Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2025.05.19, the SlowMist security team received the Sigma Money team's security audit application for SigmaMoney round 1, 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

This 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	Redundant code	Others	Suggestion	Fixed
N2	Risk of excessive authority	Authority Control Vulnerability Audit	Medium	Acknowledged

NO	Title	Category	Level	Status
N3	Inaccurate function naming and comments	Others	Suggestion	Fixed
N4	Missing zero address check	Others	Suggestion	Fixed
N5	Ignore function return values	Others	Suggestion	Fixed
N6	Encoding validity check is not comprehensive	Design Logic Audit	Low	Fixed

4 Code Overview

4.1 Contracts Description

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

Initial audit version: 4b4a85c8c8c0a292173d3d0b4d0c5544d1081f46

Final audit version: a7047fba8f7c79f5b9dc8a83fc97c09da11a1bc4

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:

PancakeV3SpotPriceReader			
Function Name	Visibility	Mutability	Modifiers
getSpotPrice	External	-	-
_getSpotPriceByAerodromeCL	Internal	-	-
_getPool	Internal	-	-

abFXN			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	External	Can Modify State	initializer
totalAssets	Public	-	-
harvest	External	Can Modify State	-
_deposit	Internal	Can Modify State	-
_withdraw	Internal	Can Modify State	-

PoolManager			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	External	Can Modify State	initializer
initializeV2	External	Can Modify State	onlyRegisteredPool reinitializer
getPoolInfo	External	-	-
operate	External	Can Modify State	onlyRegisteredPool nonReentrant whenNotPaused
redeem	External	Can Modify State	onlyRegisteredPool nonReentrant whenNotPaused
rebalance	External	Can Modify State	onlyRegisteredPool nonReentrant whenNotPaused onlyFxUSDSave
rebalance	External	Can Modify State	onlyRegisteredPool nonReentrant whenNotPaused onlyFxUSDSave
liquidate	External	Can Modify State	onlyRegisteredPool nonReentrant whenNotPaused onlyFxUSDSave
harvest	External	Can Modify State	onlyRegisteredPool onlyRole nonReentrant

PoolManager			
setPause	External	Can Modify State	onlyRole
registerPool	External	Can Modify State	onlyRole
updateRateProvider	External	Can Modify State	onlyRole
updateRewardSplitter	External	Can Modify State	onlyRole onlyRegisteredPool
updatePoolCapacity	External	Can Modify State	onlyRole onlyRegisteredPool
updateThreshold	External	Can Modify State	onlyRole
_updateRewardSplitter	Internal	Can Modify State	-
_updatePoolCapacity	Internal	Can Modify State	-
_updateThreshold	Internal	Can Modify State	-
_scaleUp	Internal	-	-
_scaleUp	Internal	-	-
_scaleDown	Internal	-	-
_scaleDownRoundingUp	Internal	-	-
_scaleDown	Internal	-	-
_beforeRebalanceOrLiquidate	Internal	-	-
_afterRebalanceOrLiquidate	Internal	Can Modify State	-
_changePoolCollateral	Internal	Can Modify State	-
_changePoolDebts	Internal	Can Modify State	-
_getTokenScalingFactor	Internal	-	-

AaveFundingPool			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	BasePool
initialize	External	Can Modify State	initializer
getOpenRatio	External	-	-
getFundingRatio	External	-	-
getOpenFeeRatio	Public	-	-
getCloseFeeRatio	External	-	-
updateOpenRatio	External	Can Modify State	onlyRole
updateCloseFeeRatio	External	Can Modify State	onlyRole
updateFundingRatio	External	Can Modify State	onlyRole
_getOpenRatio	Internal	-	-
_updateOpenRatio	Internal	Can Modify State	-
_getCloseFeeRatio	Internal	-	-
_updateCloseFeeRatio	Internal	Can Modify State	-
_getFundingRatio	Internal	-	-
_updateFundingRatio	Internal	Can Modify State	-
_getAverageInterestRate	Internal	-	-
_updateInterestRate	Internal	Can Modify State	-
_updateCollAndDebtIndex	Internal	Can Modify State	-
_deductProtocolFees	Internal	-	-

