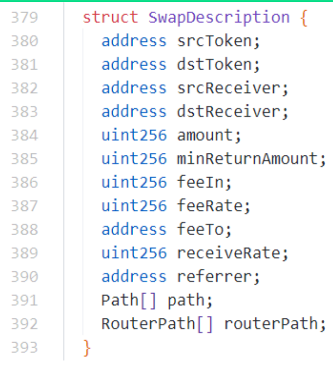
[SwapFinder-2] The parameters of *swapPlus* function can be arbitrarily specified

Severity Level	Medium
Type	General Vulnerability
Lines	AggregationExecutor.sol#L661-691
Description	Users can arbitrarily specify all parameters during swap, When the feeRate and receiveRate are specified as 0, the fee can be waived for swap; when a larger feeRate and receiveRate are specified, and the referrer and feeTo are set as their own addresses, the funds in the contract can be withdrawn; and when the front-end is hijacked by an attacker, these parameters can be exploited and designated as malicious in order to conduct a phishing attack, where the user may lose funds.



```
661  function swapPlus(SwapDescription memory desc)
662    external
663    payable
664    nonReentrant
665    returns (uint256 amountOut)
666  {
667    require(desc.minReturnAmount > 0, 'INVALID_MIN_RETURN');
668
669    bool srcETH = isETH(desc.srcToken);
670    require(msg.value >= (srcETH ? desc.amount : 0), 'INVALID_VALUE');
671
672    uint256 amountIn = calAmountIn(desc);
673    if (srcETH) {
674      wrapETH();
675      if (desc.srcReceiver != address(this)) safeTransfer(WETH, desc.srcReceiver, amountIn);
676    } else {
677      transferFromSender(desc.srcToken, desc.srcReceiver, amountIn);
678    }
679    bool dstETH = isETH(desc.dstToken);
680    bool toThis = (desc.feeIn == 1 && desc.feeRate > 0) || dstETH;
681    address to = toThis ? address(this) : desc.dstReceiver;
682    uint256 balanceBefore = balanceOf(desc.dstToken, to);
683    swapV2V3(desc.path, amountIn, to);
684    if (dstETH) unwrapWETH();
685    amountOut = balanceOf(desc.dstToken, to).sub(balanceBefore);
686    require(amountOut >= desc.minReturnAmount, 'INSUFFICIENT_OUTPUT_AMOUNT');
687    if (toThis) {
688      amountOut = calAmountOut(desc, amountOut);
689      safeTransfer(desc.dstToken, desc.dstReceiver, amountOut);
690    }
691  }
```

Figure 2 Source code of *swapPlus* function



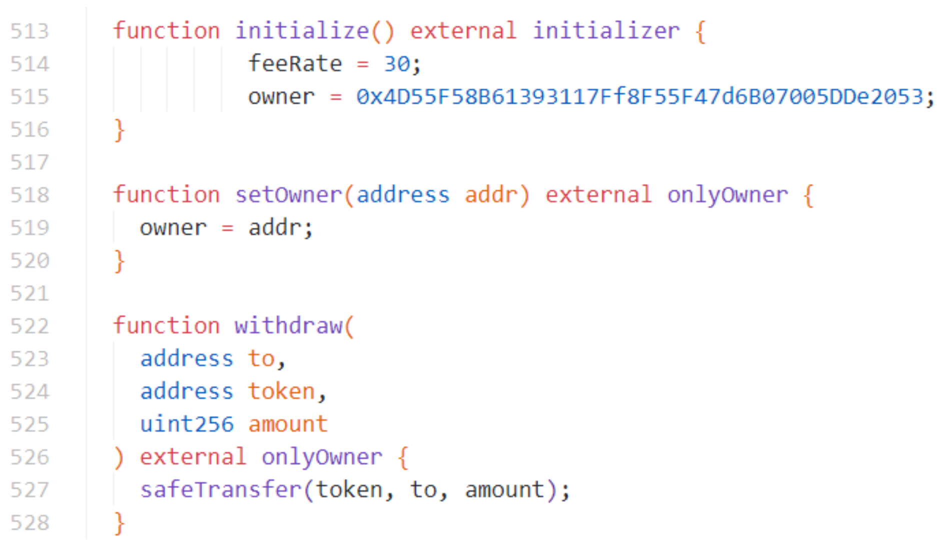
```
struct SwapDescription {
  address srcToken;
  address dstToken;
  address srcReceiver;
  address dstReceiver;
  uint256 amount;
  uint256 minReturnAmount;
  uint256 feeIn;
  uint256 feeRate;
  address feeTo;
  uint256 receiveRate;
  address referrer;
  Path[] path;
  RouterPath[] routerPath;
}
```

Figure 3 Source code of AggregationExecutor contract

Recommendations	It is recommended to minimize user-controlled parameters based on business logic.
Status	Acknowledged.

