Smart Contract
Security Audit Report

Table Of Contents

1 Executive Summary

On 2022.12.07, the SlowMist security team received the BitKeep team's security audit application for BKExchange, developed the audit plan according to the agreement of both parties and the characteristics of the project, and finally issued the security audit report.

The SlowMist security team adopts the strategy of "white box lead, black, grey box assists" to conduct a complete security test on the project in the way closest to the real attack.

The test method information:

Test method	Description
Black box testing	Conduct security tests from an attacker's perspective externally.
Grey box testing	Conduct security testing on code modules through the scripting tool, observing the internal running status, mining weaknesses.
White box testing	Based on the open source code, non-open source code, to detect whether there are vulnerabilities in programs such as nodes, SDK, etc.

The vulnerability severity level information:

Level	Description
Critical	Critical severity vulnerabilities will have a significant impact on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities.
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project team should evaluate and consider whether these vulnerabilities need to be fixed.
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.

Level	Description
Suggestion	There are better practices for coding or architecture.

2 Audit Methodology

The security audit process of SlowMist security team for smart contract includes two steps:

Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using automated analysis tools.

Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that was considered during the audit of the smart contract:

Serial Number	Audit Class	Audit Subclass
1	Overflow Audit	-
2	Reentrancy Attack Audit	-
3	Replay Attack Audit	-
4	Flashloan Attack Audit	-
5	Race Conditions Audit	Reordering Attack Audit
6	Permission Vulnerability Audit	Access Control Audit
		Excessive Authority Audit

Serial Number	Audit Class	Audit Subclass
7	Security Design Audit	External Module Safe Use Audit
		Compiler Version Security Audit
		Hard-coded Address Security Audit
		Fallback Function Safe Use Audit
		Show Coding Security Audit
		Function Return Value Security Audit
		External Call Function Security Audit
		Block data Dependence Security Audit
		tx.origin Authentication Security Audit
8	Denial of Service Audit	-
9	Gas Optimization Audit	-
10	Design Logic Audit	-
11	Variable Coverage Vulnerability Audit	-
12	"False Top-up" Vulnerability Audit	-
13	Scoping and Declarations Audit	-
14	Malicious Event Log Audit	-
15	Arithmetic Accuracy Deviation Audit	-
16	Uninitialized Storage Pointer Audit	-

3 Project Overview

3.1 Project Introduction

Project:

BKExchange

Project address:

https://github.com/bitkeepwallet/bkexchange

Commit:

ebc8de83aea4ade060193277c8d92edc19b50952

3.2 Vulnerability Information

The following is the status of the vulnerabilities found in this audit:

NO	Title	Category	Level	Status
N1	Address checking suggestions	Others	Suggestion	Fixed
N2	Check rcipient suggestion	Others	Suggestion	Fixed
N3	Redundant code	Others	Suggestion	Fixed
N4	Risk of excessive authority	Authority Control Vulnerability	Low	Acknowledged

4 Code Overview

4.1 Contracts Description

The main network address of the contract is as follows:

The code was not deployed to the mainnet.

4.2 Visibility Description

The SlowMist Security team analyzed the visibility of major contracts during the audit, the result as follows:

SeaportMarket			
Function Name	Visibility	Mutability	Modifiers
buyByFulfillBasicOrder	Public	Can Modify State	-
_buyByFulfillBasicOrder	Internal	Can Modify State	-
buyByFulfillAvailableAdvancedOrders	Public	Can Modify State	-
_buyByFulfillAvailableAdvancedOrders	Internal	Can Modify State	-
buyByFulfillAdvancedOrder	Public	Can Modify State	-
_buyByFulfillAdvancedOrder	Internal	Can Modify State	-
rescueETH	External	Can Modify State	-
rescueERC20	External	Can Modify State	-
rescueERC721	External	Can Modify State	-
rescueERC1155	External	Can Modify State	-
_transferEth	Internal	Can Modify State	-

TransferHelper			
Function Name	Visibility	Mutability	Modifiers
safeTransferFrom	Internal	Can Modify State	-
safeTransfer	Internal	Can Modify State	-
safeApprove	Internal	Can Modify State	-