SigmaClisBNBPriceOracle			
Function Name	Visibility	Mutability	Modifiers

SigmaClisBNBPriceOracle

<Constructor>	Public	Can Modify State	SpotPriceOracleBase
getSigmaClisBNBUSDTSpotPrice	External	-	-
getSigmaClisBNBUSDTSpotPrices	External	-	-
getPrice	Public	-	-
getExchangePrice	Public	-	-
getLiquidatePrice	External	-	-
getRedeemPrice	External	-	-
updateOnchainSpotEncodings	External	Can Modify State	onlyOwner
updateMaxPriceDeviation	External	Can Modify State	onlyOwner
_updateMaxPriceDeviation	Private	Can Modify State	-
_getSlisBNBBNBSpotPrice	Internal	-	-
_getBNBUSDTSpotPrice	Internal	-	-
_getSlisBNBUSDSpotPrice	Internal	-	-

SigmaClisBNBSYSlisBNBRateProvider

Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
getRate	External	-	-
getX	External	-	-

SigmaClisBNBSY

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

SigmaClisBNBSY			
initialize	External	Can Modify State	initializer
_deposit	Internal	Can Modify State	-
_redeem	Internal	Can Modify State	-
exchangeRate	Public	-	-
_previewDeposit	Internal	-	-
_previewRedeem	Internal	-	-
getTokensIn	Public	-	-
getTokensOut	Public	-	-
isValidTokenIn	Public	-	-
isValidTokenOut	Public	-	-
assetInfo	External	-	-

SigmaController			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	Ownable
deposit	External	Can Modify State	nonReentrant
redeem	External	Can Modify State	nonReentrant

4.3 Vulnerability Summary

[N1] [Suggestion] Redundant code

Category: Others

Content

In the AaveFundingPool contract, the `baseAsset` variable is unused and uninitialized.

- contracts/core/pool/AaveFundingPool.sol#L49

```
address private immutable baseAsset;
```

Solution

It is recommended to delete the redundant code.

Status

Fixed

[N2] [Medium] Risk of excessive authority

Category: Authority Control Vulnerability Audit

Content

1.In the SigmaClisBNBPriceOracle contract, the `Owner` role can modify important parameters in the contract.

- contracts/price-oracle/SigmaClisBNBPriceOracle.sol#L130-L135, L139-L141

```
  function updateOnchainSpotEncodings(bytes memory encoding) external onlyOwner {}
  function updateMaxPriceDeviation(uint256 newMaxPriceDeviation) external onlyOwner
{}
```

Solution

In the short term, transferring owner ownership to multisig contracts is an effective solution to avoid single-point risk. But in the long run, it is a more reasonable solution to implement a privilege separation strategy and set up multiple privileged roles to manage each privileged function separately. And the authority involving user funds should be managed by the community, and the EOA address can manage the authority involving emergency contract suspension. This ensures both a quick response to threats and the safety of user funds.

Status

Acknowledged

[N3] [Suggestion] Inaccurate function naming and comments

Category: Others

Content

In the SigmaClisBNBPriceOracle contract, the name of the `_getSlisBNBUSDSpotPrice` function implies obtaining the slisBNB/USD price, while the actual code calculates the slisBNB/USDT price; at the same time, the comment incorrectly uses "slisBNB/USD" to describe the calculation result.

- contracts/price-oracle/SigmaClisBNBPriceOracle.sol#L196-L208

```
  function _getSlisBNBUSDSpotPrice()
    internal
    view
    returns (uint256 chainlinkPrice, uint256 minPrice, uint256 maxPrice)
  {
    (uint256 price0, uint256 minPrice0, uint256 maxPrice0) =
_getSlisBNBBNBSpotPrice();
    (uint256 price1, uint256 minPrice1, uint256 maxPrice1) = _getBNBUSDTSpotPrice();

    // slisBNBUSDPrice = slisBNBBNBPrice * bnbUSDTPrice / 1e18
    chainlinkPrice = (price0 * price1) / 1e18;
    minPrice = (minPrice0 * minPrice1) / 1e18;
    maxPrice = (maxPrice0 * maxPrice1) / 1e18;
  }
```

Solution

It is recommended to change the function name to _getSlisBNBUSDTSpotPrice and change the comment description to "slisBNB/USDT".

Status

Fixed

[N4] [Suggestion] Missing zero address check

Category: Others

Content

1.In the SigmaClisBNBPriceOracle contract, the `constructor` function lacks a zero address check for the address type parameter.

- contracts/price-oracle/SigmaClisBNBPriceOracle.sol#L43-L52

```
  constructor(
    address _spotPriceOracle,
    address _listaStakeManager,
    bytes32 _Chainlink_BNB_USD_Spot
  ) SpotPriceOracleBase(_spotPriceOracle) {
    LISTA_STAKE_MANAGER = _listaStakeManager;
    Chainlink_BNB_USD_Spot = _Chainlink_BNB_USD_Spot;

    _updateMaxPriceDeviation(1e16); // 1%
  }
```

2.In the SigmaClisBNBSY contract, the `constructor` function lacks a zero address check for the address type parameter.