[SwapFinder-3] Missing event trigger

Severity Level	Info
Type	Coding Conventions
Lines	AggregationExecutor.sol #L513-528
Description	In the SwapFinder contract, there is no event trigger for following functions. It is recommended to add the corresponding event.



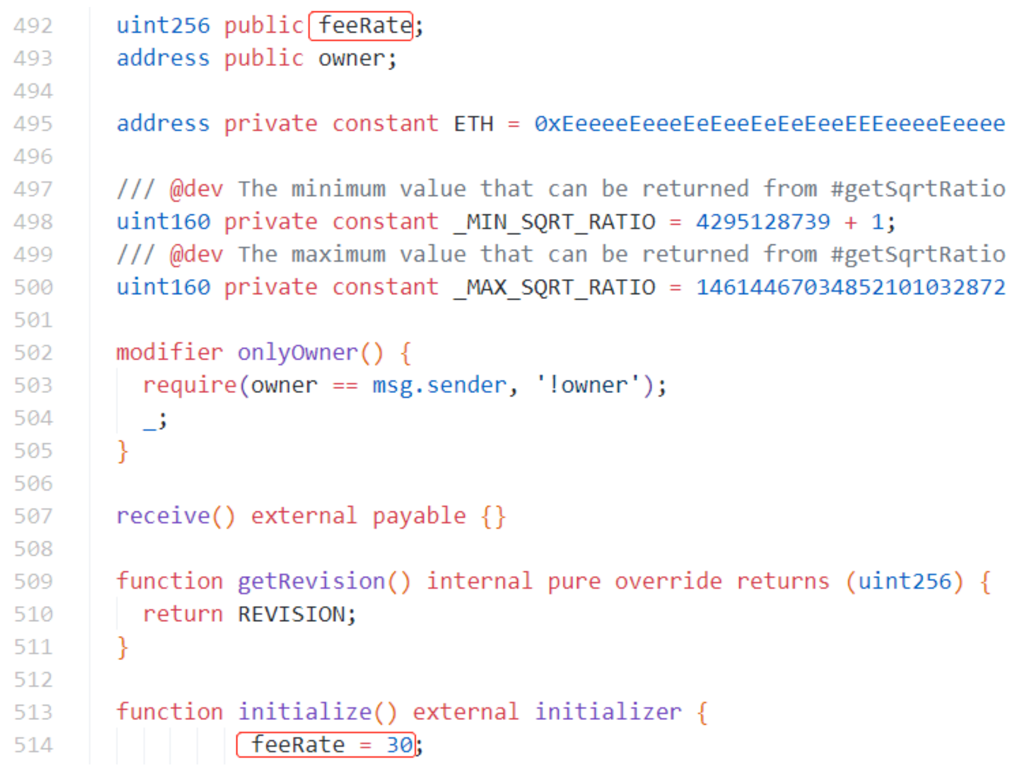
```
513  function initialize() external initializer {
514          feeRate = 30;
515          owner = 0x4D55F58B61393117Ff8F55F47d6B07005DDe2053;
516  }
517
518  function setOwner(address addr) external onlyOwner {
519    owner = addr;
520  }
521
522  function withdraw(
523    address to,
524    address token,
525    uint256 amount
526  ) external onlyOwner {
527    safeTransfer(token, to, amount);
528  }
```

Figure 4 Source code of AggregationExecutor contract

Recommendations	It is recommended to add function events.
Status	Fixed. The project party has deleted related code.

[SwapFinder-4] Redundant code

Severity Level	Info
Type	Coding Conventions
Lines	SwapFinder.sol#L492, L514
Description	In the SwapFinder contract, there is some redundant code that is not used.



```
492  uint256 public feeRate;
493  address public owner;
494
495  address private constant ETH = 0xEeeeeEeeeEeEeeEeEeeEEEeeeeEeeee
496
497  /// @dev The minimum value that can be returned from #getSqrtRatio
498  uint160 private constant _MIN_SQRT_RATIO = 4295128739 + 1;
499  /// @dev The maximum value that can be returned from #getSqrtRatio
500  uint160 private constant _MAX_SQRT_RATIO = 1461446703485210103287
501
502  modifier onlyOwner() {
503    require(owner == msg.sender, '!owner');
504    _;
505  }
506
507  receive() external payable {}
508
509  function getRevision() internal pure override returns (uint256) {
510    return REVISION;
511  }
512
513  function initialize() external initializer {
514        feeRate = 30;
```

Figure 5 Source code of AggregationExecutor contract

Recommendations	It is recommended to remove redundant code.
Status	Partially Fixed. The project party has deleted a part of related code.

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	Compiler Version Security
		Deprecated Items
		Redundant Code
		require/assert Usage
		Gas Consumption
2	General Vulnerability	Integer Overflow/Underflow
		Reentrancy
		Pseudo-random Number Generator (PRNG)
		Transaction-Ordering Dependence
		DoS (Denial of Service)
		Function Call Permissions
		call/delegatecall Security
		Returned Value Security
		tx.origin Usage
		Replay Attack
		Overriding Variables
		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.