TransferHelper			
safeTransferETH	Internal	Can Modify State	-
approveMax	Internal	Can Modify State	-
isETH	Internal	-	-

BKCommon			
Function Name	Visibility	Mutability	Modifiers
setOperator	External	Can Modify State	onlyOwner
pause	External	Can Modify State	onlyOperator
unpause	External	Can Modify State	onlyOperator
rescueERC20	External	Can Modify State	onlyOperator
rescueERC721	External	Can Modify State	onlyOperator
rescueERC1155	External	Can Modify State	onlyOperator
rescueETH	External	Can Modify State	onlyOperator
_transferEth	Internal	Can Modify State	-
_revertWithData	Internal	-	-
<Receive Ether>	External	Payable	-

BKExchangePeriphery			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-

BKExchangePeriphery			
setBKSwapAddress	External	Can Modify State	onlyOwner
setMarketRegistry	External	Can Modify State	onlyOwner
batchBuyWithETH	External	Payable	handleDustETH whenNotPaused nonReentrant
batchBuyWithERC20s	External	Payable	handleDustERC20s whenNotPaused nonReentrant
_trade	Internal	Can Modify State	-
_approveToSwap	Internal	Can Modify State	-
_swapToDesired	Internal	Can Modify State	-
_checkCallResult	Internal	-	-
setOneTimeApproval	External	Can Modify State	onlyOwner

BKExchangeRouter			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
runWithERC20s	External	Payable	whenNotPaused nonReentrant
runWithETH	External	Payable	whenNotPaused nonReentrant

MarketRegistry			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-

MarketRegistry			
addMarket	External	Can Modify State	onlyOwner
setMarketStatus	External	Can Modify State	onlyOwner
setMarketProxy	External	Can Modify State	onlyOwner

4.3 Vulnerability Summary

[N1] [Suggestion] Address checking suggestions

Category: Others

Content

When modifying important addresses in the contract, it is not checked whether the incoming address is a zero address,also check that the address set is the same as before,make sure you set a valid address.

- contracts/MarketRegistry.sol

```
 function addMarket(address proxy, bool isLib) external onlyOwner {
        markets.push(Market(proxy, isLib, true));
        emit SetMarketProxy(markets.length - 1, Market(proxy, isLib, true));
    }

function setMarketProxy(uint256 marketId, address newProxy, bool isLib) external
onlyOwner {
        Market storage market = markets[marketId];
        market.proxy = newProxy;
        market.isLib = isLib;
        emit SetMarketProxy(marketId, markets[marketId]);
    }
```

- contracts/BKExchangePeriphery.sol

```
function setBKSwapAddress(address _bkswap) external onlyOwner {
        bkswap = _bkswap;
        emit SetBKSwapAddress(msg.sender, _bkswap);
```

```
    }
function setMarketRegistry(address _marketRegistry) external onlyOwner {
        marketRegistry = MarketRegistry(_marketRegistry);
        emit SetMarketRegistry(msg.sender, _marketRegistry);
    }
```

Solution

It is recommended to add a zero address check.also check that the address set is the same as before.

