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 2025.04.21, the SlowMist security team received the KiloEx team's security audit application for KiloEx, 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

KiloEx is a decentralized perpetual contract trading platform designed with a hierarchical architecture. It mainly provides core functions such as perpetual contract trading, hybrid asset vault management, and high-speed trading. The project adopts a modular design, and its core architecture is divided into the following layers:

1.Access Control Layer (access): Responsible for the management of governance rights, owner rights, and operational operation rights.

2.Core Trading Layer (core):

- Trading Engine: Includes market order logic (PositionRouter) and limit order logic (OrderBook).
- Storage Management: KiloStorageManager is responsible for storing key parameters and users' margin.
- Price and Fees: KiloPriceFeed provides price services, and MarginFeeManager handles funding fees.
- High-speed Trading: Through TrustedForwarder and DelegateCollection, it realizes proxy transactions with user - signed authorization.

3.Asset Management Layer (hybridvault/vaultv2):

- HybridVault supports the mixed pledging of multiple assets and provides a unified deposit and withdrawal interface.
- Integrates the APY Boost function to connect with external DeFi protocols for additional income.

4.Auxiliary Service Layer: Includes peripheral functions such as data reading interfaces, an invitation rebate system, and a token economic model.

The core is to support two trading modes: users can place market/limit orders directly or use high-speed entrusted orders based on off - chain signatures, and the automated trading executed by the keeper provides users with a flexible and efficient trading experience.

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Potential interest rate attack	Arithmetic Accuracy Deviation Vulnerability	High	Fixed
N2	Incorrect update of pending profit	Design Logic Audit	Critical	Fixed
N3	Missing trustedForwarder check on initialization	Scoping and Declarations Audit	Medium	Fixed
N4	TP/SL orders will not be cancelled when	Design Logic Audit	High	Fixed

NO	Title	Category	Level	Status
	closing a position			
N5	Potential Dos attack in token permit execution	Denial of Service Vulnerability	Low	Fixed
N6	Missing scope limit	Others	Suggestion	Acknowledged
N7	Missing the event record	Others	Suggestion	Acknowledged
N8	Missing minimum delay block check in OrderBook	Design Logic Audit	Low	Acknowledged
N9	Lack of checking of product status	Design Logic Audit	Medium	Fixed
N10	Lack of check for leverage range	Design Logic Audit	Medium	Fixed
N11	Lack of position check when creating a reduction order	Design Logic Audit	Medium	Fixed
N12	Missing order type check when updating orders	Design Logic Audit	Suggestion	Fixed
N13	Lack of updating funding fees and borrowing fees when reducing positions	Design Logic Audit	Information	Acknowledged
N14	Redundant code	Others	Suggestion	Fixed
N15	Missing event record when market order cancellation fails	Others	Suggestion	Fixed
N16	Missing check for product creation status	Design Logic Audit	Low	Fixed
N17	Lack of consistency check between old and new tokens	Design Logic Audit	Medium	Fixed
N18	Incorrect trade fee rate calculation	Design Logic Audit	Medium	Fixed

NO	Title	Category	Level	Status
N19	Missing zero address check	Others	Suggestion	Fixed
N20	Missing variable update in refill function	Design Logic Audit	Low	Acknowledged
N21	Incorrect check logic for softCapQuoteAssetsMinB	Design Logic Audit	Low	Fixed
N22	Missing non-zero check for vusd quantity calculation	Design Logic Audit	Medium	Fixed
N23	Missing hTokenId check in rebalance function	Design Logic Audit	Suggestion	Fixed
N24	Incorrect logic in collateralInQuote calculation	Design Logic Audit	Information	Acknowledged
N25	Incorrect price time recorded in priceOfChainLink function	Design Logic Audit	Medium	Fixed
N26	Missing chainID check in the signature verification	Replay Vulnerability	Medium	Fixed
N27	Missing check for the sideVault	Design Logic Audit	Medium	Fixed
N28	Missing balance check when claiming rewards	Others	Suggestion	Acknowledged
N29	Use `safeMint()` Instead of `mint()`	Design Logic Audit	Suggestion	Fixed
N30	Risk of excessive authority	Authority Control Vulnerability Audit	Medium	Fixed

4 Code Overview

4.1 Contracts Description

Audit Version:

https://github.com/KiloExContract/kilo-contracts

commit: 06455a323d0dcfbaf05c25de9df61b8183c259e9

Fixed Version:

https://github.com/KiloExContract/kilo-contracts

commit: 16e1a30ec830fc091a2f7f6706cbe9fd664660fd

The main network address of the contract is as follows:

./scripts/contract_deploy/settings/*.json

4.2 Visibility Description

The SlowMist Security team analyzed the visibility of major contracts during the audit, the result as follows:

OperatorOwnerGovernable			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
setGov	External	Can Modify State	onlyGov
acceptGov	External	Can Modify State	-
setOwner	External	Can Modify State	onlyGov
acceptOwner	External	Can Modify State	-
setOperator	External	Can Modify State	onlyOwner

OperatorOwnerGovernableUpgradeable			
Function Name	Visibility	Mutability	Modifiers
__owner_governable_init	Internal	Can Modify State	initializer
setGov	External	Can Modify State	onlyGov

OperatorOwnerGovernableUpgradeable			
acceptGov	External	Can Modify State	-
setOwner	External	Can Modify State	onlyGov
acceptOwner	External	Can Modify State	-
setOperator	External	Can Modify State	onlyOwner

OwnerGovernable			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
setGov	External	Can Modify State	onlyGov
acceptGov	External	Can Modify State	-
setOwner	External	Can Modify State	onlyGov
acceptOwner	External	Can Modify State	-

OwnerGovernableUpgradeable			
Function Name	Visibility	Mutability	Modifiers
__owner_governable_init	Internal	Can Modify State	initializer
setGov	External	Can Modify State	onlyGov
acceptGov	External	Can Modify State	-
setOwner	External	Can Modify State	onlyGov
acceptOwner	External	Can Modify State	-

Delegate			
Function Name	Visibility	Mutability	Modifiers

Delegate			
_delegate	Internal	Can Modify State	-

DelegateCollection			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC2771ContextUpgradeable
initialize	Public	Can Modify State	initializer
_msgSender	Internal	-	-
_msgData	Internal	-	-
addMarginDelegate	Public	Can Modify State	delegateAround
createIncreasePositionDelegateV3	External	Can Modify State	delegateAround
createDecreasePositionDelegateV3	External	Can Modify State	delegateAround
createIncreaseOrderDelegateV3	External	Can Modify State	delegateAround
cancelIncreaseOrderDelegate	External	Can Modify State	delegateAround
updateIncreaseOrderDelegate	External	Can Modify State	delegateAround
createDecreaseOrderDelegateV3	External	Can Modify State	delegateAround
cancelDecreaseOrderDelegate	External	Can Modify State	delegateAround
updateDecreaseOrderDelegate	External	Can Modify State	delegateAround
createIncreasePositionWithCloseTriggerOrdersDelegateV3	External	Can Modify State	delegateAround
delegateExecutePositions	External	Can Modify State	-
approveDelegate	External	Can Modify State	nonReentrant

DelegateCollection			
_sendGasAndExecutionFee	Private	Can Modify State	-

KiloPriceFeed			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
getPrice	External	-	-
shouldHaveSpread	External	-	-
shouldUpdatePrice	External	-	-
shouldUpdatePriceForToken	External	-	-
shouldUpdatePriceForTokens	External	-	-
getPrice	Public	-	-
getPriceAndSource	Public	-	-
getChainlinkPrice	Public	-	-
getChainlinkPrices	External	-	-
getPrices	External	-	-
setPrices	External	Can Modify State	onlyKeeper
enableFastOracle	External	Can Modify State	onlyOwner
setPriceDuration	External	Can Modify State	onlyOwner
setUpdatedInterval	External	Can Modify State	onlyOwner
setDefaultMaxPriceDiff	External	Can Modify State	onlyOwner
setMaxPriceDiff	External	Can Modify State	onlyOwner
setKeeper	External	Can Modify State	onlyOwner

KiloPriceFeed			
setIsChainlinkOnly	External	Can Modify State	onlyOwner
setIsKiloOracleOnly	External	Can Modify State	onlyOwner
setSpreadEnabled	External	Can Modify State	onlyOwner
setDefaultSpread	External	Can Modify State	onlyOwner
setSpread	External	Can Modify State	onlyOwner
setMaxHeartBeat	External	Can Modify State	onlyOwner

KiloStorageManager			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
initVaultV2	External	Can Modify State	reinitializer
initVaultV3	External	Can Modify State	reinitializer
initTrustedForwarder	External	Can Modify State	onlyOwner reinitializer
setMinMargin	External	Can Modify State	onlyOwner
setTradeEnabled	External	Can Modify State	onlyOwner
setCanUserStake	External	Can Modify State	onlyOwner
setMinProfitTime	External	Can Modify State	onlyOwner
setExposureMultiplier	External	Can Modify State	onlyOwner
setUtilizationMultiplier	External	Can Modify State	onlyOwner
setMaxExposureMultiplier	External	Can Modify State	onlyOwner
setMaxShift	External	Can Modify State	onlyOwner
setLiquidationParams	External	Can Modify State	onlyOwner

KiloStorageManager			
setAllowPublicLiquidator	External	Can Modify State	onlyOwner
setParameters	External	Can Modify State	onlyOwner
pauseTrading	External	Can Modify State	onlyOwner
getKiloConfig	External	-	-
setAdlMultiplier	External	Can Modify State	onlyOwner
setToken	External	Can Modify State	onlyOwner
setMarginFeeManagerAddr	External	Can Modify State	onlyOwner
setPerpTradeAddr	External	Can Modify State	onlyOwner
setPendingRewardAddr	External	Can Modify State	onlyOwner
setKiloPriceFeedAddr	External	Can Modify State	onlyOwner
setOrderBookAddr	External	Can Modify State	onlyOwner
setPositionRouterAddr	External	Can Modify State	onlyOwner
setStakeRewardAddr	External	Can Modify State	onlyOwner
setProductManagerAddr	External	Can Modify State	onlyOwner
storeIncreasePosition	External	Can Modify State	onlyPerpTrade
updatePositionMargin	External	Can Modify State	onlyPerpTrade
addMargin	External	Can Modify State	onlyPerpTrade
clearPosition	External	Can Modify State	onlyPerpTrade
updateIncreaseOpenInterest	External	Can Modify State	onlyPerpTrade
updateDecreaseOpenInterest	External	Can Modify State	onlyPerpTrade
initTotalLongShortOI	External	Can Modify State	onlyOwner
transferPendingReward	External	Can Modify State	-

KiloStorageManager			
transferRemainMargin	External	Can Modify State	onlyPerpTrade
actionWithVaultForFee	External	Can Modify State	-
actionWithVaultForPnl	External	Can Modify State	-
setDelegate	External	Can Modify State	-
approveDelegate	Public	Can Modify State	-
fixDelegateData	External	Can Modify State	-
getPositionId	Public	-	-
getPosition	Public	-	-
getPositionById	Public	-	-
getPositionLeverage	External	-	-
openInterest	External	-	-

MarginFeeManager			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
updateMarginFee	External	Can Modify State	-
_updateFunding	Internal	Can Modify State	-
getBorrowingRate	Public	-	-
getBorrowing	External	-	-
getCurrentCumulativeBorrowing	External	-	-
getFundingRate	Public	-	-
getFunding	External	-	-

MarginFeeManager			
getCurrentCumulativeFunding	External	-	-
setPerpTrade	External	Can Modify State	onlyOwner
setOperator	External	Can Modify State	onlyOwner
setMaxBorrowingRate	External	Can Modify State	onlyOwner
setMinBorrowingRate	External	Can Modify State	onlyOwner
setMinFundingMultiplier	External	Can Modify State	onlyOwner
setFundingMultiplier	External	Can Modify State	onlyOwner
batchSetFundingMultiplier	External	Can Modify State	-
batchSetFundingRate	External	Can Modify State	-
setMaxFundingRate	External	Can Modify State	onlyOwner
getFundingBorrowing	External	-	-

