Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2025.09.24, the SlowMist security team received the Sigma Money team's security audit application for SigmaMoney round 4, 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	Risk of excessive privilege	Authority Control Vulnerability Audit	Medium	Acknowledged
N2	Risk of Integer overflow	Others	Suggestion	Acknowledged

4 Code Overview

4.1 Contracts Description

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

Initial audit version: b545b9e2f18832658f7c81cf29ad4ffc0929ba5b

Final audit version: b545b9e2f18832658f7c81cf29ad4ffc0929ba5b

Audit Scope:

```
- contracts/core/FlashLoans.sol
- contracts/core/PoolConfiguration.sol
- contracts/core/PoolManager.sol
- contracts/core/short/ShortPoolManager.sol
- contracts/fund/strategy/ListaStrategyV2.sol
- contracts/interfaces/IPoolConfiguration.sol
- contracts/periphery/facets/LongPositionEmergencyCloseFacet.sol
- contracts/periphery/facets/MorphoFlashLoanFacetBase.sol
- contracts/periphery/facets/PositionOperateFlashLoanFacetV2.sol
- contracts/periphery/facets/ShortPositionOperateFlashLoanFacet.sol
- contracts/periphery/facets/archived/FlashLoanCallbackFacet.sol
- contracts/periphery/facets/archived/FlashLoanFacetBase.sol
- contracts/periphery/facets/archived/MigrateFacet.sol
- contracts/periphery/facets/archived/PositionOperateFlashLoanFacet.sol
- contracts/price-oracle/BNBPriceOracle.sol
- contracts/price-oracle/InversePriceOracle.sol
- contracts/voting-escrow/SmartWalletWhitelist.sol
- contracts/voting-escrow/interfaces/ISmartWalletChecker.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:

FlashLoans			
Function Name	Visibility	Mutability	Modifiers

FlashLoans			
__FlashLoans_init	Internal	Can Modify State	onlyInitializing
maxFlashLoan	External	-	-
flashFee	Public	-	-
flashLoan	External	Can Modify State	nonReentrant whenNotPaused

PoolConfiguration			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	External	Can Modify State	initializer
isBorrowAllowed	External	-	-
isRedeemAllowed	External	-	-
isFundingEnabled	External	-	-
isStableRepayAllowed	External	-	-
getPoolFeeRatio	External	-	-
getLongPoolFundingRatio	External	-	-
getShortPoolFundingRatio	External	-	-
getAverageInterestRate	External	-	-
checkpoint	External	Can Modify State	-
lock	External	Can Modify State	-
updatePoolFeeRatio	External	Can Modify State	onlyRole
updateLongFundingRatioParameter	External	Can Modify State	onlyRole
updateShortFundingRatioParameter	External	Can Modify State	onlyRole
updateOracle	External	Can Modify State	onlyRole

PoolConfiguration			
updateStableDepegPrice	External	Can Modify State	onlyRole
register	External	Can Modify State	onlyRole
_updateOracle	Internal	Can Modify State	-
_getAverageInterestRate	Internal	-	-
_computeAverageInterestRate	Internal	-	-
_updateBorrowRateSnapshot	Internal	Can Modify State	-
_checkValueTooLarge	Internal	-	-

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

PoolManager			
borrow	External	Can Modify State	onlyCounterparty onlyRegisteredPool nonReentrant
repay	External	Can Modify State	onlyCounterparty onlyRegisteredPool nonReentrant
repayByCreditNote	External	Can Modify State	onlyCounterparty onlyRegisteredPool nonReentrant
liquidateShortPool	External	Can Modify State	onlyCounterparty onlyRegisteredPool nonReentrant
reduceDebt	External	Can Modify State	onlyRegisteredPool onlyRole nonReentrant
setPause	External	Can Modify State	onlyRole
registerPool	External	Can Modify State	onlyRole
updateRateProvider	External	Can Modify State	onlyRole
updatePoolCapacity	External	Can Modify State	onlyRole onlyRegisteredPool
updateThreshold	External	Can Modify State	onlyRole
updateShortBorrowCapacityRatio	External	Can Modify State	onlyRole
