Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

2 Audit Methodology

3 Project Overview

3.1 Project Introduction

3.2 Vulnerability Information

4 Code Overview

4.1 Contracts Description

4.2 Visibility Description

4.3 Vulnerability Summary

5 Audit Result

6 Statement

1 Executive Summary

On 2025.08.11, the SlowMist security team received the Sigma Money team's security audit application for SigmaMoney, 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	Risk of excessive authority	Authority Control Vulnerability Audit	Medium	Acknowledged
N2	Missing event records	Others	Suggestion	Fixed

NO	Title	Category	Level	Status
N3	Price deviation check inconsistency	Design Logic Audit	Low	Fixed
N4	Risk of Integer overflow	Integer Overflow and Underflow Vulnerability	Suggestion	Acknowledged

4 Code Overview

4.1 Contracts Description

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

Initial audit version: d2931e80a276825eeeddafd31fd8990692b03ef7

Final audit version: 5bfa678b4cdc930e072f1fc838723f861d8b75fd

https://github.com/SigmaMoney/aladdin-v3-contracts/tree/feat/bsc

Initial audit version: 8b18687de48dc1c82e346a0858325c00ceffbc79

Final audit version: 8b18687de48dc1c82e346a0858325c00ceffbc79

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:

BnbUSDBasePool			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	External	Can Modify State	initializer
previewDeposit	Public	-	onlyValidToken

BnbUSDBasePool			
previewRedeem	External	-	-
nav	External	-	-
getStableTokenPrice	Public	-	-
getStableTokenPriceWithScale	Public	-	-
deposit	External	Can Modify State	nonReentrant onlyValidToken sync
requestRedeem	External	Can Modify State	-
redeem	External	Can Modify State	nonReentrant sync
instantRedeem	External	Can Modify State	nonReentrant sync
rebalance	External	Can Modify State	onlyValidToken nonReentrant sync
rebalance	External	Can Modify State	onlyValidToken nonReentrant sync
liquidate	External	Can Modify State	onlyValidToken nonReentrant sync
arbitrage	External	Can Modify State	onlyValidToken onlyPegKeeper nonReentrant sync
updateStableDepegPrice	External	Can Modify State	onlyRole
updateRedeemCoolDownPeriod	External	Can Modify State	onlyRole
updateInstantRedeemFeeRatio	External	Can Modify State	onlyRole
_update	Internal	Can Modify State	-
_updateStableDepegPrice	Internal	Can Modify State	-
_updateRedeemCoolDownPeriod	Internal	Can Modify State	-

BnbUSDBasePool			
_updateInstantRedeemFeeRatio	Internal	Can Modify State	-
_deposit	Internal	Can Modify State	-
_beforeRebalanceOrLiquidate	Internal	-	-
_afterRebalanceOrLiquidate	Internal	Can Modify State	-
_getTotalStableTokenInPool	Internal	-	-

BnbUSDPriceOracle			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	External	Can Modify State	initializer
getPrice	External	-	-
isPriceAboveMaxDeviation	External	-	-
isPriceBelowMaxDeviation	External	-	-
updateCurvePool	External	Can Modify State	onlyRole
updateMaxPriceDeviation	External	Can Modify State	onlyRole
_getBnbUSDEmaPrice	Internal	-	-
_updateCurvePool	Internal	Can Modify State	-
_updateMaxPriceDeviation	Internal	Can Modify State	-

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

PegKeeper			
initialize	External	Can Modify State	initializer
isBorrowAllowed	External	-	-
isFundingEnabled	External	-	-
isRedeemAllowed	External	-	-
getFxUSDPrice	External	-	-
buyback	External	Can Modify State	onlyRole setContext
stabilize	External	Can Modify State	onlyRole setContext
onSwap	External	Can Modify State	-
updateConverter	External	Can Modify State	onlyRole
updateCurvePool	External	Can Modify State	onlyRole
updatePriceThreshold	External	Can Modify State	onlyRole
_updateConverter	Internal	Can Modify State	-
_updateCurvePool	Internal	Can Modify State	-
_updatePriceThreshold	Internal	Can Modify State	-
_doSwap	Internal	Can Modify State	-
_getFxUSDEmaPrice	Internal	-	-

PoolConfiguration			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	External	Can Modify State	initializer
isBorrowAllowed	External	-	-
isRedeemAllowed	External	-	-