MarketOrderWithTriggerOrder			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
setOrderBook	External	Can Modify State	onlyOwner
setPositionRouter	External	Can Modify State	onlyOwner
setKiloStorageAddr	External	Can Modify State	onlyOwner
createIncreasePositionWithCloseTriggerOrders Delegate	Public	Payable	-
createIncreasePositionWithCloseTriggerOrders	External	Payable	-

MarketOrderWithTriggerOrder			
createIncreasePositionWithCloseTriggerOrdersDelegateV3	Public	Payable	delegate
createIncreasePositionWithCloseTriggerOrdersV3	External	Payable	-
minExecutionFees	External	-	-

OrderBook			
Function Name	Visibility	Mutability	Modifiers
_onlyKeeper	Internal	-	-
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
initTrustedForwarder	External	Can Modify State	onlyOwner reinitializer
setApprovedRouter	External	Can Modify State	onlyOwner
setMinExecutionFee	External	Can Modify State	onlyOwner
setKeeper	External	Can Modify State	onlyOwner
setMaxOrderSize	External	Can Modify State	onlyOwner
executeOrdersWithPrices	External	Can Modify State	onlyKeeper
executeOrders	Public	Can Modify State	onlyKeeper
cancelMultiple	External	Can Modify State	-
cancelDecreaseOrderMultiple	Public	Can Modify State	-
validatePositionOrderPrice	Public	-	-
getDecreaseOrder	Public	-	-

OrderBook			
getIncreaseOrder	Public	-	-
createIncreaseOrder	External	Payable	-
createIncreaseOrderV3	External	Payable	-
__isDelegateRequest	Private	-	-
__executionQuoteFee	Private	-	-
createIncreaseOrderDelegateV3	Public	Payable	delegate nonReentrant
_createIncreaseOrder	Private	Can Modify State	-
updateIncreaseOrder	External	Can Modify State	-
updateIncreaseOrderDelegate	Public	Can Modify State	delegate nonReentrant
cancelIncreaseOrder	External	Can Modify State	-
cancelIncreaseOrderDelegate	Public	Can Modify State	delegate nonReentrant
executeIncreaseOrder	Public	Can Modify State	nonReentrant
createDecreaseOrderDelegateV3	Public	Payable	delegate nonReentrant
createDecreaseOrder	External	Payable	nonReentrant
createDecreaseOrderV3	External	Payable	nonReentrant
createDecreaseOrderWithAccount	External	Payable	nonReentrant
_createDecreaseOrder	Private	Can Modify State	-
executeDecreaseOrder	Public	Can Modify State	nonReentrant
cancelNoPositionDecreaseOrderWithAccount	External	Can Modify State	-
cancelDecreaseOrder	External	Can Modify State	-

OrderBook			
cancelDecreaseOrderDelegate	Public	Can Modify State	delegate nonReentrant
_cancelDecreaseOrder	Private	Can Modify State	-
updateDecreaseOrder	External	Can Modify State	-
updateDecreaseOrderDelegate	Public	Can Modify State	delegate nonReentrant
_transferOutETH	Private	Can Modify State	-
_transferOutExecutionFee	Internal	Can Modify State	-
getTradeFeeRate	Private	-	-
_getOrderType	Private	-	-
<Receive Ether>	External	Payable	-

PendingReward			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
setPerpTrade	External	Can Modify State	onlyOwner
setProtocolRewardRatio	External	Can Modify State	onlyOwner
setRewardReceivers	External	Can Modify State	onlyOwner
updatePendingRewards	Public	Can Modify State	onlyPerpTrade
updateLiquidationReward	External	Can Modify State	onlyPerpTrade
withdrawProtocolReward	External	Can Modify State	nonReentrant
withdrawLiquidationReward	External	Can Modify State	nonReentrant
withdrawVaultReward	External	Can Modify State	nonReentrant

PendingReward			
updatePendingPnlAndGetFee	External	Can Modify State	onlyPerpTrade
decrPendingPnl	Public	Can Modify State	onlyPerpTrade
incrPendingPnl	Public	Can Modify State	onlyPerpTrade
distributePendingPnl	Public	Can Modify State	nonReentrant
getPendingTransferring	External	-	-
getVaultPendingBalance	External	-	-

PerpTrade			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
setVaultStakeReward	External	Can Modify State	onlyOwner
initializePriceImpactLogic	External	Can Modify State	onlyOwner reinitializer
increasePosition	Public	Can Modify State	nonReentrant
addMargin	External	Can Modify State	-
_addMargin	Internal	Can Modify State	-
addMarginDelegate	Public	Can Modify State	delegate nonReentrant
decreasePosition	External	Can Modify State	nonReentrant
adlDecreasePosition	External	Can Modify State	nonReentrant
decreasePositionWithId	Internal	Can Modify State	-
liquidatePositionsWithPrices	External	Can Modify State	-
_liquidatePosition	Private	Can Modify State	-
_getVaultBalance	Private	-	-

PerpTrade			
_validateRouter	Private	-	-
getPositionId	Public	-	-
getMaxExposure	Public	-	-
_getMaxExposure	Private	-	-
_getMaxExposure2	Private	-	-
setApprovedRouter	External	Can Modify State	onlyOwner
setOracle	External	Can Modify State	onlyOwner
setMarginFeeManager	External	Can Modify State	onlyOwner
setLiquidator	External	Can Modify State	onlyOwner

PositionRouter			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
initOrderBook	External	Can Modify State	onlyOwner reinitializer
initTrustedForwarder	External	Can Modify State	onlyOwner reinitializer
setApprovedRouter	External	Can Modify State	onlyOwner
setReferralStorage	External	Can Modify State	onlyOwner
setPositionKeeper	External	Can Modify State	onlyOwner
setMinExecutionFee	External	Can Modify State	onlyOwner
setIsUserExecuteEnabled	External	Can Modify State	onlyOwner
setIsUserCancelEnabled	External	Can Modify State	onlyOwner
setDelayValues	External	Can Modify State	onlyOwner

PositionRouter			
executePositionsWithPricesT	External	Can Modify State	onlyPositionKeeper
createIncreasePositionWithAccount	Public	Payable	nonReentrant
createIncreasePosition	External	Payable	-
createIncreasePositionV3	External	Payable	-
createIncreasePositionDelegateV3	Public	Payable	delegate nonReentrant
__isDelegateRequest	Private	-	-
__executionQuoteFee	Private	-	-
__createIncreasePosition	Internal	Can Modify State	-
_createIncreasePosition	Internal	Can Modify State	-
createDecreasePosition	External	Payable	-
createDecreasePositionV3	External	Payable	-
createDecreasePositionDelegateV3	Public	Payable	delegate nonReentrant
_createDecreasePosition	Internal	Can Modify State	-
executeIncreasePosition	Public	Can Modify State	nonReentrant
_transferOutExecutionFee	Private	Can Modify State	-
cancelIncreasePosition	Public	Can Modify State	nonReentrant
executeDecreasePosition	Public	Can Modify State	nonReentrant
cancelDecreasePosition	Public	Can Modify State	nonReentrant
transferQuoteGasFee	External	Can Modify State	-
getRequestKey	Public	-	-
getIncreasePositionRequest	Public	-	-
getDecreasePositionRequest	Public	-	-

PositionRouter			
getIncreasePositionRequestFromKey	Public	-	-
getDecreasePositionRequestFromKey	Public	-	-
_validateExecution	Internal	-	-
_validateCancellation	Internal	-	-
<Fallback>	External	Payable	-
<Receive Ether>	External	Payable	-

PriceImpactLogic			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
_getTradePriceImpact	Internal	-	-
tradeOpeningPriceImpact	External	Can Modify State	-
getTradeOpeningPriceImpact	Public	-	-
tradeClosingPriceImpact	Public	Can Modify State	-
getTradeClosingPriceImpact	Public	-	-
_getFundingPayment	Internal	-	-
batchSetPairFactor	External	Can Modify State	onlyOperator
batchSetMinSpread	External	Can Modify State	onlyOperator
setProtectionCloseFactorWhitelist	External	Can Modify State	onlyOperator
getPairFactor	Public	-	-
minSpreads	External	-	-

ProductManager			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
addProduct	External	Can Modify State	onlyOwner
updateProduct	External	Can Modify State	onlyOwner
setBorrowingFactor	Public	Can Modify State	onlyOwner
batchSetBorrowingFactor	External	Can Modify State	-
setMaxPositionSize	External	Can Modify State	onlyOwner
setMaxShift	External	Can Modify State	onlyOwner
setFee	External	Can Modify State	onlyOwner
setLeverage	External	Can Modify State	onlyOwner
setIsActive	External	Can Modify State	onlyOwner
setMinPriceChange	External	Can Modify State	onlyOwner
setWeight	External	Can Modify State	onlyOwner
setToken	External	Can Modify State	onlyOwner
setReserve	External	Can Modify State	onlyOwner
getProduct	External	-	-
getTotalWeight	External	-	-
getTradeFeeRateV2	External	-	-
enableFeeDiscount	External	Can Modify State	onlyOwner
setAccountDiscount	External	Can Modify State	onlyOwner
batchSetAccountDiscount	External	Can Modify State	-

ProductManager			
batchSetMinSpread	External	Can Modify State	-
batchSetReserve	External	Can Modify State	-
setOperator	External	Can Modify State	onlyOwner

TrustedForwarder			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
<Receive Ether>	External	Payable	-
setApprovedRouter	External	Can Modify State	onlyOwner
updateQuoteGasFee	External	Can Modify State	onlyOwner
updateMaxQuoteGasFee	External	Can Modify State	onlyOwner
updateExecutionQuoteFee	External	Can Modify State	onlyOwner
setKeeper	External	Can Modify State	onlyOwner
forwardExecute	Public	Payable	nonReentrant
execute	Public	Payable	-
forwardBundle	Public	Payable	nonReentrant
forwardBundleCheck	External	Payable	-
withdrawQuoteToken	External	Can Modify State	onlyOwner
getUserForwarderData	External	-	-
nonces	External	-	-

VaultStakeReward			
Function Name	Visibility	Mutability	Modifiers

VaultStakeReward			
_checkHybridVault	Internal	-	-
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
initializeV2	External	Can Modify State	onlyOwner reinitializer
initializeV3	External	Can Modify State	onlyOwner reinitializer
getStake	External	-	-
setMIN_LOCK_DURATION	External	Can Modify State	onlyOwner
updatePnlHandler	External	Can Modify State	onlyOwner
updateOpenTradesPnlFeed	External	Can Modify State	onlyOwner
updateMaxAccOpenPnlDelta	External	Can Modify State	onlyOwner
updateMaxDailyAccPnlDelta	External	Can Modify State	onlyOwner
updateWithdrawLockThresholdsP	External	Can Modify State	onlyOwner
updateMaxSupplyIncreaseDailyP	External	Can Modify State	onlyOwner
updateLossesBurnP	External	Can Modify State	onlyOwner
updateMaxDiscountP	External	Can Modify State	onlyOwner
updateMaxDiscountThresholdP	External	Can Modify State	onlyOwner
maxAccPnlPerToken	Public	-	-
collateralizationP	Public	-	-
withdrawEpochsTimelock	Public	-	-