_takeAccumulatedPoolFee	Internal	Can Modify State	-
_updatePoolCapacity	Internal	Can Modify State	-
_updateThreshold	Internal	Can Modify State	-
_scaleUp	Internal	-	-
_scaleUp	Internal	-	-
_scaleDown	Internal	-	-
_scaleDownRoundingUp	Internal	-	-
_scaleDown	Internal	-	-

PoolManager			
_handleSupply	Internal	Can Modify State	-
_handleWithdraw	Internal	Can Modify State	-
_handleBorrow	Internal	Can Modify State	-
_handleRepay	Internal	Can Modify State	-
_beforeRebalanceOrLiquidate	Internal	-	-
_afterRebalanceOrLiquidate	Internal	Can Modify State	-
_changePoolCollateral	Internal	Can Modify State	-
_changePoolDebts	Internal	Can Modify State	-
_getTokenScalingFactor	Internal	-	-
_getPoolCollateralInfo	Internal	-	-
_transferCollateralOut	Internal	Can Modify State	-
_transferFrom	Internal	Can Modify State	-

ShortPoolManager			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	External	Can Modify State	initializer
getPoolInfo	External	-	-
getMinRawDebt	External	-	-
operate	Public	Can Modify State	onlyRegisteredPool nonReentrant whenNotPaused onlyTopLevelCall lock

ShortPoolManager			
redeem	External	-	-
redeemByCreditNote	External	Can Modify State	onlyRegisteredPool nonReentrant whenNotPaused
rebalance	External	Can Modify State	onlyRegisteredPool nonReentrant whenNotPaused
rebalance	External	Can Modify State	onlyRegisteredPool nonReentrant whenNotPaused lock
liquidate	External	Can Modify State	onlyRegisteredPool nonReentrant whenNotPaused lock
killPool	External	Can Modify State	onlyRegisteredPool onlyRole
harvest	External	Can Modify State	onlyRegisteredPool onlyRole nonReentrant
reduceDebt	External	Can Modify State	onlyRole onlyRegisteredPool nonReentrant
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
updatePoolMinDebt	External	Can Modify State	onlyRole onlyRegisteredPool
_updateRewardSplitter	Internal	Can Modify State	-
_updatePoolCapacity	Internal	Can Modify State	-
_updatePoolMinDebt	Internal	Can Modify State	-
_getPoolMinRawDebt	Internal	-	-

ShortPoolManager			
_checkRawDebtValues	Internal	-	-
_scaleUp	Internal	-	-
_scaleUp	Internal	-	-
_scaleDown	Internal	-	-
_scaleDownRoundingUp	Internal	-	-
_scaleDown	Internal	-	-
_handleSupply	Internal	Can Modify State	-
_handleWithdraw	Internal	Can Modify State	-
_handleBorrow	Internal	Can Modify State	-
_handleRepay	Internal	Can Modify State	-
_beforeRebalanceOrLiquidate	Internal	-	-
_afterRebalanceOrLiquidate	Internal	Can Modify State	-
_changePoolCollateral	Internal	Can Modify State	-
_changePoolRawDebts	Internal	Can Modify State	-
_getTokenScalingFactor	Internal	-	-

ListaStrategyV2			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	StrategyBase
totalSupply	Public	-	-
deposit	External	Can Modify State	onlyOperator

ListaStrategyV2			
withdraw	External	Can Modify State	onlyOperator
kill	External	Can Modify State	onlyOperator
_harvest	Internal	Can Modify State	-
withdrawPartial	External	Can Modify State	onlyRole
claim	External	Can Modify State	onlyRole
updateTreasury	External	Can Modify State	onlyRole

LongPositionEmergencyCloseFacet			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	MorphoFlashLoanFacetBase
closeOrRemoveLongPositionFlashLoan	External	Can Modify State	nonReentrant onlyTopLevelCall
closeOrRemoveLongPositionFlashLoanWithUSDC	External	Can Modify State	nonReentrant onlyTopLevelCall
onCloseOrRemoveLongPositionFlashLoan	External	Can Modify State	onlySelf
onCloseOrRemoveLongPositionFlashLoanWithUSDC	External	Can Modify State	onlySelf
_redeemCreditNote	Internal	Can Modify State	-
_swap	Internal	Can Modify State	-
_swapWithConverter	Internal	Can Modify State	-
_checkPositionDebtRatio	Internal	-	-

MorphoFlashLoanFacetBase			
Function Name	Visibility	Mutability	Modifiers