PoolConfiguration			
isFundingEnabled	External	-	-
isStableRepayAllowed	External	-	-
getPoolFeeRatio	External	-	-
getLongPoolFundingRatio	External	-	-
getShortPoolFundingRatio	External	-	-
getAverageInterestRate	External	-	-
checkpoint	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
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	-

PoolManager			
initialize	External	Can Modify State	initializer
getPoolInfo	External	-	-
operate	External	Can Modify State	-
operate	Public	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
borrow	External	Can Modify State	onlyCounterparty onlyRegisteredPool
repay	External	Can Modify State	onlyCounterparty onlyRegisteredPool
repayByCreditNote	External	Can Modify State	onlyCounterparty onlyRegisteredPool
liquidateShortPool	External	Can Modify State	onlyCounterparty onlyRegisteredPool
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

PoolManager			
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	-	-
_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	-	-

PoolManager			
_getPoolCollateralInfo	Internal	-	-
_transferCollateralOut	Internal	Can Modify State	-
_transferFrom	Internal	Can Modify State	-

ProtocolFees			
Function Name	Visibility	Mutability	Modifiers
__ProtocolFees_init	Internal	Can Modify State	onlyInitializing
getFundingExpenseRatio	Public	-	-
getRewardsExpenseRatio	Public	-	-
getLiquidationExpenseRatio	Public	-	-
getHarvesterRatio	Public	-	-
getFlashLoanFeeRatio	Public	-	-
getRedeemFeeRatio	Public	-	-
withdrawAccumulatedPoolFee	External	Can Modify State	-
updateReservePool	External	Can Modify State	onlyRole
updateTreasury	External	Can Modify State	onlyRole
updateOpenRevenuePool	External	Can Modify State	onlyRole
updateCloseRevenuePool	External	Can Modify State	onlyRole
updateMiscRevenuePool	External	Can Modify State	onlyRole
updateExpenseRatio	External	Can Modify State	onlyRole
updateHarvesterRatio	External	Can Modify State	onlyRole
updateFlashLoanFeeRatio	External	Can Modify State	onlyRole
updateRedeemFeeRatio	External	Can Modify State	onlyRole

ProtocolFees			
_updateTreasury	Private	Can Modify State	-
_updateOpenRevenuePool	Private	Can Modify State	-
_updateCloseRevenuePool	Private	Can Modify State	-
_updateMiscRevenuePool	Private	Can Modify State	-
_updateReservePool	Private	Can Modify State	-
_updateMiscData	Private	Can Modify State	-
_updateRewardsExpenseRatio	Private	Can Modify State	-
_updateLiquidationExpenseRatio	Private	Can Modify State	-
_updateFundingExpenseRatio	Private	Can Modify State	-
_updateHarvesterRatio	Private	Can Modify State	-
_updateFlashLoanFeeRatio	Private	Can Modify State	-
_updateRedeemFeeRatio	Private	Can Modify State	-
_accumulatePoolOpenFee	Internal	Can Modify State	-
_accumulatePoolCloseFee	Internal	Can Modify State	-
_accumulatePoolMiscFee	Internal	Can Modify State	-
_takeAccumulatedPoolFee	Internal	Can Modify State	-
_checkValueTooLarge	Internal	-	-
_checkAddressNotZero	Internal	-	-

BasePool			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
__BasePool_init	Internal	Can Modify State	onlyInitializing

BasePool			
operate	External	Can Modify State	onlyPoolManager
redeem	External	Can Modify State	onlyPoolManager
rebalance	External	Can Modify State	onlyPoolManager
rebalance	External	Can Modify State	onlyPoolManager
liquidate	External	Can Modify State	onlyPoolManager
reduceDebt	External	Can Modify State	onlyPoolManager
updateBorrowAndRedeemStatus	External	Can Modify State	onlyRole
updateDebtRatioRange	External	Can Modify State	onlyRole
updateMaxRedeemRatioPerTick	External	Can Modify State	onlyRole
updateRebalanceRatios	External	Can Modify State	onlyRole
updateLiquidateRatios	External	Can Modify State	onlyRole
updatePriceOracle	External	Can Modify State	onlyRole
updateCounterparty	External	Can Modify State	onlyRole
_redeem	Internal	Can Modify State	-
_getRawDebtToRebalance	Internal	-	-
_getTickRawCollAndDebts	Internal	-	-
_rebalanceTick	Internal	Can Modify State	-
_liquidateTick	Internal	Can Modify State	-
_updateCollAndDebtIndex	Internal	Can Modify State	-
_deductProtocolFees	Internal	-	-