Status

Fixed

[N2] [Suggestion] Check rcipient suggestion

Category: Others

Content

- contracts/market/SeaportMarket.sol

Check if `fulfilladvancedOrderDerBuy.rcipient` and `fulfillAvailableAdvancedOrdersBuy.recipient` is address(0), and if it is address(0), NFT will be left in the current contract.

```
function _buyByFulfillAdvancedOrder(
    FulfillAdvancedOrderBuy calldata fulfillAdvancedOrderBuy,
    bool _revertIfTrxFails
) internal {
    bytes memory _data = abi.encodeWithSelector(
        ISeaport.fulfillAdvancedOrder.selector,
        fulfillAdvancedOrderBuy.advancedOrder,
        fulfillAdvancedOrderBuy.criteriaResolvers,
        fulfillAdvancedOrderBuy.fulfillerConduitKey,
        fulfillAdvancedOrderBuy.recipient//SLOWMIST//Check to make sure it's not
addres(0)
    );

    (bool success, ) = SEAPORT1_1.call{value: fulfillAdvancedOrderBuy.currentPrice}
(_data);

    if (!success && _revertIfTrxFails) {
```

```
        // Copy revert reason from call
        assembly {
            returndatacopy(0, 0, returndatasize())
            revert(0, returndatasize())
        }
    }
}

function _buyByFulfillAvailableAdvancedOrders(
    FulfillAvailableAdvancedOrdersBuy memory fulfillAvailableAdvancedOrdersBuy,
    bool _revertIfTrxFails
) internal {
    bytes memory _data = abi.encodeWithSelector(
        ISeaport.fulfillAvailableAdvancedOrders.selector,
        fulfillAvailableAdvancedOrdersBuy.advancedOrders,
        fulfillAvailableAdvancedOrdersBuy.criteriaResolvers,
        fulfillAvailableAdvancedOrdersBuy.offerFulfillments,
        fulfillAvailableAdvancedOrdersBuy.considerationFulfillments,
        fulfillAvailableAdvancedOrdersBuy.fulfillerConduitKey,
        fulfillAvailableAdvancedOrdersBuy.recipient,//SLOWMIST//Check to make sure
it's not addres(0)
        fulfillAvailableAdvancedOrdersBuy.maximumFulfilled
    );

    (bool success, ) = SEAPORT1_1.call{value:
fulfillAvailableAdvancedOrdersBuy.currentPrice}(_data);

    if (!success && _revertIfTrxFails) {
        // Copy revert reason from call
        assembly {
            returndatacopy(0, 0, returndatasize())
            revert(0, returndatasize())
        }
    }
}
```

Solution

Check if recipient is address(0).

Status

Fixed

[N3] [Suggestion] Redundant code

Category: Others

Content

- contracts/market/SeaportMarket.sol

Unused interfaces and constants.

```
address public constant Owner = 0x5DEFa9C83085c7F606CEB3B5f75Fc107945ed7de;


interface IERC20 {
    function transfer(address to, uint256 amount) external returns (bool);
    function balanceOf(address account) external view returns (uint256);
}

interface IERC721 {
    function safeTransferFrom(
        address from,
        address to,
        uint256 id
    ) external;
}

interface IERC1155 {
    function safeTransferFrom(
        address from,
        address to,
        uint256 id,
        uint256 amount,
        bytes calldata data
    ) external;
}
```

Solution

If it is determined that it will not be used, the redundant code can be deleted.

Status

Fixed

[N4] [Low] Risk of excessive authority

Category: Authority Control Vulnerability

Content

- contracts/BKExchangePeriphery.sol

The owner can set the addresses of `bkswap` and `marketRegistry`. If the private key is compromised, the attacker can transfer the money transferred by the user to the current contract by modifying the addresses of `bkswap` and `marketRegistry`.

```
function setBKSwapAddress(address _bkswap) external onlyOwner {
        bkswap = _bkswap;
        emit SetBKSwapAddress(msg.sender, _bkswap);
    }
function setMarketRegistry(address _marketRegistry) external onlyOwner {
        marketRegistry = MarketRegistry(_marketRegistry);
        emit SetMarketRegistry(msg.sender, _marketRegistry);
    }
```

Solution

It is recommended to transfer the Owner and Executor role to TimeLock contract governance, and at least multi-signature management should be used.

Status

Acknowledged; The project party will use multi-signature to manage the owner.

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002212140002	SlowMist Security Team	2022.12.07 - 2022.12.14	Passed

Summary conclusion: The SlowMist security team use a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 1 low risk, 3 suggestion vulnerabilities.

6 Statement

SlowMist issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility based on these.

For the facts that occurred or existed after the issuance, SlowMist is not able to judge the security status of this project, and is not responsible for them. The security audit analysis and other contents of this report are based on the documents and materials provided to SlowMist by the information provider till the date of the insurance report (referred to as "provided information"). SlowMist assumes: The information provided is not missing, tampered with, deleted or concealed. If the information provided is missing, tampered with, deleted, concealed, or inconsistent with the actual situation, the SlowMist shall not be liable for any loss or adverse effect resulting therefrom. SlowMist only conducts the agreed security audit on the security situation of the project and issues this report. SlowMist is not responsible for the background and other conditions of the project.

Official Website

www.slowmist.com

E-mail

team@slowmist.com

Twitter

@SlowMist_Team

Github

https://github.com/slowmist