VaultStakeReward			
lockDiscountP	Public	-	-
totalSharesBeingWithdrawn	Public	-	-
tryUpdateCurrentMaxSupply	Public	Can Modify State	-
tryResetDailyAccPnlDelta	Public	Can Modify State	-
updateShareToAssetsPrice	Private	Can Modify State	-
_assetIERC20	Private	-	-
transfer	Public	Can Modify State	-
transferFrom	Public	Can Modify State	-
decimals	Public	-	-
_convertToShares	Internal	-	-
_convertToAssets	Internal	-	-
maxMint	Public	-	-
maxDeposit	Public	-	-
maxRedeem	Public	-	-
maxWithdraw	Public	-	-
deposit	Public	Can Modify State	onlyHybridVault checks
depositForReBalance	External	Can Modify State	onlyHybridVault checks
mint	Public	Can Modify State	onlyHybridVault checks
withdraw	Public	Can Modify State	checks
redeem	Public	Can Modify State	onlyHybridVault checks

VaultStakeReward			
redeemForReBalance	External	Can Modify State	onlyHybridVault checks
scaleVariables	Internal	Can Modify State	-
makeWithdrawRequest	External	Can Modify State	onlyHybridVault
getUnlockEpoch	Public	-	-
cancelWithdrawRequest	External	Can Modify State	onlyHybridVault
depositWithDiscountAndLock	External	Can Modify State	checks validDiscount onlyHybridVault
mintWithDiscountAndLock	External	Can Modify State	checks validDiscount
_executeDiscountAndLock	Private	Can Modify State	-
unlockDeposit	External	Can Modify State	onlyHybridVault
distributeReward	External	Can Modify State	onlyHybridVault
_sendAssets	Internal	Can Modify State	-
sendAssets	External	Can Modify State	onlyHybridVault
receiveAssets	External	Can Modify State	onlyHybridVault
deplete	External	Can Modify State	-
withdrawDeplete	External	Can Modify State	onlyHybridVault
refill	External	Can Modify State	onlyHybridVault
updateAccPnlPerTokenUsed	External	Can Modify State	-
getLockedDeposit	External	-	-
tvl	Public	-	-

VaultStakeReward			
availableAssets	Public	-	-
marketCap	Public	-	-
reBalanceForShares	External	Can Modify State	onlyHybridVault
reBalanceForWithdrawRequest	External	Can Modify State	onlyHybridVault
badDebtHandle	External	Can Modify State	onlyHybridVault

VaultStakeRewardTemp			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
initializeV2	External	Can Modify State	onlyOwner reinitializer
getStake	External	-	-
setMIN_LOCK_DURATION	External	Can Modify State	onlyOwner
updatePnlHandler	External	Can Modify State	onlyOwner
updateOpenTradesPnlFeed	External	Can Modify State	onlyOwner
updateMaxAccOpenPnlDelta	External	Can Modify State	onlyOwner
updateMaxDailyAccPnlDelta	External	Can Modify State	onlyOwner
updateWithdrawLockThresholdsP	External	Can Modify State	onlyOwner
updateMaxSupplyIncreaseDailyP	External	Can Modify State	onlyOwner
updateLossesBurnP	External	Can Modify State	onlyOwner
updateMaxDiscountP	External	Can Modify State	onlyOwner
updateMaxDiscountThresholdP	External	Can Modify State	onlyOwner

VaultStakeRewardTemp			
maxAccPnlPerToken	Public	-	-
collateralizationP	Public	-	-
withdrawEpochsTimelock	Public	-	-
lockDiscountP	Public	-	-
totalSharesBeingWithdrawn	Public	-	-
tryUpdateCurrentMaxSupply	Public	Can Modify State	-
tryResetDailyAccPnlDelta	Public	Can Modify State	-
updateShareToAssetsPrice	Private	Can Modify State	-
_assetIERC20	Private	-	-
transfer	Public	Can Modify State	isActive
transferFrom	Public	Can Modify State	isActive
decimals	Public	-	-
_convertToShares	Internal	-	-
_convertToAssets	Internal	-	-
maxMint	Public	-	-
maxDeposit	Public	-	-
maxRedeem	Public	-	-
maxWithdraw	Public	-	-
deposit	Public	Can Modify State	checks isActive
mint	Public	Can Modify State	checks isActive
withdraw	Public	Can Modify State	checks isActive
redeem	Public	Can Modify State	checks isActive

VaultStakeRewardTemp			
scaleVariables	Private	Can Modify State	-
makeWithdrawRequest	External	Can Modify State	isActive
cancelWithdrawRequest	External	Can Modify State	isActive
depositWithDiscountAndLock	External	Can Modify State	checks validDiscount isActive
mintWithDiscountAndLock	External	Can Modify State	checks validDiscount
_executeDiscountAndLock	Private	Can Modify State	-
unlockDeposit	External	Can Modify State	isActive
distributeReward	External	Can Modify State	isInitialize
sendAssets	External	Can Modify State	isInitialize
receiveAssets	External	Can Modify State	isInitialize
deplete	External	Can Modify State	-
refill	External	Can Modify State	-
updateAccPnlPerTokenUsed	External	Can Modify State	-
getLockedDeposit	External	-	-
tvl	Public	-	-
availableAssets	Public	-	-
marketCap	Public	-	-

MinimalForwarderUpgradeable			
Function Name	Visibility	Mutability	Modifiers
__MinimalForwarder_init	Internal	Can Modify State	onlyInitializing
__MinimalForwarder_init_unchained	Internal	Can Modify State	onlyInitializing
getNonce	Public	-	-

MinimalForwarderUpgradeable			
verify	Public	-	-
execute	Public	Payable	-

HToken			
Function Name	Visibility	Mutability	Modifiers
_checkHybridVault	Internal	-	-
initialize	Public	Can Modify State	initializer
decimals	Public	-	-
transfer	Public	Can Modify State	-
transferFrom	Public	Can Modify State	-
burn	External	Can Modify State	onlyHybridVault
mint	External	Can Modify State	onlyHybridVault

HybridVault			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
configHToken	Public	Can Modify State	onlyOwner
setLiquidationBonus	External	Can Modify State	onlyOwner
setCanUserStake	External	Can Modify State	onlyOwner
setQuoteAssetsMinBp	External	Can Modify State	onlyOwner
manageHTokenLtv	External	Can Modify State	onlyOperator
deposit	External	Payable	isActive

HybridVault			
totalSharesBeingWithdrawn	Public	-	-
makeWithdrawRequest	External	Can Modify State	isActive
cancelWithdrawRequest	External	Can Modify State	isActive
redeem	External	Can Modify State	isActive
depositWithDiscountAndLock	External	Can Modify State	isActive
unlockDeposit	External	Can Modify State	isActive
distributeReward	External	Can Modify State	-
sendAssets	External	Can Modify State	-
receiveAssets	External	Can Modify State	-
refill	External	Can Modify State	-
withdrawDeplete	External	Can Modify State	onlyOwner
claimSettlePnl	External	Can Modify State	-
reBalance	Public	Can Modify State	onlyOperator
_reBalanceWithdrawRequestShares	Internal	Can Modify State	-
_calculateUserAccountData	Internal	Can Modify State	-
liquidate	External	Can Modify State	onlyOperator
payDebt	External	Can Modify State	-
getHTokenIds	External	-	-
setSideVaultEntry	External	Can Modify State	onlyOwner
boost	External	Can Modify State	-

HybridVaultLogic			
Function Name	Visibility	Mutability	Modifiers

HybridVaultLogic			
calculateUserAccountData	Internal	-	-
calculateDebt	Internal	-	-
calculateAvailableCollateralToLiquidate	Internal	-	-
checkQuoteAssetsMinBp	Internal	-	-

HybridVaultReader			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
calculateUserAccountData	Public	-	-
accountDepositData	Public	-	-
listAccountDepositData	External	-	-
accountHVaultValues	Public	-	-
accountDepositValues	External	-	-
lockedDeposits	Public	-	-
calculateLtvChangeFactor	External	-	-

KiloERC4626Upgradeable			
Function Name	Visibility	Mutability	Modifiers
__ERC4626_init	Internal	Can Modify State	onlyInitializing
__ERC4626_init_unchained	Internal	Can Modify State	-
asset	Public	-	-
totalAssets	Public	-	-

KiloERC4626Upgradeable			
convertToShares	Public	-	-
convertToAssets	Public	-	-
maxDeposit	Public	-	-
maxMint	Public	-	-
maxWithdraw	Public	-	-
maxRedeem	Public	-	-
previewDeposit	Public	-	-
previewMint	Public	-	-
previewWithdraw	Public	-	-
previewRedeem	Public	-	-
deposit	Public	Can Modify State	-
mint	Public	Can Modify State	-
withdraw	Public	Can Modify State	-
redeem	Public	Can Modify State	-
_convertToShares	Internal	-	-
_convertToAssets	Internal	-	-
_deposit	Internal	Can Modify State	-
_withdraw	Internal	Can Modify State	-
_isVaultCollateralized	Private	-	-

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

PriceRouter			
initialize	Public	Can Modify State	initializer
setChainLinkOracleSource	External	Can Modify State	onlyOperator
setPythOracleSource	External	Can Modify State	onlyOperator
setKiloSignatureOracleSource	External	Can Modify State	onlyOperator
setSignerAddress	External	Can Modify State	onlyOwner
setMaxOldAge	External	Can Modify State	onlyOwner
batchSetPrices	External	Payable	-
priceOfUnderlying	Public	Payable	-
priceOfPyth	Internal	Can Modify State	-
priceOfChainLink	Internal	Can Modify State	-
priceOfSupraOracle	Internal	Can Modify State	-
priceOfKiloEx	Public	Can Modify State	-
priceOfSignature	Public	Can Modify State	-
priceOfMock	Internal	Can Modify State	-
getPriceNoOlderThan	External	-	-
getMessageHash	Public	-	-
toEthSignedMessageHash	Public	-	-
verify	Public	-	-
<Receive Ether>	External	Payable	-

PriceRouterHelper			
Function Name	Visibility	Mutability	Modifiers
encodePriceDataWithSignature	External	-	-

VUSD			
Function Name	Visibility	Mutability	Modifiers
_checkHybridVault	Internal	-	-
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
setHybridVault	External	Can Modify State	onlyOwner
decimals	Public	-	-
burn	External	Can Modify State	onlyHybridVault
mint	External	Can Modify State	onlyHybridVault

GenesisPassCard			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
claim	External	Can Modify State	-
_beforeTokenTransfer	Internal	Can Modify State	-
setContractURI	External	Can Modify State	onlyOwner
setSignerAddress	External	Can Modify State	onlyOwner
setBaseURI	External	Can Modify State	onlyOwner
tokensOfOwner	External	-	-
tokenURI	Public	-	-
contractURI	Public	-	-
getMessageHash	Internal	-	-
toEthSignedMessageHash	Internal	-	-

GenesisPassCard			
verify	Internal	-	-
toString	Internal	-	-

KiloPassCard			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
whitelistMint	External	Can Modify State	callerIsUser
setBaseURI	External	Can Modify State	onlyOperator
setSaleStartTime	External	Can Modify State	onlyOperator
setStatus	External	Can Modify State	onlyOperator
tokenURI	Public	-	-
currentTime	Internal	-	-
setMerkleRoot	External	Can Modify State	onlyOperator
isWhiteListed	Public	-	-
leaf	Internal	-	-
_verify	Internal	-	-
toString	Internal	-	-

CheckIn			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer

CheckIn			
checkIn	External	Can Modify State	-

CommonReward			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
claimReward	External	Can Modify State	nonReentrant
updateRewards	External	Can Modify State	-
updateReward	Public	Can Modify State	onlyOperator
withdrawEmergency	External	Can Modify State	onlyOwner
getRewardsInfo	External	-	-

KeeperReader			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
getProductOI	External	-	-

KiloExReader			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
setTrustForwarder	External	Can Modify State	-
getProductsV2	Public	-	-
productSummaryExt	Public	-	-

KiloExReader			
productSummaryExt2	Public	-	-
getFundingPayment	Public	-	-
getPositionProfits	Public	-	-