PoolStorage			
Function Name	Visibility	Mutability	Modifiers

PoolStorage			
__PoolStorage_init	Internal	Can Modify State	onlyInitializing
supportsInterface	Public	-	-
isBorrowPaused	External	-	-
isRedeemPaused	External	-	-
getTopTick	External	-	-
getNextPositionId	External	-	-
getNextTreeNodeId	External	-	-
getDebtRatioRange	External	-	-
getMaxRedeemRatioPerTick	External	-	-
getRebalanceRatios	External	-	-
getLiquidateRatios	External	-	-
getDebtAndCollateralIndex	External	-	-
getDebtAndCollateralShares	External	-	-
_updatePriceOracle	Internal	Can Modify State	-
_isBorrowPaused	Internal	-	-
_updateBorrowStatus	Internal	Can Modify State	-
_isRedeemPaused	Internal	-	-
_updateRedeemStatus	Internal	Can Modify State	-
_getTopTick	Internal	-	-
_updateTopTick	Internal	Can Modify State	-
_getNextPositionId	Internal	-	-
_updateNextPositionId	Internal	Can Modify State	-

PoolStorage			
_getNextTreeNodeId	Internal	-	-
_updateNextTreeNodeId	Internal	Can Modify State	-
_getDebtRatioRange	Internal	-	-
_updateDebtRatioRange	Internal	Can Modify State	-
_getMaxRedeemRatioPerTick	Internal	-	-
_updateMaxRedeemRatioPerTick	Internal	Can Modify State	-
_getRebalanceRatios	Internal	-	-
_updateRebalanceRatios	Internal	Can Modify State	-
_getLiquidateRatios	Internal	-	-
_updateLiquidateRatios	Internal	Can Modify State	-
_getDebtAndCollateralIndex	Internal	-	-
_updateDebtIndex	Internal	Can Modify State	-
_updateCollateralIndex	Internal	Can Modify State	-
_getDebtAndCollateralShares	Internal	-	-
_updateDebtAndCollateralShares	Internal	Can Modify State	-
_updateDebtShares	Internal	Can Modify State	-
_updateCollateralShares	Internal	Can Modify State	-
_updateCounterparty	Internal	Can Modify State	-

SigmaLongPool			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	BasePool
initialize	External	Can Modify State	initializer

SigmaLongPool			
reduceCollateral	External	Can Modify State	onlyPoolManager
_updateCollAndDebtIndex	Internal	Can Modify State	-
_deductProtocolFees	Internal	-	-

ShortPool			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	BasePool
initialize	External	Can Modify State	initializer
isUnderCollateral	External	Can Modify State	onlyPoolManager
kill	External	Can Modify State	onlyPoolManager
redeemByCreditNote	External	Can Modify State	onlyPoolManager
_updateCollAndDebtIndex	Internal	Can Modify State	-
_deductProtocolFees	Internal	-	-

ListaStrategyV2			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	StrategyBase
totalSupply	Public	-	-
deposit	External	Can Modify State	onlyOperator
withdraw	External	Can Modify State	onlyOperator
kill	External	Can Modify State	onlyOperator
_harvest	Internal	Can Modify State	-
withdrawPartial	External	Can Modify State	onlyRole

ListaStrategyV2			
claim	External	Can Modify State	onlyRole
updateTreasury	External	Can Modify State	onlyRole

MoolahFlashLoanCallbackFacet			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
onMoolahFlashLoan	External	Can Modify State	-

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	-	-

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

InverseBnbPriceOracle			
getPrice	Public	-	-
getExchangePrice	Public	-	-
getLiquidatePrice	External	-	-
getRedeemPrice	External	-	-

LongPositionEmergencyCloseFacet			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	MorphoFlashLoanFacetBase
closeOrRemoveLongPositionFlashLoan	External	Can Modify State	nonReentrant
closeOrRemoveLongPositionFlashLoanWithUSDC	External	Can Modify State	nonReentrant
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	-	-

PositionOperateFlashLoanFacetV2			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	MorphoFlashLoanFacetBase
openOrAddPositionFlashLoanV2	External	Payable	nonReentrant