MorphoFlashLoanFacetBase			
<Constructor>	Public	Can Modify State	-
_invokeFlashLoan	Internal	Can Modify State	onFlashLoan

PositionOperateFlashLoanFacetV2			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	MorphoFlashLoanFacetBase
openOrAddPositionFlashLoanV2	External	Payable	nonReentrant onlyTopLevelCall
closeOrRemovePositionFlashLoanV2	External	Can Modify State	nonReentrant onlyTopLevelCall
onOpenOrAddPositionFlashLoanV2	External	Can Modify State	onlySelf
onCloseOrRemovePositionFlashLoanV2	External	Can Modify State	onlySelf
_swap	Internal	Can Modify State	-
_checkPositionDebtRatio	Internal	-	-

ShortPositionOperateFlashLoanFacet			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	MorphoFlashLoanFacetBase
openOrAddShortPositionFlashLoan	External	Payable	nonReentrant onlyTopLevelCall
closeOrRemoveShortPositionFlashLoan	External	Can Modify State	nonReentrant onlyTopLevelCall
onOpenOrAddShortPositionFlashLoan	External	Can Modify State	onlySelf
onCloseOrRemoveShortPositionFlashLoan	External	Can Modify State	onlySelf
_swap	Internal	Can Modify State	-

ShortPositionOperateFlashLoanFacet			
_checkPositionDebtRatio	Internal	-	-

FlashLoanCallbackFacet			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
receiveFlashLoan	External	Can Modify State	-

FlashLoanFacetBase			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
_invokeFlashLoan	Internal	Can Modify State	onFlashLoan

MigrateFacet			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	FlashLoanFacetBase
migrateXstETHPosition	External	Can Modify State	nonReentrant
migrateXfrxETHPosition	External	Can Modify State	nonReentrant
onMigrateXstETHPosition	External	Can Modify State	onlySelf
onMigrateXfrxETHPosition	External	Can Modify State	onlySelf
_swapUSDCToFxUSD	Internal	Can Modify State	-
_swapFxUSDToUSDC	Internal	Can Modify State	-
_swapSfrxETHToWstETH	Internal	Can Modify State	-
_swap	Internal	Can Modify State	-
_checkPositionDebtRatio	Internal	-	-

PositionOperateFlashLoanFacet			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	FlashLoanFacetBase
openOrAddPositionFlashLoan	External	Payable	nonReentrant
closeOrRemovePositionFlashLoan	External	Can Modify State	nonReentrant
onOpenOrAddPositionFlashLoan	External	Can Modify State	onlySelf
onCloseOrRemovePositionFlashLoan	External	Can Modify State	onlySelf
_swap	Internal	Can Modify State	-
_checkPositionDebtRatio	Internal	-	-

BNBPriceOracle			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
getBNBUSDSpotPrice	External	-	-
getPrice	Public	-	-
getExchangePrice	Public	-	-
getLiquidatePrice	External	-	-
getRedeemPrice	External	-	-
updateMaxPriceDeviation	External	Can Modify State	onlyRole
_updateMaxPriceDeviation	Private	Can Modify State	-
_getBNBUSDSpotPrice	Internal	-	-
_readSpotPriceByChainlink	Internal	-	-

InversePriceOracle			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
getPrice	Public	-	-
getExchangePrice	Public	-	-
getLiquidatePrice	External	-	-
getRedeemPrice	External	-	-

SmartWalletWhitelist			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	Ownable
commitSetChecker	External	Can Modify State	onlyOwner
applySetChecker	External	Can Modify State	onlyOwner
approveWallet	Public	Can Modify State	onlyOwner
revokeWallet	External	Can Modify State	onlyOwner
check	External	-	-

4.3 Vulnerability Summary

[N1] [Medium] Risk of excessive privilege

Category: Authority Control Vulnerability Audit

Content

1.In the PoolConfiguration contract, the `DEFAULT_ADMIN_ROLE` role is responsible for role permission management; the `OPERATOR_ROLE` is responsible for performing key configuration update operations.

- contracts/core/PoolConfiguration.sol#L351-L388, L395-L408, L414-L425, L429-L431, L435-L439, L444-L448