KiloPerpView			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
initializeVaultV2	External	Can Modify State	reinitializer
initializePriceImpactLogic	External	Can Modify State	reinitializer
getTradeFee	External	-	-
getProductsV2	Public	-	-
getPositions	Public	-	-
getPositionWithPending	External	-	-
getCurrentCumulativeFunding	External	-	-
getCurrentCumulativeBorrowing	External	-	-
getFundingBorrowings	Public	-	-
productSummary	Public	-	-
productSummaryExt	External	-	-
getParameters	External	-	-
canTakeProfit	External	-	-
getMaxOpens	Public	-	-
getPendingPositions	Public	-	-

KiloPerpView			
vaultOf	External	-	-
vaultOwnerSummary	External	-	-
vaultSummary	External	-	-
getOpenPnlNextRequestTime	External	-	-
getOpenPnlNextRequest	External	-	-
positionMargins	External	-	-
batchGetPositions	External	-	-

KolRewardDistributor			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
claimReward	External	Can Modify State	nonReentrant
updateKolRewards	External	Can Modify State	-
updateKolReward	Public	Can Modify State	onlyOperator
withdrawEmergency	External	Can Modify State	onlyOwner
getRewardsInfo	External	-	-

LiquidationPriceReader			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
_getFundingPayment	Internal	-	-
_getBorrowingPayment	Internal	-	-

LiquidationPriceReader			
getLiquidationPrice	External	-	-
getLiquidationPrices	External	-	-
getPrices	External	-	-

PerpTradeReader			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
getPositions	Public	-	-
getFundings	Public	-	-
calculateSpread	External	-	-
productSummary	Public	-	-
getMaxOpens	Public	-	-
balanceOf	External	-	-
accountBalances	External	-	-

TeamContestReward			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
claimReward	External	Can Modify State	nonReentrant
updateTeamContestRewards	External	Can Modify State	-
updateTeamContestReward	Public	Can Modify State	onlyOperator

TeamContestReward			
withdrawExpiredReward	External	Can Modify State	onlyOwner
getRewardsInfo	External	-	-

V2PlusTemp			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
batchGets	External	-	-

ReferralReader			
Function Name	Visibility	Mutability	Modifiers
getCodeOwners	Public	-	-

ReferralStorageManager			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
setHandler	External	Can Modify State	onlyGov
setTier	External	Can Modify State	onlyOwner
setDiscountShareLimiter	External	Can Modify State	onlyOwner
setReferrerTier	External	Can Modify State	onlyOperator
getReferrerTier	Public	-	-
setReferrerDiscountShare	External	Can Modify State	-
setTraderReferralCode	External	Can Modify State	onlyHandler
setTraderReferralCodeByUser	External	Can Modify State	-

ReferralStorageManager			
govSetCodeOwner	External	Can Modify State	onlyOwner
registerCode	External	Can Modify State	-
getTraderReferralInfo	External	-	-
_setTraderReferralCode	Private	Can Modify State	-

ListaDaoWbnbStrategy			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
deposit	Public	Can Modify State	onlySideVault
withdraw	External	Can Modify State	onlySideVault
payDebt	Public	Can Modify State	-
toWBNB	Public	Can Modify State	-
setLaunchStartTime	External	Can Modify State	onlyOwner
setSideVault	External	Can Modify State	onlyOwner
beforeDeposit	External	Can Modify State	-
want	Public	-	-
balanceOf	Public	-	-
balanceOfWant	Public	-	-
balanceOfPool	Public	-	-
balanceOfPendingWithdrawal	Public	-	-
balSummary	External	-	-
<Receive Ether>	External	Payable	-

SideVaultWithPending			
Function Name	Visibility	Mutability	Modifiers
_checkHybridVault	Internal	-	-
_checkSideVaultEntry	Internal	-	-
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
want	Public	-	-
balance	Public	-	-
available	Public	-	-
boost	Public	Can Modify State	nonReentrant onlySideVaultEntry
unboost	Public	Can Modify State	onlySideVaultEntry
claimUnBoosts	Public	Can Modify State	onlySideVaultEntry
assetValuesOf	External	-	-
_beforeTokenTransfer	Internal	Can Modify State	-
<Receive Ether>	External	Payable	-

AaveV3Strategy			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
deposit	Public	Can Modify State	onlySideVault
beforeWithdraw	External	Can Modify State	-
withdraw	External	Can Modify State	onlySideVault
setSideVault	External	Can Modify State	onlyOwner

AaveV3Strategy			
beforeDeposit	External	Can Modify State	-
want	Public	-	-
balanceOf	Public	-	-
balanceOfWant	Public	-	-
balanceOfPool	Public	-	-
price	Public	-	-
pause	Public	Can Modify State	onlyOwner
unpause	External	Can Modify State	onlyOwner
debts	External	-	-

SideVault			
Function Name	Visibility	Mutability	Modifiers
_checkHybridVault	Internal	-	-
_checkSideVaultEntry	Internal	-	-
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
want	Public	-	-
balance	Public	-	-
available	Public	-	-
boost	Public	Can Modify State	nonReentrant onlySideVaultEntry
unboost	Public	Can Modify State	onlySideVaultEntry
claimUnBoosts	Public	Can Modify State	onlySideVaultEntry
assetValuesOf	External	-	-

SideVault			
_beforeTokenTransfer	Internal	Can Modify State	-

SideVaultEntry			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
addSideVault	Public	Can Modify State	onlyOwner
removeSideVault	Public	Can Modify State	onlyOwner
boost	External	Can Modify State	onlyHybridVault
maxBoost	Public	-	-
unboost	External	Can Modify State	-
claimUnBoost	External	Can Modify State	-
getDepositedTokens	Public	-	-
deposits	External	-	-
getDayIdx	External	-	-

VenusVTokenStrategy			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
deposit	Public	Can Modify State	onlySideVault
beforeWithdraw	External	Can Modify State	-
withdraw	External	Can Modify State	onlySideVault

VenusVTokenStrategy			
setSideVault	External	Can Modify State	onlyOwner
beforeDeposit	External	Can Modify State	-
want	Public	-	-
balanceOf	Public	-	-
balanceOfWant	Public	-	-
balanceOfPool	Public	-	-
pause	Public	Can Modify State	onlyOwner
unpause	External	Can Modify State	onlyOwner
debts	External	-	-

AirdropRewardDistributor			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
claim	External	Can Modify State	whenNotPaused nonReentrant
updateTradingRewards	External	Can Modify State	onlyOperator
pauseDistribution	External	Can Modify State	onlyGov whenNotPaused
unpauseDistribution	External	Can Modify State	onlyGov whenPaused
withdrawKiloTokenRewards	External	Can Modify State	onlyGov whenPaused
pendingRewards	External	-	-
_pendingRewards	Internal	-	-

KiloVestingWallet			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer

StakingReader			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
summaryOf	External	-	-
getUserRedeems	Public	-	-
dividendTokens	Public	-	-

XKiloToken			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
getXKiloBalance	External	-	-
getKiloByVestingDuration	Public	-	-
getUserRedeemsLength	External	-	-
getEndedUserRedeemsLength	External	-	-
getUserRedeems	External	-	-
getUsageApproval	External	-	-
getUsageAllocation	External	-	-

XKiloToken			
transferWhitelistLength	External	-	-
transferWhitelist	External	-	-
isTransferWhitelisted	External	-	-
updateRedeemSettings	External	Can Modify State	onlyOwner
updateDividendsAddress	External	Can Modify State	onlyOwner
updateDeallocationFee	External	Can Modify State	onlyOwner
updateTransferWhitelist	External	Can Modify State	onlyOwner
approveUsage	External	Can Modify State	nonReentrant
convert	External	Can Modify State	nonReentrant
convertTo	External	Can Modify State	nonReentrant
redeem	External	Can Modify State	nonReentrant
finalizeRedeem	External	Can Modify State	nonReentrant validateRedeem
updateRedeemDividendsAddress	External	Can Modify State	nonReentrant validateRedeem
cancelRedeem	External	Can Modify State	nonReentrant validateRedeem
allocate	External	Can Modify State	nonReentrant
allocateFromUsage	External	Can Modify State	nonReentrant
deallocate	External	Can Modify State	nonReentrant
deallocateFromUsage	External	Can Modify State	nonReentrant
_convert	Internal	Can Modify State	-
_finalizeRedeem	Internal	Can Modify State	-
_allocate	Internal	Can Modify State	-
_deallocate	Internal	Can Modify State	-

XKiloToken			
_deleteRedeemEntry	Internal	Can Modify State	-
_beforeTokenTransfer	Internal	-	-
_currentBlockTimestamp	Internal	-	-

XKiloDividends			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
cycleDurationSeconds	External	-	-
distributedTokensLength	External	-	-
distributedToken	External	-	validateDistributedTokensIndex
isDistributedToken	External	-	-
nextCycleStartTime	Public	-	-
getDividendsInfo	External	-	-
pendingDividendsAmount	External	-	-
updateCurrentCycleStartTime	Public	Can Modify State	-
updateDividendsInfo	External	Can Modify State	validateDistributedToken
massUpdateDividendsInfo	External	Can Modify State	-
harvestDividends	External	Can Modify State	nonReentrant
harvestAllDividends	External	Can Modify State	nonReentrant
addDividendsToPending	External	Can Modify State	nonReentrant

XKiloDividends			
emergencyWithdraw	Public	Can Modify State	nonReentrant onlyOwner
emergencyWithdrawAll	External	Can Modify State	nonReentrant onlyOwner
allocate	External	Can Modify State	nonReentrant xKiloTokenOnly
deallocate	External	Can Modify State	nonReentrant xKiloTokenOnly
enableDistributedToken	External	Can Modify State	onlyOwner
disableDistributedToken	External	Can Modify State	onlyOwner
updateCycleDividendsPercent	External	Can Modify State	onlyOwner
removeTokenFromDistributedTokens	External	Can Modify State	onlyOwner
_dividendsAmountPerSecond	Internal	-	-
_updateDividendsInfo	Internal	Can Modify State	-
_updateUser	Internal	Can Modify State	-
_harvestDividends	Internal	Can Modify State	-
_safeTokenTransfer	Internal	Can Modify State	-
_currentBlockTimestamp	Internal	-	-

AffiliateRewardDistributor			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
setProtocolReward	External	Can Modify State	onlyOwner

AffiliateRewardDistributor			
claim	External	Can Modify State	whenNotPaused nonReentrant
updateTradingRewards	External	Can Modify State	onlyOperator
pauseDistribution	External	Can Modify State	onlyGov whenNotPaused
unpauseDistribution	External	Can Modify State	onlyGov whenPaused
withdrawTokenRewards	External	Can Modify State	onlyGov whenPaused
pendingRewards	External	-	-
_pendingRewards	Internal	-	-

PendingReward			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
setPerpTrade	External	Can Modify State	onlyOwner
setProtocolRewardRatio	External	Can Modify State	onlyOwner
setRewardReceivers	External	Can Modify State	onlyOwner
updatePendingRewards	Public	Can Modify State	onlyPerpTrade
updateLiquidationReward	External	Can Modify State	onlyPerpTrade
withdrawProtocolReward	External	Can Modify State	nonReentrant
withdrawLiquidationReward	External	Can Modify State	nonReentrant
withdrawVaultReward	External	Can Modify State	nonReentrant
updatePendingPnlAndGetFee	External	Can Modify State	onlyPerpTrade
decrPendingPnl	Public	Can Modify State	onlyPerpTrade
incrPendingPnl	Public	Can Modify State	onlyPerpTrade

PendingReward			
distributePendingPnl	Public	Can Modify State	nonReentrant
getPendingTransferring	External	-	-
getVaultPendingBalance	External	-	-