PositionOperateFlashLoanFacetV2			
closeOrRemovePositionFlashLoanV2	External	Can Modify State	nonReentrant
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
closeOrRemoveShortPositionFlashLoan	External	Can Modify State	nonReentrant
onOpenOrAddShortPositionFlashLoan	External	Can Modify State	onlySelf
onCloseOrRemoveShortPositionFlashLoan	External	Can Modify State	onlySelf
_swap	Internal	Can Modify State	-
_checkPositionDebtRatio	Internal	-	-

ConverterBase			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
<Receive Ether>	External	Payable	-
withdrawFund	External	Can Modify State	-

ConverterBase			
_getPoolType	Internal	-	-
_getAction	Internal	-	-
_getPool	Internal	-	-
_isETH	Internal	-	-
_wrapTokenIfNeeded	Internal	Can Modify State	-
_unwrapIfNeeded	Internal	Can Modify State	-
_approve	Internal	Can Modify State	-

GeneralTokenConverter			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ConverterBase
getTokenPair	External	-	-
queryConvert	External	Can Modify State	-
convert	External	Payable	-
_getTokenPair	Internal	-	-
_getTokenMinter	Internal	-	-
_getCurveTokenByIndex	Internal	-	-
_querySwap	Internal	Can Modify State	-
_swap	Internal	Can Modify State	-
_queryWrap	Internal	-	-
_wrap	Internal	Can Modify State	-
_queryUnwrap	Internal	-	-
_unwrap	Internal	Can Modify State	-

MultiPathConverter			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
convert	External	Payable	-

4.3 Vulnerability Summary

[N1] [Medium] Risk of excessive authority

Category: Authority Control Vulnerability Audit

Content

1.The BnbUSDBasePool contract contains three main role permissions: the `DEFAULT_ADMIN_ROLE` role is granted at initialization and can manage other roles; the `OPERATOR_ROLE` role has the permission to adjust key parameters; and the `PegKeeper` role is the only privileged address that can call the arbitrage function.

- contracts/core/BnbUSDBasePool.sol#L522-L524, L528-L530, L532-L534

```
  function updateStableDepegPrice(uint256 newPrice) external onlyRole(OPERATOR_ROLE)
{}

  function updateRedeemCoolDownPeriod(uint256 newPeriod) external
onlyRole(OPERATOR_ROLE) {}

  function updateInstantRedeemFeeRatio(uint256 newRatio) external
onlyRole(OPERATOR_ROLE) {}

  function arbitrage(
    address srcToken,
    uint256 amountIn,
    address receiver,
    bytes calldata data
  ) external onlyValidToken(srcToken) onlyPegKeeper nonReentrant sync returns
(uint256 amountOut, uint256 bonusOut) {}
```

2.In the BnbUSDPriceOracle contract, only the `DEFAULT_ADMIN_ROLE` role has administrative permissions and can modify important parameters.

- contracts/core/BnbUSDPriceOracle.sol#L94-L96, L101-L106

```
function updateCurvePool(address newPool) external onlyRole(DEFAULT_ADMIN_ROLE) {}

function updateMaxPriceDeviation(
  uint256 newDePegDeviation,
  uint256 newUpPegDeviation
) external onlyRole(DEFAULT_ADMIN_ROLE) {}
```

3.In the PoolConfiguration contract, the `DEFAULT_ADMIN_ROLE` role has the highest permissions and is granted at initialization; the `OPERATOR_ROLE` role has core configuration permissions; any whitelisted pool contract can call the `checkpoint` function to update the lending rate snapshot.

- contracts/core/PoolConfiguration.sol#L305-L316, L330-L367, L374-L387, L393-L404, L408-L410, L414-L418, L423-L427

```
function checkpoint(address pool) external {}

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 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) {}
```

4.In the PoolManager contract, the `DEFAULT_ADMIN_ROLE` role has the highest authority; the `OPERATOR_ROLE` role is responsible for core business configuration; the `EMERGENCY_ROLE` role can pause/resume the entire system; the `HARVESTER_ROLE` role can execute the `harvest` function; the `DEBT_REDUCER_ROLE` role can perform debt reduction operations; the counterparty role can perform lending and repayment operations; the fxBASE contract or anyone who meets the threshold conditions can perform rebalancing and liquidation operations.

