Smart Contract
Security Audit Report

Table Of Contents

1 Executive Summary

On 2025.11.27, the SlowMist security team received the Sigma Money team's security audit application for SigmaMoney Round 7, 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.
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
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

Serial Number	Audit Class	Audit Subclass
7	Security Design 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

Sigma Money protocol is forked from Fx Protocol and Pendle finance.

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Funds stranded in contract due to missing transfer logic	Design Logic Audit	Critical	Fixed

NO	Title	Category	Level	Status
N2	Incorrect parameter input caused collateral redemption logic error.	Design Logic Audit	Critical	Fixed
N3	Repayment amount mismatch between user input and actual converted funds	Design Logic Audit	Medium	Fixed
N4	Missing zero address check	Others	Suggestion	Acknowledged
N5	Risk of position NFT being permanently locked	Design Logic Audit	Critical	Fixed
N6	Unprocessed excess repayment funds	Design Logic Audit	Medium	Fixed

4 Code Overview

4.1 Contracts Description

https://github.com/SigmaMoney/contracts/tree/feat/bsc

Initial audit version: e758c01378c664c987f3ff71e40b1e4fc4b7307d

Final audit version: 3907b6803a08136bd36041a350b815fcb2cd04ec

Audit Scope:

```
- contracts/periphery/facets/LongPositionOperateFacet.sol
```

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:

LongPositionOperateFacet			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	MorphoFlashLoanFacetBase
openOrAddPosition	External	Payable	nonReentrant onlyTopLevelCall
closeOrRemovePosition	External	Can Modify State	nonReentrant onlyTopLevelCall
_closeOrRemove	Internal	Can Modify State	-
_swap	Internal	Can Modify State	-
_checkPositionDebtRatio	Internal	-	-

4.3 Vulnerability Summary

[N1] [Critical] Funds stranded in contract due to missing transfer logic

Category: Design Logic Audit

Content

In the LongPositionOperateFacet contract, the `openOrAddPosition` and `closeOrRemovePosition` functions fail to transfer the final tokens to the caller after executing `_swap` to convert borrowed assets or redeemed collateral into the target token. This results in funds being erroneously stranded in the contract address, leading to severe financial loss for the user.

- contracts/periphery/facets/LongPositionOperateFacet.sol#L74-L102

```
function openOrAddPosition(
  LibRouter.ConvertInParams memory params,
  address pool,
  uint256 positionId,
  bytes calldata data
) external payable nonReentrant onlyTopLevelCall {
  //...
  // swap bnbUSD to other token
  _swap(bnbUSD, tokenOut, bnbUSDAmount, minOut, swapTarget, swapData);
}

function closeOrRemovePosition(
    LibRouter.ConvertInParams memory params,
    address pool,
    uint256 positionId,
    bytes calldata data
  ) external nonReentrant onlyTopLevelCall {
    uint256 amountIn = LibRouter.transferInAndConvert(params, bnbUSD);

    _closeOrRemove(pool, positionId, amountIn, data);
  }

  function _closeOrRemove(address pool, uint256 positionId, uint256 collAmount, bytes
calldata data) internal {
    //...
    _swap(collateralToken, tokenOut,
IERC20(collateralToken).balanceOf(address(this)), minOut, swapTarget, swapData);
  }
```

Solution

It is recommended to transfer the obtained tokens to the user after the _swap is executed.

Status

Fixed

[N2] [Critical] Incorrect parameter input caused collateral redemption logic error.

Category: Design Logic Audit

Content

In the `closeOrRemovePosition` function of the LongPositionOperateFacet contract, `amountIn` represents the amount of bnbUSD (debt tokens) obtained from the swap, but it is erroneously passed as the `collAmount` parameter (the amount of collateral to redeem) to the `_closeOrRemovefunction`. This causes the subsequent `IPoolManager.operate` call to attempt to redeem an amount of collateral numerically equal to the debt token amount, resulting in a severe business logic error.

- contracts/periphery/facets/LongPositionOperateFacet.sol#L109-L118, L120-L142

```
  function closeOrRemovePosition(
    LibRouter.ConvertInParams memory params,
    address pool,
    uint256 positionId,
    bytes calldata data
  ) external nonReentrant onlyTopLevelCall {
    uint256 amountIn = LibRouter.transferInAndConvert(params, bnbUSD);

    _closeOrRemove(pool, positionId, amountIn, data);
  }

  function _closeOrRemove(address pool, uint256 positionId, uint256 collAmount, bytes
calldata data) internal {
    //...
    if (bnbUSDAmount > maxBnbUSD) {
      // close entire position
      IPoolManager(poolManager).operate(pool, positionId, type(int256).min,
type(int256).min);
    } else {
      IPoolManager(pool).operate(pool, positionId, -int256(collAmount), -
int256(bnbUSDAmount));
      _checkPositionDebtRatio(pool, positionId, miscData);
    }
    //...
  }
```

Solution

It is recommended to add the amount of collateral to be redeemed to the data parameter and use the amountIn parameter as the amount of debt to be repaid.

Status

Fixed

[N3] [Medium] Repayment amount mismatch between user input and actual converted funds

Category: Design Logic Audit

Content

In the `_closeOrRemove` function of the LongPositionOperateFacet contract, the repayment amount `bnbUSDAmount` depends on the user-provided `data` decoded value rather than the actual converted `amountIn`. If `bnbUSDAmount` < `amountIn`, the remaining funds will be stranded in the contract without being used for repayment or returned to the user. If `bnbUSDAmount` > `amountIn`, the transaction will revert due to insufficient balance.