TradeRewardDistributor			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
setProtocolReward	External	Can Modify State	onlyOwner
setXkiloToken	External	Can Modify State	onlyOwner
claim	External	Can Modify State	whenNotPaused nonReentrant
updateTradingRewards	External	Can Modify State	onlyOperator
pauseDistribution	External	Can Modify State	onlyGov whenNotPaused
unpauseDistribution	External	Can Modify State	onlyGov whenPaused
withdrawTokenRewards	External	Can Modify State	onlyGov whenPaused
withdrawXkiloRewards	External	Can Modify State	onlyGov whenPaused
pendingRewards	External	-	-
_pendingRewards	Internal	-	-

KTokenLockedDepositNft			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC721
updateDesign	External	Can Modify State	onlyKTokenManager

KTokenLockedDepositNft			
mint	External	Can Modify State	onlyKToken
burn	External	Can Modify State	onlyKToken
tokenURI	Public	-	-

KTokenLockedDepositNftDesign			
Function Name	Visibility	Mutability	Modifiers
buildTokenURI	External	-	-
generateBase64Image	Private	-	-
numberToRoundedString	Public	-	-

KTokenOpenPnlFeed			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
updateRequestsStart	Public	Can Modify State	onlyOwner
updateRequestsEvery	Public	Can Modify State	onlyOwner
updateRequestsCount	Public	Can Modify State	onlyOwner
updateRequestsInfoBatch	External	Can Modify State	onlyOwner
forceNewEpoch	External	Can Modify State	onlyOperator
makeOpenPnlRequest	External	Can Modify State	onlyOperator
effectiveOpenPnlRequest	Internal	Can Modify State	-
startNewEpoch	Private	Can Modify State	-
average	Private	-	-

4.3 Vulnerability Summary

[N1] [High] Potential interest rate attack

Category: Arithmetic Accuracy Deviation Vulnerability

Content

In the SideVault and SideVaultWithPending contracts, users can can deposit assets and obtain the corresponding share of the SideVault by calling the boost function, But there is a risk of interest rate inflation attacks here:

Consider this example: Bob finds out that Alice is making a boost (e.g. via mempool).

Pre-condition: no one deposit before （totalSupply = 0)

Now, Alice wants to deposit 1 (1 * 1e18 wei) want token and the tx is spied on by the attacker(Bob). Here is the breakdown:

	totalSupply	balance()
original state	0	0
(after) Step 1	1	1
(after) Step 2	1	1e18 + 1
(after) Step 3	1	2 * 1e18 + 1

1.Bob front-runs Alice and boosts 1 wei want token and gets 1 share: since totalSupply is 0, shares = amount = 1.

2.Bob then directly transfers 1 * 1e18 wei want token to the strategy contract, making the balance of the strategy (balance()) become 1e18 + 1 wei.

3.Alice boosts 1e18 wei quote token. However, alice gets 0 shares: 1e18 * 1 / (1e18 + 1) = 0. At this time, Alice gets 0 shares, totalSupply remains at 1.

4.Bob still has the 1 only share ever minted and the withdrawal of that 2 share takes away everything in the strategy, including the alice’s 1e18 wei quote token.

Code Location:

src/sidevault/SideVault.sol#L153

```
    function boost(address _account, uint _assets) public nonReentrant
onlySideVaultEntry {
        ...

        uint shares = 0;
        if (totalSupply() == 0) {
            shares = _assets;
        } else {
            shares = (_assets * totalSupply()) / _pool;
        }
        _mint(_account, shares);

        ...
    }
```

contracts/sidevault/SideVaultWithPending.sol#L156

```
    function boost(address _account, uint _assets) public nonReentrant
onlySideVaultEntry {
        ...

        uint shares = 0;
        if (totalSupply() == 0) {
            shares = _assets;
        } else {
            shares = (_assets * totalSupply()) / _pool;
        }
        _mint(_account, shares);

        ...
    }
```

Solution

Introduce a offset for internal accounting.

Refs: https://ethereum-magicians.org/t/address-eip-4626-inflation-attacks-with-virtual-shares-and-assets/12677/3

Status

Fixed

[N2] [Critical] Incorrect update of pending profit

Category: Design Logic Audit

Content

In the SideVaultWithPending contract, at this point, users can use the unboost function to withdraw the previously deposited assets and the profits generated during the period. If the funds in the strategy contract on the day are insufficient to meet the withdrawal amount required, a liability will occur. When the actual received token quantity is less than the profit to be obtained, the actually received tokens will first be used to pay part of the profit, and the unobtained principal and the remaining profit will be updated and recorded as a liability. However, an incorrect value is updated when updating the remaining profit to be claimed. The part of the profit that has been paid previously is wrongly updated as the profit that has not been claimed yet.

Code Location:

src/sidevault/SideVaultWithPending.sol#L209

```
    function unboost(address _account, uint _assets) public onlySideVaultEntry {
        ...

        if (debtToSideVault > 0) {
            ...

            if (wantReceived > 0) {
                if (wantReceived < profit) {
                    accountPendingUnBoostAsset[_account] += withdrawAssets;
                    accountPendingUnBoostAssets[_account][todayIdx] += withdrawAssets;
                    accountPendingUnBoostProfits[_account][todayIdx] += wantReceived;
                    erc20.safeTransfer(_account, wantReceived);
                    emit SideVaultUnBoost(_account, 0, wantReceived,
                        withdrawAssets, profit - wantReceived, todayIdx,
address(this), address(erc20), ++sid);
                } else {
                    ...

                }
            } else {
                ...
            }
        } else {
            ...
        }
    }
```

Solution

It is recommended to update the value of accountPendingUnBoostProfits to the amount of unclaimed profit, that is, profit - wantReceived.

Status

Fixed

[N3] [Medium] Missing trustedForwarder check on initialization

Category: Scoping and Declarations Audit

Content

In the DelegateCollection contract, when calling the initialize function to set the address of trustedForwarder, there is no check to see if it is consistent with the trustedForwarder set for ERC2771 in the constructor. If the contract needs to be deployed on other chains in the future and these two values are inconsistent, it may lead to unexpected errors.

Code Location:

src/core/DelegateCollection.sol#L43

```
    constructor(address _trustedForwarder)
ERC2771ContextUpgradeable(_trustedForwarder) {
        _disableInitializers();
    }

    function initialize(
        address _trustedForwarder,
        address _kiloStorageAddr
    ) public initializer {

        trustedForwarder = _trustedForwarder;

        ...
    }
```

Solution

It is recommended to add a check for the trustedForwarder address in the initialize function.

Status

Fixed

[N4] [High] TP/SL orders will not be cancelled when closing a position

Category: Design Logic Audit

Content

In the PositionRouter contract, when the executePositionsWithPricesT function is called to perform a position - closing operation, if a position is fully closed, orders will be taken from cancelOrders and looped through for cancellation. According to the comments written by the project team, after a position is fully closed, the corresponding take-profit or stop-loss orders should be cancelled. However, in the delegateExecutePositions function of the DelegateCollection contract, the take-profit or stop-loss orders corresponding to the position are not obtained and passed into cancelOrders for use by the executePositionsWithPricesT function of the PositionRouter contract. This means that even after a position is fully closed, the take-profit or stop-loss orders will not be cancelled simultaneously. If a user opens a new position before cancellation, it may lead to the use of old take-profit or stop-loss orders and the position being quickly liquidated.

Code Location:

src/core/DelegateCollection.sol#L289-323

```
    function delegateExecutePositions(
        address[] calldata _tokens,
        uint256[] calldata _prices,
        address[] calldata _increasePositionAccounts,
        uint256[] calldata _increaseSubIndexes,
        address[] calldata _decreasePositionAccounts,
        uint256[] calldata _decreaseSubIndexes,
        address payable _executionFeeReceiver
    ) external {
        require(msg.sender == trustedForwarder, "DelegateCollection: not
trustedForwarder");
        require(ITrustedForwarder(trustedForwarder).isKeeper(tx.origin),
"DelegateCollection: not keeper");
        IPositionRouter.CloseOrdersWithoutPosition[] memory cancelOrders;

        ...

        positionRouter.executePositionsWithPricesT(
            _tokens,
            _prices,
            _increasePositionAccounts,
            increasePositionIndexes,
```

```
            _decreasePositionAccounts,
            decreasePositionIndexes,
            cancelOrders,
            _executionFeeReceiver
        );
    }
```

Solution

It is recommended to add the operation of obtaining take-profit or stop-loss orders and passing them into the cancelOrders array in the delegateExecutePositions function of the DelegateCollection contract.

Status

Fixed

[N5] [Low] Potential Dos attack in token permit execution

Category: Denial of Service Vulnerability

Content

In the DelegateCollection contract, Users perform account authorization operations by calling the approveDelegate function, which in turn calls the permit function of the ERC20 token contract to grant a spending limit. However, if the permit function is preemptively executed by a malicious user (attackers can obtain the corresponding parameters by monitoring the mempool), it may roll back, causing the entire transaction to fail.

Code Location:

src/core/DelegateCollection.sol#L345-353

```
    function approveDelegate(
        address delegate,
        bool enable,
        PermitParams[] memory permitParams
    ) external nonReentrant {
        require(permitParams.length < 10, "inconsistent params length");
        ITrustedForwarder forwarder = ITrustedForwarder(trustedForwarder);
        for (uint256 i; i < permitParams.length; i++) {
            IERC20Permit(permitParams[i].token).permit(
                permitParams[i].owner,
                permitParams[i].spender,
                permitParams[i].value,
                permitParams[i].deadline,
                permitParams[i].v,
                permitParams[i].r,
```

```
                permitParams[i].s
            );
        }

        ...
    }
```

Solution

It is recommended to wrap the permit function call with a try-catch block or implement conditional checks to ensure that if a permit call in a for-loop fails, it does not cause the entire transaction to roll back.

Status

Fixed

[N6] [Suggestion] Missing scope limit

Category: Others

Content

1.In the KiloPriceFeed contract, the owner role can call the setSpreadEnabled and setDefaultSpread functions to set the parameters defaultSpread and spreads[_token]. However, there is a lack of range checking here, and if the values set are too large, it may lead to unexpected situations.

Code Location:

src/core/oracle/OracleManager.sol

```
    function setDefaultSpread(uint256 _defaultSpread) external onlyOwner {
        ...
    }

    function setSpread(address _token, uint256 _spread) external onlyOwner {
        ...
    }
```

2.In the KiloStorageManager contract, the owner role can call the setMinMargin, setExposureMultiplier, setUtilizationMultiplier, setMaxExposureMultiplier, setLiquidationParams and setParameters function to set the core parameters for trading. However, there is a lack of range checking here, and if the values set are too small or large, it may lead to unexpected situations.

Code Location:

src/core/KiloStorageManager.sol

```
    function setMinMargin(uint64 _minMargin) external onlyOwner {
        ...
     }

    function setExposureMultiplier(uint16 _exposureMultiplier) external onlyOwner {
        ...
     }

     function setUtilizationMultiplier(uint16 _utilizationMultiplier) external
onlyOwner {
        ...
     }

     function setMaxExposureMultiplier(uint8 _maxExposureMultiplier) external
onlyOwner {
        ...
     }

     function setLiquidationParams(uint16 _liquidationBounty, uint16
_liquidationThreshold) external onlyOwner {
        ...
     }

    function setParameters(
        uint32 _maxShift, ///@dev not used any more, replaced by ProductManage
        uint32 _minProfitTime,
        bool _canUserStake,
        bool _allowPublicLiquidator,
        uint16 _exposureMultiplier,
        uint16 _utilizationMultiplier,
        uint8 _maxExposureMultiplier,
        uint16 _liquidationBounty,
        uint16 _liquidationThreshold
     ) external onlyOwner {
        ...
     }
```

3.In the PendingReward contract, the owner role can call the setProtocolRewardRatio function to set the protocolRewardRatio parameter. However, there is a lack of range checking here, and if the values set are too large, it may lead to unexpected situations.