```
  function updatePoolFeeRatio(
    address pool,
    address recipient,
    uint256 supplyRatio,
    uint256 supplyRatioStep,
    uint256 withdrawFeeRatio,
    uint256 borrowFeeRatio,
    uint256 repayFeeRatio
  ) external onlyRole(OPERATOR_ROLE) {}

  function updateLongFundingRatioParameter(
    address pool,
    uint64 scalarA,
    uint64 scalarB,
    uint64 maxBnbUSDratio
  ) external onlyRole(OPERATOR_ROLE) {}

  function updateShortFundingRatioParameter(
    address pool,
    uint64 scalarC,
    uint64 maxBorrowRatio
  ) external onlyRole(OPERATOR_ROLE) {}

  function updateShortFundingRatioParameter(
    address pool,
    uint64 scalarC,
    uint64 maxBorrowRatio
  ) external onlyRole(OPERATOR_ROLE) {}

  function updateOracle(address newOracle) external onlyRole(OPERATOR_ROLE) {}

  function updateStableDepegPrice(uint256 newStableDepegPrice) external
onlyRole(OPERATOR_ROLE) {}

  function register(bytes32 key, address addr) external onlyRole(OPERATOR_ROLE) {}
```

2.In the PoolManager contract, the `DEFAULT_ADMIN_ROLE` role is responsible for role permission management; the `OPERATOR_ROLE` is responsible for performing key configuration update operations; the `EMERGENCY_ROLE` role can suspend or resume system operation; the `DEBT_REDUCER_ROLE` role has the right to reduce the debt of a specific pool; and the `HARVESTER_ROLE` role can harvest rewards and funds from the pool, obtain performance fees, and harvest rewards.

- contracts/core/PoolManager.sol#L567-L635, L726-L736, L744-L747, L753-L762, L767-L772, L778-L784, L788-L790, L795-L800

```
  function harvest(
    address pool
  )
    external
    onlyRegisteredPool(pool)
    onlyRole(HARVESTER_ROLE)
    nonReentrant
    returns (uint256 amountRewards, uint256 amountFunding)
  {}

  function reduceDebt(
    address pool,
    uint256 amount
  ) external onlyRegisteredPool(pool) onlyRole(DEBT_REDUCER_ROLE) nonReentrant {}

  function setPause(bool status) external onlyRole(EMERGENCY_ROLE) {}

  function registerPool(address pool, uint96 collateralCapacity, uint96 debtCapacity)
external onlyRole(OPERATOR_ROLE) {}

  function updateRateProvider(address token, address provider) external
onlyRole(OPERATOR_ROLE) {}

  function updatePoolCapacity(
    address pool,
    uint96 collateralCapacity,
    uint96 debtCapacity
  ) external onlyRole(OPERATOR_ROLE) onlyRegisteredPool(pool) {}

  function updateThreshold(uint256 newThreshold) external onlyRole(OPERATOR_ROLE) {}

  function updateShortBorrowCapacityRatio(address longPool, uint256 newRatio)
external onlyRole(OPERATOR_ROLE) {}
```

3.In the ShortPoolManager contract, the `DEFAULT_ADMIN_ROLE` role can grant permissions to other roles; the `EMERGENCY_ROLE` role can suspend or resume system operation to respond to emergencies; the `OPERATOR_ROLE` role is responsible for registering new pools, updating pool capacity, updating token exchange rate providers and reward allocators; the `DEBT_REDUCER_ROLE` role can reduce the debt of a specific pool; the `HARVESTER_ROLE` role

can harvest rewards and funds from the pool; the `POOL_KILLER_ROLE` role has the authority to close the pool when the pool is insufficiently collateralized.

- contracts/core/short/ShortPoolManager.sol#L422-L438, L441-L487, L491-L506, L514-L517, L522-L540, L545-L550, L555-L560, L566-L572, L577-L579

```
  function killPool(address pool) external onlyRegisteredPool(pool)
onlyRole(POOL_KILLER_ROLE) {}

  function harvest(
    address pool
  )
    external
    onlyRegisteredPool(pool)
    onlyRole(HARVESTER_ROLE)
    nonReentrant
    returns (uint256 amountRewards, uint256 amountFunding)
  {}

  function reduceDebt(
    address pool,
    uint256 amount
  ) external onlyRole(DEBT_REDUCER_ROLE) onlyRegisteredPool(pool) nonReentrant {}

  function setPause(bool status) external onlyRole(EMERGENCY_ROLE) {}

  function registerPool(
    address pool,
    address splitter,
    uint96 collateralCapacity,
    uint96 debtCapacity,
    uint64 minDebt
  ) external onlyRole(OPERATOR_ROLE) {}

  function updateRateProvider(address token, address provider) external
onlyRole(OPERATOR_ROLE) {}

  function updateRewardSplitter(
    address pool,
    address newSplitter
  ) external onlyRole(OPERATOR_ROLE) onlyRegisteredPool(pool) {}

  function updatePoolCapacity(
    address pool,
    uint96 collateralCapacity,
    uint96 debtCapacity
```