- contracts/periphery/facets/MorphoFlashLoanFacetBase.sol#L109-L118, L120-L142

```
  function closeOrRemovePosition(
    LibRouter.ConvertInParams memory params,
    address pool,
    uint256 positionId,
    bytes calldata data
  ) external nonReentrant onlyTopLevelCall {
    uint256 amountIn = LibRouter.transferInAndConvert(params, bnbUSD);

    _closeOrRemove(pool, positionId, amountIn, data);
  }

  function _closeOrRemove(address pool, uint256 positionId, uint256 collAmount, bytes
calldata data) internal {
    address collateralToken = IPool(pool).collateralToken();
    (
      bytes32 miscData,
      uint256 bnbUSDAmount,
      address swapTarget,
      bytes memory swapData,
      address tokenOut,
      uint256 minOut
    ) = abi.decode(data, (bytes32, uint256, address, bytes, address, uint256));

    IERC721(pool).transferFrom(msg.sender, address(this), positionId);
    (, uint256 maxBnbUSD) = IPool(pool).getPosition(positionId);
    if (bnbUSDAmount > maxBnbUSD) {
      // close entire position
      IPoolManager(poolManager).operate(pool, positionId, type(int256).min,
type(int256).min);
    } else {
      IPoolManager(pool).operate(pool, positionId, -int256(collAmount), -
int256(bnbUSDAmount));
      _checkPositionDebtRatio(pool, positionId, miscData);
    }
    //...
  }
```

Solution

It is recommended to use amountIn directly as the repayment amount.

Status

Fixed

[N4] [Suggestion] Missing zero address check

Category: Others

Content

In the LongPositionOperateFacet contract, the `constructor` function lacks a zero address check for `poolManager` and `bnbUSD`.

- contracts/periphery/facets/LongPositionOperateFacet.sol#L55-L63

```
  constructor(
    address _morpho,
    address _poolManager,
    address _bnbUSD,
    address _whitelist
  ) MorphoFlashLoanFacetBase(_morpho, _whitelist) {
    poolManager = _poolManager;
    bnbUSD = _bnbUSD;
  }
```

Solution

It is recommended to add a zero address check.

Status

Acknowledged

[N5] [Critical] Risk of position NFT being permanently locked

Category: Design Logic Audit

Content

In the `_closeOrRemove` function of the LongPositionOperateFacet contract, the code first transfers the user's position NFT to the contract address, but lacks the logic to return the NFT to the user before the function completes execution, causing the user to permanently lose control of the position and resulting in asset loss.

- contracts/periphery/facets/LongPositionOperateFacet.sol#L120-L142

```
  function _closeOrRemove(address pool, uint256 positionId, uint256 collAmount, bytes
calldata data) internal {
    //...
    IERC721(pool).transferFrom(msg.sender, address(this), positionId);
    (, uint256 maxBnbUSD) = IPool(pool).getPosition(positionId);
    if (bnbUSDAmount > maxBnbUSD) {
      // close entire position
      IPoolManager(poolManager).operate(pool, positionId, type(int256).min,
type(int256).min);
    } else {
      IPoolManager(pool).operate(pool, positionId, -int256(collAmount), -
int256(bnbUSDAmount));
      _checkPositionDebtRatio(pool, positionId, miscData);
    }

    _swap(collateralToken, tokenOut,
IERC20(collateralToken).balanceOf(address(this)), minOut, swapTarget, swapData);
  }
```

Solution

It is recommended to add IERC721(pool).transferFrom(address(this), msg.sender, positionId) after executing the operate operation and before calling _swap to return the position NFT to the user.

Status

Fixed

[N6] [Medium] Unprocessed excess repayment funds

Category: Design Logic Audit

Content

In the `_closeOrRemove` function of the LongPositionOperateFacet contract, when the user-provided repayment amount (`amountIn`) exceeds the actual debt (`maxBnbUSD`), the `IPoolManager(pool).operate` call with `type(int256).min` only deducts the exact debt amount required. The surplus bnbUSD (`amountIn` - `maxBnbUSD`) remains locked in the contract without any logic to refund or convert these excess funds back to the user, resulting in permanent fund loss.

- contracts/periphery/facets/LongPositionOperateFacet.sol#L120-L142

```
  function _closeOrRemove(address pool, uint256 positionId, uint256 collAmount, bytes
calldata data) internal {
    address collateralToken = IPool(pool).collateralToken();
    (
      bytes32 miscData,
      uint256 bnbUSDAmount,
      address swapTarget,
      bytes memory swapData,
      address tokenOut,
      uint256 minOut
    ) = abi.decode(data, (bytes32, uint256, address, bytes, address, uint256));

    IERC721(pool).transferFrom(msg.sender, address(this), positionId);
    (, uint256 maxBnbUSD) = IPool(pool).getPosition(positionId);
    if (bnbUSDAmount > maxBnbUSD) {
      // close entire position
      IPoolManager(poolManager).operate(pool, positionId, type(int256).min,
type(int256).min);
    } else {
      IPoolManager(pool).operate(pool, positionId, -int256(collAmount), -
int256(bnbUSDAmount));
      _checkPositionDebtRatio(pool, positionId, miscData);
    }

    _swap(collateralToken, tokenOut,
IERC20(collateralToken).balanceOf(address(this)), minOut, swapTarget, swapData);
  }
```

Solution

It is recommended to refund the remaining repayment funds to the user or transfer them to the revenuePool.

Status

Fixed

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002511270002	SlowMist Security Team	2025.11.27 - 2025.11.27	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 critical risk, 2 medium risk, 1 suggestion.

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.

SLOWMIST

Official Website

www.slowmist.com

E-mail

team@slowmist.com

Twitter

@SlowMist_Team

Github

https://github.com/slowmist