Code Location:

src/core/PendingReward.sol

```
    function setProtocolRewardRatio(uint256 _protocolRewardRatio) external onlyOwner {
        ...
     }
```

4.In the ProductManager contract, the owner role can call the addProduct and updateProduct functions to set the fee[token] parameter. However, there is a lack of range checking here, and if the values set are too large(exceeds the MAX_FEE_RATE.), it may lead to unexpected situations.

Code Location:

src/core/ProductManager.sol#L70-107

```
    function addProduct(address token, Product memory _product) external onlyOwner {
        ...

        fee[token] = _product.fee;

        ...
     }
    function updateProduct(address token, Product memory _product) external onlyOwner
{
        ...

        fee[token] = _product.fee;

        ...
     }
```

5.In the TrustedForwarder contract, the owner role can call the updateExecutionQuoteFee function to set the quoteExecutionFee parameter. However, there is a lack of range checking here, and if the values set are too large, it may lead to unexpected situations.

Code Location:

src/core/TrustedForwarder.sol

```
function updateExecutionQuoteFee(uint256 _quoteExecutionFee) external onlyOwner {
     ...
 }
```

6.In the HybridVault contract, the owner role can call the configHToken function to set the config.liquidationThreshold parameter. However, there is a lack of range checking here, and if the values set are too large, it may lead to unexpected situations.

Code Location:

src/hybirdvault/HybridVault.sol

```
function configHToken(
     uint hTokenId,
     address hTokenAddress,
     string memory name,
     address originToken,
     uint ltv,
     uint liquidationThreshold
 ) public onlyOwner {
     ...

     config.liquidationThreshold = liquidationThreshold;

     ...
 }
```

Solution

It is recommended to add reasonable range limit checks for the setting of core parameters.

Status

Acknowledged

[N7] [Suggestion] Missing the event record

Category: Others

Content

The following functions are missing event records after calling the modification:

Code Location:

src/core/KiloStorageManager.sol

```
    function initTrustedForwarder(address _delegateCollectionAddr, address
_trustedFroward) external onlyOwner reinitializer(4) {
        ...
    }
```

src/core/MarketOrderWithTriggerOrder.sol

```
    function setOrderBook(address _orderBook) external onlyOwner {
        ...
    }

    function setPositionRouter(address _positionRouter) external onlyOwner {
        ...
    }

    function setKiloStorageAddr(address _kiloStorageAddr) external onlyOwner {
        ...
    }
```

src/core/PendingReward.sol

```
    function setPerpTrade(address _perpTrade) external onlyOwner {
        ...
    }
```

src/core/OrderBook.sol

```
    function initTrustedForwarder(address _delegateCollectionAddr, address
_trustedForwarderAddr) external onlyOwner reinitializer(3) {
        ...
    }

    ...

    function setApprovedRouter(address _manager, bool _isActive) external onlyOwner {
        ...
    }
```

src/core/PerpTrade.sol

```
    function setVaultStakeReward(address _vaultStakeReward) external onlyOwner {
        ...
    }

    function setApprovedRouter(address _manager, bool _isActive) external onlyOwner {
        ...
    }

    function setOracle(address _oracle) external onlyOwner {
        ...
    }

    function setMarginFeeManager(address _marginFeeManager) external onlyOwner {
        ...
    }

    function setLiquidator(address _liquidator, bool _isActive) external onlyOwner {
        ...
    }
```

src/core/PositionRouter.sol

```
    function setApprovedRouter(address _manager, bool _isActive) external onlyOwner {
        ...
    }
```

src/core/ProductManager.sol

```
    function batchSetMinSpread(uint256[] calldata _productIds, uint256[] calldata
_minSpreads) external {
        ...
    }
```

src/core/VaultStakeReward.sol

```
    function setMIN_LOCK_DURATION(uint _MIN_LOCK_DURATION) external onlyOwner {
        ...
    }
```

src/hybirdvault/HybridVault.sol

```
    function setCanUserStake(bool _canUserStake) external onlyOwner {
         ...
     }
```

src/hybirdvault/HybridVault.sol

```
    function setHybridVault(address _hybridVault) external onlyOwner {
         ...
     }
```

src/peripherals/KiloExReader.sol

```
    function setTrustForwarder() external {
         ...
     }
```

src/tradereward/AffiliateRewardDistributor.sol

src/tradereward/TradeRewardDistributor.sol

```
    function setProtocolReward(address _protocolReward) external onlyOwner {
         ...
     }
```

src/tradereward/ProtocolReward.sol

```
    function setAffiliateRewardDistributorAddr(address
_affiliateRewardDistributorAddr) external onlyOwner {
         ...
     }

     function setAffiliateRewardDistributorMaxClaim(uint256
_affiliateRewardDistributorMaxClaim) external onlyOwner {
         ...
     }
```

Solution

It is recommended to record events when sensitive parameters are modified for self-inspection or community review.

Status

Acknowledged; Partially fixed.

[N8] [Low] Missing minimum delay block check in OrderBook

Category: Design Logic Audit

Content

In the PositionRouter contract, when executing an order, the _validateExecution function is called to check whether the order is executable. It checks that the current execution block number should be greater than the block number when the order was created plus minBlockDelayKeeper. This measure can prevent malicious keepers from instantaneously influencing prices for arbitrage operations. However, the validatePositionOrderPrice function in the OrderBook contract lacks such a check.

Code Location:

src/core/OrderBook.sol#L316-326

```
    function validatePositionOrderPrice(
        bool _triggerAboveThreshold,
        uint256 _triggerPrice,
        uint256 _productId,
        bool _isLong
    ) public view returns (uint256 currentPrice) {
        ...
    }
```

Solution

It is recommended to also add a check for the minimum delay block at execution in the validatePositionOrderPrice function of the OrderBook contract.

Status

Acknowledged

[N9] [Medium] Lack of checking of product status

Category: Design Logic Audit

Content

In the OrderBook contract, users can create an increase position order by calling the createIncreaseOrderDelegateV3

function. However, when creating an order, there is a lack of inspection of the product status (that is, productToken is not equal to the zero address and isActive is equal to true). As a result, users can create orders for unopened markets, which is not as expected.

Code Location:

src/core/OrderBook.sol#L420-460

```
function createIncreaseOrderDelegateV3(
     uint256 _productId,
     uint256 _margin,
     uint256 _leverage,
     bool _isLong,
     uint256 _triggerPrice,
     bool _triggerAboveThreshold,
     uint256 _executionFee,
     bytes32 _referralCode,
     address _account,
     bool _1ct,
     bytes memory _extraInfo
 ) public payable delegate(address(kiloStorage), _account, _1ct) nonReentrant {
     ...
 }
```

Solution

It is recommended to add a check for the product status in the createIncreaseOrderDelegateV3 function.

Status

Fixed

[N10] [Medium] Lack of check for leverage range

Category: Design Logic Audit

Content

1.In the OrderBook contract, users can create an increase position order by calling the createIncreaseOrderDelegateV3 function. However, in this function, only the minimum limit of the leverage is checked, while whether the leverage parameter passed in by the user exceeds the maximum limit is not examined.

Code Location:

src/core/OrderBook.sol#L420-460

```
function createIncreaseOrderDelegateV3(
      uint256 _productId,
      uint256 _margin,
      uint256 _leverage,
      bool _isLong,
      uint256 _triggerPrice,
      bool _triggerAboveThreshold,
      uint256 _executionFee,
      bytes32 _referralCode,
      address _account,
      bool _1ct,
      bytes memory _extraInfo
  ) public payable delegate(address(kiloStorage), _account, _1ct) nonReentrant {
      ...
  }
```

2.In the PositionRouter contract, the __createIncreasePosition function is used to create a market-price increase order. However, in this function, there is a lack of inspection of the leverage range.

Code Location:

src/core/PositionRouter.sol#L528-568

```
 function __createIncreasePosition(
      address _account,
      uint256 _productId,
      uint256 _margin,
      uint256 _leverage,
      bool _isLong,
      uint256 _acceptablePrice,
      uint256 _executionFee,
      bytes32 _referralCode,
      bytes memory extraInfo
  ) internal {
      ...
  }
```

Solution

It is recommended to check both the minimum and maximum limits of the leverage when creating an increase order.

Status

Fixed;

[N11] [Medium] Lack of position check when creating a reduction order

Category: Design Logic Audit

Content

In the OrderBook contract, the _createDecreaseOrder function is used to create a limit decrease order. However, this function does not check whether the user already has a position in the corresponding product. That is, a user can create a decrease order even if they have not opened a position before. This means that if the user opens a position later, it may trigger the previously created decrease order and cause losses.

Code Location:

src/core/OrderBook.sol#L708-751

```
function _createDecreaseOrder(
    address _account,
    uint256 _productId,
    uint256 _size,
    bool _isLong,
    uint256 _triggerPrice,
    bool _triggerAboveThreshold,
    uint256 _executionFee,
    bytes memory _extraInfo
) private {
    ...
}
```

Solution

It is recommended to add a check to see if the user has an existing position in the _createDecreaseOrder function of the OrderBook contract.

Status

Fixed;

[N12] [Suggestion] Missing order type check when updating orders

Category: Design Logic Audit

Content

In the OrderBook contract, the updateDecreaseOrderDelegate function is used to update the information of an already-created limit order. However, this function does not check whether the order type is a take-profit or stop-loss

order. If the order is a take-profit or stop-loss order, modification of triggerAboveThreshold should not be allowed. Otherwise, it does not meet the expectations of take-profit or stop-loss orders.

Code Location:

src/core/OrderBook.sol#L851-875

```
function updateDecreaseOrderDelegate(
    uint256 _orderIndex,
    uint256 _size,
    uint256 _triggerPrice,
    bool _triggerAboveThreshold,
    address _account,
    bool _lct
) public delegate(address(kiloStorage), _account, _lct) nonReentrant {
    ...
}
```

Solution

It is recommended to add a check for the order type in the updateDecreaseOrderDelegate function. If it is a take-profit or stop-loss order, modifying triggerAboveThreshold to the opposite direction should not be allowed.

Status

Fixed

[N13] [Information] Lack of updating funding fees and borrowing fees when reducing positions

Category: Design Logic Audit

Content

In the PerpTrade contract, the decreasePositionWithId function is called when executing a position - reducing order. When reducing the position, it will obtain the latest funding rate and borrowing rate and deduct the corresponding fees. However, after reducing the position, the funding fees and borrowing fees of the position are not updated. This results in subsequent transactions using the old position.funding and position.borrowing when calculating the funding fees.

Code Location:

src/core/PerpTrade.sol

```
    function decreasePositionWithId(
         uint256 positionId,
         uint256 margin,
         uint256 orderType,
         uint256 oraclePrice,
         bytes memory extraInfo
     )  internal {
        ...

         (vars.fundingPayment, vars.borrowingFee) =
PerpTradeUtil._getMarginFee(marginFeeManager, position.productId, position.isLong,
position.leverage, margin, position.funding, position.borrowing);
         vars.pnl = PerpTradeUtil._getPnl(position.isLong, uint256(position.price),
uint256(position.leverage), margin, vars.price) - vars.fundingPayment -
int256(vars.borrowingFee);
         ...

         if (vars.isFullClose) {
             kiloStorage.clearPosition(positionId);
         } else {
             position.margin -= uint64(margin);
             kiloStorage.updatePositionMargin(positionId, position.margin);
         }
     }
```

Solution

It is recommended to update the funding fees and borrowing fees of the current position after reducing the position.

Status

Acknowledged; The project team responded: the position.funding and position.borrowing do not need to be updated when reducing positions and always remain the values at the time of opening a position. In this way, when charging fees, the funding to be paid from the opening of the position to the present is calculated.

[N14] [Suggestion] Redundant code

Category: Others

Content

There are useless codes in the file and codes that are not used in actual business.