- contracts/core/PoolManager.sol#L432-L461, L464-L476, L495-L507, L543-L611, L627-L646, L649-L657, L660-L666, L669-L698, L702-L712, L720-L723, L729-L738, L743-L748, L754-L760, L764-L766, L771-L776

```
  function rebalance(
    address pool,
    address receiver,
    int16 tick,
    uint256 maxFxUSD,
    uint256 maxStable
  )
    external
    onlyRegisteredPool(pool)
    nonReentrant
    whenNotPaused
    onlyFxUSDSave
    returns (uint256 colls, uint256 fxUSDUsed, uint256 stableUsed)
  {}

  function rebalance(
    address pool,
    address receiver,
    uint256 maxFxUSD,
    uint256 maxStable
  )
    external
    onlyRegisteredPool(pool)
    nonReentrant
    whenNotPaused
    onlyFxUSDSave
    returns (uint256 colls, uint256 fxUSDUsed, uint256 stableUsed)
  {}

  function liquidate(
```

```
    address pool,
    address receiver,
    uint256 maxFxUSD,
    uint256 maxStable
  )
    external
    onlyRegisteredPool(pool)
    nonReentrant
    whenNotPaused
    onlyFxUSDSave
    returns (uint256 colls, uint256 fxUSDUsed, uint256 stableUsed)
  {}

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

  function borrow(
    address longPool,
    address shortPool,
    uint256 amount
  ) external onlyCounterparty onlyRegisteredPool(longPool) {}

  function repay(
    address longPool,
    address shortPool,
    uint256 amount
  ) external onlyCounterparty onlyRegisteredPool(longPool) {}

  function repayByCreditNote(
    address longPool,
    address shortPool,
    uint256 amount
  ) external onlyCounterparty onlyRegisteredPool(longPool) {}

  function liquidateShortPool(
    address longPool,
    address shortPool,
    uint256 amountFxUSD,
    uint256 totalBorrowed
  ) external onlyCounterparty onlyRegisteredPool(longPool) returns (uint256
shortfall) {}
```

```
  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) {}
```

5.In the ProtocolFees contract, the `DEFAULT_ADMIN_ROLE` role has the highest authority and can manage other roles; the `OPERATOR_ROLE` role has full authority over fee configuration.

- contracts/core/ProtocolFees.sol#L209-L211, L215-L217, L221-L223, L227-L229, L233-L235, L241-L245, L253-L255, L259-L261, L265-L267

```
  function updateReservePool(address _newReservePool) external
onlyRole(OPERATOR_ROLE) {}

  function updateTreasury(address _newTreasury) external onlyRole(OPERATOR_ROLE) {}

  function updateOpenRevenuePool(address _newPool) external onlyRole(OPERATOR_ROLE)
{}

  function updateCloseRevenuePool(address _newPool) external onlyRole(OPERATOR_ROLE)
{}

  function updateMiscRevenuePool(address _newPool) external onlyRole(OPERATOR_ROLE)
{}

  function updateExpenseRatio(
    uint32 newRewardsRatio,
```

```
    uint32 newFundingRatio,
    uint32 newLiquidationRatio
  ) external onlyRole(OPERATOR_ROLE) {}

  function updateHarvesterRatio(uint32 newRatio) external onlyRole(OPERATOR_ROLE) {}

  function updateFlashLoanFeeRatio(uint32 newRatio) external onlyRole(OPERATOR_ROLE)
{}

  function updateRedeemFeeRatio(uint32 newRatio) external onlyRole(OPERATOR_ROLE) {}
```

6.In the BasePool contract, the `poolManager` role has core business operation permissions and is the only address that can call key functions; the `EMERGENCY_ROLE` role can update lending and redemption status; and the `OPERATOR_ROLE` role is responsible for risk parameter configuration.

