Smart Contract
Security Audit Report

Table Of Contents

1 Executive Summary

On 2022.11.10, the SlowMist security team received the BitKeep team's security audit application for BKSwap V2, 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:

BKSwapV2

Project address:

https://github.com/bitkeepwallet/bkswapv2

Commit: f1c8812a79bbc89c3bfb4645ffeb109be0f369c8

Review Commit:a24e6006b4b5d02710089c6535ab80b3de45fb01

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Risk of excessive authority	Authority Control Vulnerability	Medium	Ignored
N2	Risk of excessive authority	Authority Control Vulnerability	Medium	Ignored
N3	Design flaw	Design Logic Audit	Medium	Fixed
N4	Missing zero address check	Others	Suggestion	Fixed
N5	Event log missing	Malicious Event Log Audit	Suggestion	Fixed
N6	Potential external call risk	Unsafe External Call Audit	Suggestion	Fixed

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:

AggregationFeature			
Function Name	Visibility	Mutability	Modifiers
swap	Public	Can Modify State	-
_swapForFree	Internal	Can Modify State	-
_swapEth2Token	Internal	Can Modify State	-
_swapToken2Others	Internal	Can Modify State	-
_swapToken2ETH	Internal	Can Modify State	-
_swapToken2token	Internal	Can Modify State	-
_swapToken2white	Internal	Can Modify State	-
_checkCallResult	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	onlyOwner

BKCommon			
rescueETH	External	Can Modify State	onlyOwner
_transferEth	Internal	Can Modify State	-
<Receive Ether>	External	Payable	-

BKFees			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
setFeeTo	External	Can Modify State	onlyOwner whenNotPaused
getFeeTo	External	-	-
setSigner	External	Can Modify State	onlyOwner whenNotPaused
getSigner	External	-	-
checkIsSigner	External	Can Modify State	-

BKRegistry			
Function Name	Visibility	Mutability	Modifiers
setFeature	External	Can Modify State	onlyOwner whenNotPaused
getFeature	External	-	-
setCallTarget	External	Can Modify State	onlyOwner whenNotPaused
isCallTarget	External	-	-
setApproveTarget	External	Can Modify State	onlyOwner whenNotPaused
isApproveTarget	External	-	-

BKSwap			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
setBKRegistryAddress	External	Can Modify State	onlyOwner
<Fallback>	External	Payable	whenNotPaused
_revertWithData	Private	-	-
_returnWithData	Private	-	-

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

4.3 Vulnerability Summary

[N1] [Medium] Risk of excessive authority

Category: Authority Control Vulnerability

Content

- contracts/BKSwap.sol

The Owner permission is too large. Since the contract calls the proxy address obtained from `bkRegistry` and the call parameters are passed in from outside, if the owner permission is lost, setting up a malicious `bkRegistry` contract will lead to the theft of user funds authorized to this contract.

```
function setBKRegistryAddress(address _bkRegistry) external onlyOwner {
    bkRegistry = _bkRegistry;
}
```

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

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

[N2] [Medium] Risk of excessive authority

Category: Authority Control Vulnerability

Content

- contracts/BKFees.sol

In the `BKFees` contract, the `Owner` role has too much authority. In this contract, the Owner role has the right to call the following functions to set the fee to modify the fee. These functions do not limit the scope of the fee modification. As a result, the Owner role can call these functions to arbitrarily modify the transaction fee.

```
function setFeeTo (
    address payable _feeTo,
    address payable _altcoinsFeeTo,
    uint _feeRate
)  external onlyOwner whenNotPaused {
    feeTo = _feeTo;
    altcoinsFeeTo = _altcoinsFeeTo;
```

```
    feeRate = _feeRate;

    emit SetFeeTo(msg.sender, _feeTo, _altcoinsFeeTo, _feeRate);
}
```

Solution

It is recommended to transfer the Owner and Executor role to TimeLock contract governance, and at least multi-signature management should be used. When modifying the handling fee calculation parameter, its value range should be limited.

Status

Ignored; The project party will use multi-signature to manage the owner, And the fee will only be set within a reasonable range.

[N3] [Medium] Design flaw

Category: Design Logic Audit

Content

- contracts/BKSwap.sol

`BkSwap` manages the authorization of user tokens and the logic part of the contract is an extensible logic contract. Once there is a problem with the extended logic contract, the amount authorized by the user may be transferred away.

Solution

A new router contract is added to control the user's authorization limit. When calling, the amount is transferred from the router to the BKSwap contract. BKSwap does not control the user's authorization limit, so even if the proxy contract in BKSwap fails, it will not affect the user's unauthorized amount.

Status

Fixed

[N4] [Suggestion] Missing zero address check

Category: Others

Content

When modifying important addresses in the contract, it is not checked whether the incoming address is a zero address.

- contracts/BKFees.sol

```
constructor (
    address _signer,
    address payable _feeTo,
    address payable _altcoinsFeeToTo,
    uint _feeRate
) {
    signer = _signer;
    feeTo = _feeTo;
    altcoinsFeeTo = _altcoinsFeeToTo;
    feeRate = _feeRate;
}

function setFeeTo (
    address payable _feeTo,
    address payable _altcoinsFeeTo,
    uint _feeRate
)  external onlyOwner whenNotPaused {
    feeTo = _feeTo;
    altcoinsFeeTo = _altcoinsFeeTo;
    feeRate = _feeRate;

    emit SetFeeTo(msg.sender, _feeTo, _altcoinsFeeTo, _feeRate);
}
```

- contracts/BKRegistry.sol

```
function setFeature(
    bytes4 _methodId,
    address _proxy,
    bool _isLib,
    bool _isActive
) external onlyOwner whenNotPaused {

    Feature memory feat = Feature({
        proxy : _proxy,
        isLib : _isLib,
        isActive : _isActive
    });
    features[_methodId] = feat;

    emit SetFeature(msg.sender, _methodId, _proxy, _isLib, _isActive);
}
```

- contracts/BKSwap.sol

```
constructor(address _bkRegistry) {
    bkRegistry = _bkRegistry;
}

function setBKRegistryAddress(address _bkRegistry) external onlyOwner {
    bkRegistry = _bkRegistry;
}
```

Solution

It is recommended to add a zero address check.

Status

Fixed

[N5] [Suggestion] Event log missing

Category: Malicious Event Log Audit

Content

- contracts/BKSwap.sol

`setBKRegistryAddress` key function calls do not record events.

```
function setBKRegistryAddress(address _bkRegistry) external onlyOwner {
    bkRegistry = _bkRegistry;
}
```

Solution

Record key events.

Status

Fixed

[N6] [Suggestion] Potential external call risk

Category: Unsafe External Call Audit

Content

- contracts/utils/TransferHelper.sol

Combined with the code context `to` address is incoming from the outside, when the `to` address is the contract address, it may bring reentrancy risk to the external logic. No exploitable scenarios have been found so far, but given that the code is extensible in the future, we should pay attention to the risk of reentrancy.

```
function safeTransferETH(address to, uint256 value) internal {
    (bool success, ) = to.call{value: value}(new bytes(0));
    require(success, "STE");
}
```

Solution

Add a reentrant lock to the logic, or limit the gas of the call.

Status

Fixed

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002211170002	SlowMist Security Team	2022.11.10 - 2022.11.17	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 3 medium 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