Code Location:

src/core/OrderBook.sol#L877-879

```
    function _transferOutETH(uint256 _amountOut, address payable _receiver) private {
        ...
     }
```

src/core/PositionRouter.sol#L263-266

```
    function setIsUserExecuteEnabled(bool _isUserExecuteEnabled) external onlyOwner {
         ...
     }
```

src/core/VaultStakeReward.sol

```
    function getStake(address stakeOwner) external view returns (Stake memory) {
         ...
     }
```

src/core/VaultStakeReward.sol

```
    function reBalanceForShares(address owner, int changedShares) external
onlyHybridVault {
         ...
     }
```

src/sidevault/listadao/SideVaultWithPending.sol

src/sidevault/listadao/SideVault.sol

```
   modifier onlyHybridVault() {
         _checkHybridVault();
         _;
     }

     function _checkHybridVault() internal view virtual {
         require(hybridVault == msg.sender, "Vault: caller is not hVault");
     }
```

src/sidevault/SideVaultEntry.sol

```
    function getDayIdx() external view returns(uint) {
         ...
```

```
    }
```

Solution

It is recommended to remove redundant commented codes and useless codes.

Status

Fixed

[N15] [Suggestion] Missing event record when market order cancellation fails

Category: Others

Content

In the PositionRouter contract, the executePositionsWithPricesT function is used to execute market-price increase or decrease orders. When an error occurs during execution, such as a failed check, it will fail, and at this time, the cancelIncreasePosition function will be called to cancel the market-price order. However, when the order cancellation fails, no event is emitted, making it difficult for the front-end or users to perceive, and the market-price order still remains in the order list.

Code Location:

src/core/PositionRouter.sol

```
    function executePositionsWithPricesT(
        address[] calldata _tokens,
        uint256[] calldata _prices,
        address[] calldata _increasePositionAccounts,
        uint256[] calldata _increasePositionIndexes,
        address[] calldata _decreasePositionAccounts,
        uint256[] calldata _decreasePositionIndexes,
        CloseOrdersWithoutPosition[] calldata cancelOrders,
        address payable _executionFeeReceiver
    ) external onlyPositionKeeper {
        ...

        for (uint256 i; i < incrLength; ) {
            ...

            try this.executeIncreasePosition(key, _executionFeeReceiver) returns
(bool _wasExecuted) {
                if (!_wasExecuted) {
                    ...
```

```
                }
            } catch Error(string memory executionError) {
                ...

                try this.cancelIncreasePosition(key, _executionFeeReceiver) returns
(bool _wasCancelled) {
                    if (!_wasCancelled) { break; }
                } catch {}
            } catch (bytes memory /*lowLevelData*/) {
                ...

                try this.cancelIncreasePosition(key, _executionFeeReceiver) returns
(bool _wasCancelled) {
                    if (!_wasCancelled) { break; }
                } catch {}
            }

            unchecked { ++i; }
        }

        uint256 cancelOrdersLen = cancelOrders.length;
        for (uint256 i; i < decrLength; ) {
            ...

            try this.executeDecreasePosition(key, _executionFeeReceiver) returns
(bool _wasExecuted, bool _isFullClose) {
                if (_wasExecuted) {
                    ...
                } else {
                    ...
            } catch Error(string memory executionError)  {
                ...

                try this.cancelDecreasePosition(key, _executionFeeReceiver) returns
(bool _wasCancelled) {
                    if (!_wasCancelled) { break; }
                } catch {}
            } catch (bytes memory /*lowLevelData*/) {
                ...

                try this.cancelDecreasePosition(key, _executionFeeReceiver) returns
(bool _wasCancelled) {
                    if (!_wasCancelled) { break; }
                } catch {}
            }
            unchecked { ++i; }
        }
    }
```

Solution

It is recommended that an event be emitted upon failed order cancellation to make it perceptible to the front-end or users. Alternatively, if the cancellation fails, the order could be forcibly deleted and funds returned to the user.

Status

Fixed

[N16] [Low] Missing check for product creation status

Category: Design Logic Audit

Content

In the ProductManager contract, the addProduct function is used to create a product corresponding to a specified token. However, during creation, it does not check whether a product corresponding to this token has been created before. This may lead to a situation where two different products in the market correspond to the same token.

In addition, the updateProduct function is used to update various configurations of a specified product. However, this function does not check whether the market corresponding to the token exists. This means that configurations can be directly updated for products that have not been created yet.

Code Location:

src/core/ProductManager.sol#L70-107

```
    function addProduct(address token, Product memory _product) external onlyOwner {
        ...
    }

    function updateProduct(address token, Product memory _product) external onlyOwner
{
        ...
    }
```

Solution

It is recommended to check whether a product corresponding to the token already exists when creating or updating.

Status

Fixed

[N17] [Medium] Lack of consistency check between old and new tokens

Category: Design Logic Audit

Content

In the ProductManager contract, the updateProduct function is used to update various configurations of a specified product. However, during the update process, it does not check whether the passed - in token parameter is consistent with the _product.token parameter. Moreover, only productToken[_product.productId] is updated, while the corresponding id[token] is not updated. If they are inconsistent, unexpected problems may occur.

Code Location:

src/core/ProductManager.sol#L98-107

```
    function updateProduct(address token, Product memory _product) external onlyOwner
{
        ...
    }
```

Solution

It is recommended to add a check for the consistency between the token parameter and the _product.token parameter in the updateProduct function, and the corresponding id[token] also needs to be updated.

Status

Fixed

[N18] [Medium] Incorrect trade fee rate calculation

Category: Design Logic Audit

Content

In the ProductManager contract, the getTradeFeeRateV2 function is used to calculate the transaction fee rate charged during a trade. When tradeFeeRate < 1e4, tradeFeeRate is multiplied by 1e4 and then returned as the transaction fee rate. However, under normal expectations, the transaction fee rate should not exceed MAX_FEE_RATE, which is 1e6. If tradeFeeRate is between 1e2 and 1e4, the transaction fee rate finally obtained by the getTradeFeeRateV2 function will exceed this maximum limit, leading to unexpected trading results.

Code Location:

src/core/ProductManager.sol#L210-212

```
    function getTradeFeeRateV2(uint256 productId, address account) external view
returns (uint256) {
        uint256 tradeFeeRate = fee[productToken[productId]];
        if (tradeFeeRate <= RATE_BASE) {
            tradeFeeRate = tradeFeeRate * RATE_BASE;
        }

        ...
    }
```

Solution

It is recommended to change the check of tradeFeeRate <= RATE_BASE to tradeFeeRate <= 1e2 here.

Status

Fixed

[N19] [Suggestion] Missing zero address check

Category: Others

Content

In the TrustedForwarder contract, the Owner role can call the withdrawQuoteToken function to transfer the quote tokens to any addresses. However, here there is a lack of a non-zero address check for the `to` parameter. If a zero address is passed in by mistake, it may cause the tokens to be lost.

Code Location:

src/core/TrustedForwarder.sol

```
    function withdrawQuoteToken(address to, uint256 amount) external onlyOwner {
        ...
    }
```

Solution

It is recommended to add a zero address check.

Status

Fixed

[N20] [Low] Missing variable update in refill function

Category: Design Logic Audit

Content

In the VaultStakeReward contract, the refill function is used to manually transfer assets into the contract and offset part of the vault's losses. However, in this function, only variables such as accPnlPerToken, accPnlPerTokenUsed, and totalRefilled are updated, while dailyAccPnlDelta, totalLiability, and totalClosedPnl are not updated. These three variables are updated every time assets are received in the receiveAssets function.

Code Location:

src/core/VaultStakeReward.sol#L892-908

```
    function refill(uint assets) external onlyHybridVault {
         ...
     }
```

Solution

It is recommended to also update the dailyAccPnlDelta, totalLiability, and totalClosedPnl variables in the refill function.

Status

Acknowledged

[N21] [Low] Incorrect check logic for softCapQuoteAssetsMinB

Category: Design Logic Audit

Content

In the HybridVault contract, the setQuoteAssetsMinBp function is used to set the quoteAssetsMinBp and softCapQuoteAssetsMinBp variables. When checking the range of the newly set softCapQuoteAssetsMinBp, the comparison is made using _quoteAssetsMinBp and PercentageMath.PERCENTAGE_FACTOR instead of _softCapQuoteAssetsMinBp.

Code Location:

src/hybirdvault/HybridVault.sol#L165

```
    function setQuoteAssetsMinBp(uint _quoteAssetsMinBp, uint
_softCapQuoteAssetsMinBp) external onlyOwner {
```

```
        ...

        require(_softCapQuoteAssetsMinBp > 0 && _quoteAssetsMinBp <
PercentageMath.PERCENTAGE_FACTOR, "HybridVault: invalid value");

        ...
    }
```

Solution

It is recommended to change it here to compare _softCapQuoteAssetsMinBp with PercentageMath.PERCENTAGE_FACTOR to check the range.

Status

Fixed

[N22] [Medium] Missing non-zero check for vusd quantity calculation

Category: Design Logic Audit

Content

In the deposit and reBalance functions of the HybridVault contract, the calculation of the amount of VUSD is obtained by multiplying the amount of hToken to be minted by the LTV and then converting according to the token precision and price. However, there is no non-zero check on the calculated amount of VUSD. If the amount of tokens passed in is very small or the hToken corresponding to the QUOTE_TOKEN has not been configured yet (i.e., QUOTE_PRECISION == 0), the calculated amount of VUSD will be 0, and the user's assets will be locked in the contract.

Code Location:

src/hybirdvault/HybridVault.sol

```
    function deposit(uint hTokenId, uint hAssets, bytes calldata priceData) external
isActive payable returns (uint) {
        ...

        uint vusd = hTokenId == QUOTE_TOKEN_ID ? hAssets
            : hAssets.percentMul(hTokenConfig.ltv)  * price * QUOTE_PRECISION /
(hTokenConfig.precision * PRICE_BASE);

        ...
```

```
        address sender = msg.sender;
        IERC20Upgradeable(hTokenConfig.oToken).safeTransferFrom(sender, address(this),
hAssets);

        ...
    }

   function reBalance(
        address[] memory owners,
        uint[] memory hTokenIds,
        uint[] memory prices
    ) public onlyOperator {
        ...

        for (uint i; i < len; ) {

            uint hTokenAmount = IERC20Upgradeable(ht.hToken).balanceOf(owner);
            //uint newVusd = hTokenAmount.percentMul(ht.ltv)  * hTokenPrice  /
PRICE_BASE;
            uint newVusd = hTokenAmount.percentMul(ht.ltv)  * hTokenPrice *
QUOTE_PRECISION / (ht.precision * PRICE_BASE);

            ...

        }
    }
```

Solution

It is recommended to add a check that the amount of VUSD is not equal to 0, and also check that QUOTE_PRECISION is not equal to 0.

Status

Fixed; The project team responded: There is no need to check QUOTE_PRECISION. The quote token will be initialized directly after the contract is deployed. It is the basis for the operation of vault. There will be no situation where QUOTE_PRECISION=0.

[N23] [Suggestion] Missing hTokenId check in rebalance function

Category: Design Logic Audit

Content

In the HybridVault contract, the reBalance function is used to redeem and burn the old VUSD and calculate and mint new VUSD for deposit when there are significant price fluctuations or changes in LTV. However, in this function, there

is no check on the passed-in hTokenIds parameter. When hTokenId is equal to QUOTE_TOKEN_ID, a re-balance operation is not required.

Code Location:

src/hybirdvault/HybridVault.sol#L434-482

```
    function reBalance(
        address[] memory owners,
        uint[] memory hTokenIds,
        uint[] memory prices
    ) public onlyOperator {
        ...
    }
```

Solution

It is recommended to add a check in the reBalance function that hTokenId is not equal to QUOTE_TOKEN_ID.

Status

Fixed

[N24] [Information] Incorrect logic in collateralInQuote calculation

Category: Design Logic Audit

Content

In the HybridVaultLogic contract, the calculateUserAccountData function is used to calculate the user's account-related data, including the expected profit and loss and the user's account health factor. The value of the user's collateral is obtained by multiplying the quantity of the user's hToken by the liquidation threshold, and then performing token precision and price conversions.