```
    ) external onlyRole(OPERATOR_ROLE) onlyRegisteredPool(pool) {}

    function updatePoolMinDebt(address pool, uint64 minDebt) external
onlyRole(OPERATOR_ROLE) onlyRegisteredPool(pool) {}
```

4.In the ListaStrategyV2 contract, the `DEFAULT_ADMIN_ROLE` role can partially withdraw funds, withdraw non-pool LP tokens and update the treasury address; the `operator` role is responsible for deposits, withdrawals, termination of strategies and execution of arbitrary external calls.

- contracts/fund/strategy/ListaStrategyV2.sol#L52-L57, L59-L66, L68-L74, L86-L93, L95-L100, L102-L109

```
    function deposit(uint256 amount) external onlyOperator {}

    function withdraw(uint256 amount, address recipient) external onlyOperator {}

    function kill() external onlyOperator {}

    function withdrawPartial(uint256 amount) external onlyRole(DEFAULT_ADMIN_ROLE) {}

    function claim(address token, uint256 amount) external onlyRole(DEFAULT_ADMIN_ROLE)
{}

    function updateTreasury(address _treasury) external onlyRole(DEFAULT_ADMIN_ROLE) {}
```

5.In the BNBPriceOracle contract, the `DEFAULT_ADMIN_ROLE` role can manage the permission allocation of other roles; the `OPERATOR_ROLE` role is responsible for updating `updateMaxPriceDeviation`.

- contracts/price-oracle/BNBPriceOracle.sol#L115-L117

```
    function updateMaxPriceDeviation(uint256 newMaxPriceDeviation) external
onlyRole(OPERATOR_ROLE) {}
```

6.In the SmartWalletWhitelist contract, the `owner` role can add or remove wallets from the whitelist and change the external checker through a two-step process to control which smart contracts can interact with the system.

- contracts/voting-escrow/SmartWalletWhitelist.sol#L20-L22, L24-L26, L28-L32, L34-L38

```
    function commitSetChecker(address _checker) external onlyOwner {}

    function applySetChecker() external onlyOwner {}
```

```
function approveWallet(address _wallet) public onlyOwner {}

function revokeWallet(address _wallet) external onlyOwner {}
```

Solution

In the short term, to satisfy business requirements, managing the privileged role through a multi-signature scheme can effectively mitigate single-point risk. In the long term, entrusting these privileged roles to DAO governance can effectively resolve the risk of excessive privilege. During the transition period, managing through a multi-signature scheme combined with delayed transaction execution via a timelock can significantly alleviate the risk of excessive privilege.

Status

Acknowledged

[N2] [Suggestion] Risk of Integer overflow

Category: Others

Content

In the ListaStrategyV2 contract, the `deposit` function uses an unchecked block. The `principal += amount` operation overflows without throwing an exception, causing the principal variable to wrap around to a smaller value. Although reaching the maximum value of `uint256` requires a very large amount of funds in practice, the use of an unchecked block removes the automatic overflow protection in Solidity versions 0.8+, increasing potential risks.

- contracts/fund/strategy/ListaStrategyV2.sol#L52-L57

```
function deposit(uint256 amount) external onlyOperator {
  IMoolahVault(POOL).deposit(amount, address(this));
  unchecked {
    principal += amount;
  }
}
```

Solution

It is recommended to remove the unchecked block and allow the automatic overflow checking mechanism of Solidity 0.8+ to automatically roll back the transaction when an overflow occurs in the principal += amount operation, ensuring system security.

Status

Acknowledged

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002509260001	SlowMist Security Team	2025.09.24 - 2025.09.24	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 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.

Official Website

www.slowmist.com

E-mail

team@slowmist.com

Twitter

@SlowMist_Team

Github

https://github.com/slowmist