- contracts/scy/SigmaClisBNBSY.sol#L16-L26

```
  constructor(
    address _listaStakeManager,
    address _slisBnb,
    address _clisBnbSwap,
    address _delegatee
  ) SYBaseUpg(_slisBnb) {
    LISTA_STAKE_MANAGER = _listaStakeManager;
    SLIS_BNB = _slisBnb;
    CLIS_BNB_SWAP = _clisBnbSwap;
    DELEGATEE = _delegatee;
  }
```

3.In the SigmaController contract, the `constructor` function lacks a zero address check for the address type parameter.

- contracts/sigma/SigmaController.sol#L77-L91

```
  constructor(
    IERC20 _slisBNB,
    ISuperComposableYield _sy,
    ISlisBNBProvider _slisBNBProvider,
    IPoolManager _fxPoolManager,
    address _listaLpDelegateTo
  ) Ownable(msg.sender) {
    // Set the addresses for the contracts
    // These should be set to the actual deployed addresses of the respective
contracts
    slisBNB = _slisBNB;
    sy = _sy;
    slisBNBProvider = _slisBNBProvider;
    fxPoolManager = _fxPoolManager;
    listaLpDelegateTo = _listaLpDelegateTo;
  }
```

4.In the abFXN contract, the `constructor` function lacks a zero address check for the address type parameter.

- contracts/base/abFXN.sol#L29-L32

```
  constructor(address _gauge) {
    xbFXN = IGauge(_gauge).stakingToken();
    gauge = _gauge;
  }
```

5.In the PoolManager contract, the `constructor` function lacks a zero address check for the address type parameter.

- contracts/core/PoolManager.sol#L178-L182

```
  constructor(address _fxUSD, address _fxBASE, address _pegKeeper) {
    fxUSD = _fxUSD;
    fxBASE = _fxBASE;
    pegKeeper = _pegKeeper;
  }
```

Solution

It is recommended to add a zero address check.

Status

Fixed

[N5] [Suggestion] Ignore function return values

Category: Others

Content

1.In the SigmaController contract, the `deposit` function did not check the return values of `slisBNB.approve()` and `sy.approve()`.

- contracts/sigma/SigmaController.sol#L100-L143

```
function deposit(
  address _pool,
  uint256 amount,
  uint256 positionId,
  int256 newColl,
  int256 newDebt
) external nonReentrant {
  //...
  slisBNB.approve(address(sy), amount);
  //...
  sy.approve(address(fxPoolManager), uint256(newColl));
  //...
}
```

2.In the SigmaController contract, the `redeem` function did not check the return value of `bnbUSD.approve()`.

- contracts/sigma/SigmaController.sol#L152-L192

```
function redeem(
  address _pool,
  uint256 amount,
  uint256 positionId,
  int256 newColl,
  int256 newDebt
) external nonReentrant {
  //...
    bnbUSD.approve(address(fxPoolManager), uint256(-newDebt));
  //...
}
```

Solution

It is recommended to check the return value of the function.

Status

Fixed

[N6] [Low] Encoding validity check is not comprehensive

Category: Design Logic Audit

Content

In the SigmaClisBNBPriceOracle contract, the `updateOnchainSpotEncodings` function verifies the validity of the `encoding` parameter through the `_getSpotPriceByEncoding` function. However, the `_getSpotPriceByEncoding` function can only provide the most basic verification (checking that `encoding1` is non-zero), not a comprehensive validity verification. For example, the check can also pass when the `encoding` parameter is 0.

- contracts/price-oracle/SigmaClisBNBPriceOracle.sol#L130-L135

```
function updateOnchainSpotEncodings(bytes memory encoding) external onlyOwner {
  // validate encoding
  _getSpotPriceByEncoding(encoding);

  onchainSpotEncodings_BNBUSDT = encoding;
}
```

Solution

It is recommended to add more comprehensive validity checks and not rely solely on the _getSpotPriceByEncoding function.

Status

Fixed

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002505210002	SlowMist Security Team	2025.05.19 - 2025.05.21	Medium Risk

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 medium risk, 1 low risk, 4 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.

SLOWMIST

Official Website

www.slowmist.com

✉

E-mail

team@slowmist.com

Twitter

@SlowMist_Team

Github

https://github.com/slowmist