- contracts/core/pool/BasePool.sol#L73-L117, L191-L195, L198-L244, L262-L311, L, L330-L381, L384-L395, L404-L407, L412-L414, L418-L420, L425-L427, L432-L434, L438-L440, L444-L446

```
  function operate(
    uint256 positionId,
    int256 newRawColl,
    int256 newRawDebt,
    address owner
  ) external onlyPoolManager returns (uint256, int256, int256, uint256) {}

  function redeem(uint256 rawDebts) external onlyPoolManager returns (uint256
actualRawDebts, uint256 rawColls) {}

  function rebalance(int16 tick, uint256 maxRawDebts) external onlyPoolManager
returns (RebalanceResult memory result) {}

  function rebalance(uint256 maxRawDebts) external onlyPoolManager returns
(RebalanceResult memory result) {}

  function liquidate(
    uint256 maxRawDebts,
    uint256 reservedRawColls
  ) external onlyPoolManager returns (LiquidateResult memory result) {}

  function reduceDebt(uint256 rawAmount) external onlyPoolManager {}

  function updateBorrowAndRedeemStatus(bool borrowStatus, bool redeemStatus) external
onlyRole(EMERGENCY_ROLE) {}

  function updateDebtRatioRange(uint256 minRatio, uint256 maxRatio) external
```

```
onlyRole(OPERATOR_ROLE) {}

  function updateMaxRedeemRatioPerTick(uint256 ratio) external
onlyRole(OPERATOR_ROLE) {}

  function updateRebalanceRatios(uint256 debtRatio, uint256 bonusRatio) external
onlyRole(OPERATOR_ROLE) {}

  function updateLiquidateRatios(uint256 debtRatio, uint256 bonusRatio) external
onlyRole(OPERATOR_ROLE) {}

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

  function updateCounterparty(address newCounterparty) external
onlyRole(OPERATOR_ROLE) {}
```

7.In the SigmaLongPool contract, the contract inherits the permission system of BasePool and expands on it: a new unique permission for reducing collateral is added.

- contracts/core/pool/SigmaLongPool.sol#L79-L89

```
  function reduceCollateral(uint256 amount) external onlyPoolManager {}
```

8.The ShortPool contract inherits the permission system of BasePool and adds functional permissions unique to the short pool.

- contracts/core/short/ShortPool.sol#L71-L81, L88-L90, L93-L95

```
  function isUnderCollateral() external onlyPoolManager returns (bool
underCollateral, uint256 shortfall) {}

  function kill() external onlyPoolManager {}

  function redeemByCreditNote(uint256 creditNoteAmount) external onlyPoolManager
returns (uint256 rawColls) {}
```

9.In the ListaStrategyV2 contract, the `DEFAULT_ADMIN_ROLE` role has the highest authority and can perform high-risk operations; the `OPERATOR_ROLE` role has daily strategy management permissions.

- contracts/fund/strategy/ListaStrategyV2.sol#L47-L52, L54-L61, L63-L69, L81-L88, L90-L95, L97-L100

```
  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) {}
```

10.In the BNBPriceOracle contract, the `DEFAULT_ADMIN_ROLE` role has the highest authority and can manage other roles; the `OPERATOR_ROLE` role has the authority to control key price parameters.

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

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

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

[N2] [Suggestion] Missing event records

Category: Others

Content

In the ListaStrategyV2 contract, the `updateTreasury` function modifies the `treasury` variable, but there is no event record.

- contracts/fund/strategy/ListaStrategyV2.sol#L97-L100

```
  function updateTreasury(address _treasury) external onlyRole(DEFAULT_ADMIN_ROLE) {
    require(_treasury != address(0), "Invalid treasury address");
    treasury = _treasury;
  }
```

Solution

It is recommended to add event logging.

Status

Fixed

[N3] [Low] Price deviation check inconsistency

Category: Design Logic Audit

Content

In the BNBPriceOracle contract, the `getPrice` function has an inconsistent price deviation check basis. When checking `minPrice`, the `minPrice` is used as the calculation basis, while when checking `maxPrice`, the `anchorPrice` is used as the basis. This asymmetric calculation leads to different sensitivities to upper and lower price deviations.

- contracts/price-oracle/BNBPriceOracle.sol#L71-L90

```
  function getPrice() public view override returns (uint256 anchorPrice, uint256
minPrice, uint256 maxPrice) {
    //...
    if ((anchorPrice - minPrice) * PRECISION > cachedMaxPriceDeviation * minPrice) {
      minPrice = anchorPrice;
    }

    // use anchor price when the price deviation between anchor price and max price
exceed threshold
    if ((maxPrice - anchorPrice) * PRECISION > cachedMaxPriceDeviation * anchorPrice)
{
      maxPrice = anchorPrice;
    }
  }
```

Solution

It is recommended to uniformly use anchorPrice as the deviation calculation benchmark to ensure symmetrical risk control.

Status

Fixed

[N4] [Suggestion] Risk of Integer overflow

Category: Integer Overflow and Underflow Vulnerability

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#L47-L52

```
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
0X002508110001	SlowMist Security Team	2025.08.11 - 2025.08.11	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, 2 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