However, in the hybirdvault contract, when converting the user's collateral into VUSD value, it will be multiplied by hTokenConfig.ltv, while in the calculateUserAccountData function, when calculating vars.collateralInQuote, it is not multiplied by hTokenConfig.ltv. This may lead to unexpected results. For example, when calculating the health factor, it will be larger than expected.

Code Location:

src/hybirdvault/HybridVaultLogic.sol#L105

```
    function calculateUserAccountData(
        address owner,
        IPriceRouter priceRouter,
        address vUSD,
        IKToken kToken,
        mapping(uint => DataTypes.HTokenConfig) storage hTokenConfigs,
        mapping(address => mapping(uint => DataTypes.AccountDeposit)) storage
accountDeposits,
        uint hTokenId
    ) internal view returns(DataTypes.UserAccountData memory vars) {
        ...

        vars.collateralInQuote +=
IERC20MetadataUpgradeable(ht.hToken).balanceOf(owner).percentMul(ht.liquidationThresho
ld) * priceRouter.getPriceNoOlderThan(hTokenId) * 10**vusdDecimals / (ht.precision *
PRICE_BASE);

        if (vars.profitsInQuote >= 0 || hTokenId == QUOTE_TOKEN_ID) {
            vars.healthFactor = type(uint).max;
        } else {
            vars.healthFactor = vars.collateralInQuote.percentDiv(uint(-
vars.profitsInQuote));
        }
    }
```

Solution

It is recommended to multiply hTokenConfig.ltv when calculating vars.collateralInQuote in the calculateUserAccountData function.

Status

Acknowledged; The project team responded: LTV and liquidationThreshold are separate. LTV is lower than liquidationThreshold to ensure that the system is more robust.

[N25] [Medium] Incorrect price time recorded in priceOfChainLink function

Category: Design Logic Audit

Content

In the PriceRouter contract, the priceOfChainLink function is used to obtain the price of a specified token and its price update time, and record them in the kiloExPrices mapping. However, the price update time recorded in the

kiloExPrices mapping is the current timestamp (block.timestamp), rather than the actual price submission time on Chainlink obtained from getChainlinkPrice.

Code Location:

src/hybirdvault/PriceRouter.sol

```
    function priceOfChainLink(uint tokenId, address token) internal {
        require(oracleSources[tokenId] == OracleSource.CHAINLINK, "PriceRouter: not allowed");
        require(tokenIdToChainLinkToken[tokenId] == token, "PriceRouter: tokenId and token not match");
        (uint price, uint timestamp) = kiloPriceFeed.getChainlinkPrice(token);
        kiloExPrices[tokenId] = PriceInfo(price, block.timestamp);
    }
```

Solution

It is recommended to change the price update time here to the actual price submission time obtained from Chainlink.

Status

Fixed

[N26] [Medium] Missing chainID check in the signature verification

Category: Replay Vulnerability

Content

In the PriceRouter contract, the priceOfSignature function allows users to submit price data and update the token price after signature verification. However, when hashing the data that needs to be signed, the chainId is not included in the hash. This means that if the project is deployed on multiple chains, the signature may be maliciously replayed by attackers on another chain, resulting in unexpected fee information being passed in.

Code Location:

src/hybirdvault/PriceRouter.sol#L154-166

```
    function priceOfSignature(uint tokenId, bytes calldata data) public {
        ...

        if(priceOfSignatureId[tokenId] < timestamp) {
            bytes32 _msgHash = toEthSignedMessageHash(getMessageHash(tokenId, price, timestamp));
```

```
            require(verify(_msgHash, signature), "PriceRouter: Invalid Signer!");

            ...
        }
    }
```

Solution

It's recommended to add the chainId in the hash calculation of the signature message.

Status

Fixed

[N27] [Medium] Missing check for the sideVault

Category: Design Logic Audit

Content

In the SideVaultEntry contract, the boost function is used to transfer assets into a specified sideVault to obtain additional returns. However, in this function, there is no check whether the passed-in _sideVault is in the sideVaults array. If a user transfers assets to a sideVault that has been deleted or is not created by the project team, it may cause unexpected risks.

Code Location:

src/sidevault/SideVaultEntry.sol#L72-77

```
    function boost(ISideVault _sideVault, address _account, uint _assets, uint
_hTokenId) external onlyHybridVault {
        ...
    }
```

Solution

It is recommended to add a check in the boost function to verify whether the passed-in _sideVault parameter exists in the sideVaults array.

Status

Fixed

[N28] [Suggestion] Missing balance check when claiming rewards

Category: Others

Content

In the ProtocolReward contract, the claimTradeReward, claimAffiliateReward and claimXkiloReward functions are used to claim the reward fees collected from trading in the PendingReward contract, and then transfer the collected tokens to the reward distribution contract. However, in these functions, there is no check to determine whether the token balance in the contract is greater than or equal to the passed-in _amount parameter after obtaining the reward by calling the withdrawProtocolReward function of the PendingReward contract. If the reward collected from the PendingReward contract is less than _amount, the entire transaction will fail and roll back.

Code Location:

src/tradereward/ProtocolReward.sol

```
    function claimTradeReward(uint256 _amount) external whenNotPaused {
        ...

        IPendingReward(pendingReward).withdrawProtocolReward();
        rewardToken.safeTransfer(msg.sender, _amount);
    }

    function claimAffiliateReward(uint256 _amount) external whenNotPaused {
        ...

        IPendingReward(pendingReward).withdrawProtocolReward();
        rewardToken.safeTransfer(msg.sender, _amount);
    }

    function claimXkiloReward(uint256 _amount) external whenNotPaused {
        require(msg.sender == tradeRewardDistributor, "ProtocolReward: not allowed");
        require (address(xkiloToken) != address(0), "ProtocolReward: xkiloToken
zero");
        xkiloToken.safeTransfer(msg.sender, _amount);
    }
```

Solution

It is recommended to add a check in the claimTradeReward, claimAffiliateReward and claimXkiloReward functions to see if the balance in the contract after claiming the reward from the PendingReward contract is not less than the _amount parameter.

Status

Acknowledged

[N29] [Suggestion] Use `safeMint()` Instead of `mint()`

Category: Design Logic Audit

Content

In the KTokenLockedDepositNft contract, the mint function is called by the kToken contract to mint deposit vouchers (ERC712 tokens) to a specified user. However, in this function, the _mint function is used instead of the _safeMint function. The mint function lacks a check on whether the recipient is a smart contract capable of handling ERC721 tokens. If the recipient is a contract address that cannot handle ERC721, the tokens will be locked forever.

Code Location:

src/vaultv2/KTokenLockedDepositNft.sol

```
function mint(address to, uint tokenId) external onlyKToken {
     ...
 }
```

Solution

It is recommended to replace the _mint function with the _safeMint function when minting NFTs. In addition, add the nonReentrant modifier to prevent re-entry.

Status

Fixed

[N30] [Medium] Risk of excessive authority

Category: Authority Control Vulnerability Audit

Content

1.In the KiloStorageManager contract, the owner role can set the addresses of core contracts involved in the project's business logic. If this role is set to an EOA address and its permission is compromised, it could affect the normal operation of the project.

Code Location:

src/core/KiloStorageManager.sol

```
    function setToken(address _token, uint256 _decimals) external onlyOwner {
        ...
    }

    function setMarginFeeManagerAddr(address _marginFeeManagerAddr) external
onlyOwner {
        ...
    }

    function setPerpTradeAddr(address _perpTradeAddr) external onlyOwner {
        ...
    }

    function setPendingRewardAddr(address _pendingRewardAddr) external onlyOwner {
        ...
    }

    function setKiloPriceFeedAddr(address _kiloPriceFeedAddr) external onlyOwner {
        ...
    }

    function setOrderBookAddr(address _orderBookAddr) external onlyOwner {
        ...
    }

    function setPositionRouterAddr(address _positionRouterAddr) external onlyOwner {
        ...
    }

    function setStakeRewardAddr(address _stakeRewardAddr) external onlyOwner {
        ...
    }

    function setProductManagerAddr(address _productManagerAddr) external onlyOwner {
        ...
    }
```

2.In the MarginFeeManager contract, the owner role can set the address of the operator by calling the setOperator function. The operator can then call the batchSetFundingMultiplier and batchSetFundingRate functions to set the fundingMultipliers and fundingRates. If this role is set to an EOA address and its permission is compromised, it could affect the normal operation of the project.

Code Location:

src/core/MarginFeeManager.sol

```
    function setOperator(address _operator, bool _isOperator) external onlyOwner {
        ...
    }

    function setFundingMultiplier(uint256 _productId, uint256 _fundingMultiplier)
external onlyOwner {
        ...
    }

    function batchSetFundingMultiplier(uint256[] calldata _productIds, uint256[]
calldata _fundingMultipliers) external {
        ...
    }
    function batchSetFundingRate(uint256[] calldata _productIds, int256[] calldata
_fundingRates) external {
        ...
    }
```

3.In the OrderBook contract, the owner can call the setKeeper and setApprovedRouter functions to set the keeper and approved router, and these two roles are involved in the functions of creating and executing market order. If these roles are set to the EOA addresses and their permissions are compromised, it could affect the normal operation of the project.

Code Location:

src/core/OrderBook.sol

```
    function setApprovedRouter(address _manager, bool _isActive) external onlyOwner {
        approvedRouters[_manager] = _isActive;
    }

    function setKeeper(address _account, bool _isActive) external onlyOwner {
        isKeeper[_account] = _isActive;
        emit OwnerSetKeeper(_account, _isActive);
    }
```

4.In the PendingReward contract, the owner role can call the setPerpTrade function to set the address of perpTrade. And perpTrade can call the updatePendingRewards and updateLiquidationReward functions to transfer the funds of

users in the kiloStorage contract to the pool as trading fees collected. If this role is set to an EOA address and its permission is compromised, it could affect the normal operation of the project.

Code Location:

src/core/PendingReward.sol

```
    function setPerpTrade(address _perpTrade) external onlyOwner {
        ...
    }
```

5.In the PerpTrade contract, the owner role can call the functions setApprovedRouter, setOracle and setMarginFeeManager to configure the approvedRouters, oracle and marginFeeManager. If this role is set to an EOA address and its permission is compromised, it could affect the normal operation of the project.

Code Location:

src/core/PerpTrade.sol

```
    function setApprovedRouter(address _manager, bool _isActive) external onlyOwner {
        ...
    }

    function setOracle(address _oracle) external onlyOwner {
        ...
    }

    function setMarginFeeManager(address _marginFeeManager) external onlyOwner {
        ...
    }
```

6.In the PositionRouter contract, the owner role can call the functions setApprovedRouter, setReferralStorage and setPositionKeeper to set the approvedRouters, the referralStorage contract, and the PositionKeeper role. If these roles are set to the EOA addresses and their permissions are compromised, it could affect the normal operation of the project.

Code Location:

src/core/PositionRouter.sol

```
    function setApprovedRouter(address _manager, bool _isActive) external onlyOwner {
        ...
    }

    function setReferralStorage(address _referralStorage) external onlyOwner {
        ...
    }

    function setPositionKeeper(address _account, bool _isActive) external onlyOwner {
        ...
    }
```

7.In the ProductManager contract, the owner role can set the operators by calling the setOperator function, and the operator role can set the reserve of a specified product by calling the batchSetReserve function. If these roles are set to the EOA addresses and their permissions are compromised, it could affect the normal operation of the project.

Code Location:

src/core/ProductManager.sol

```
    function batchSetReserve(uint256[] calldata _productIds, uint256[] calldata
_reserves) external {
        ...
    }

    function setOperator(address _operator, bool _isOperator) external onlyOwner {
        ...
    }
```

8.In the TrustedForwarder contract, the owner role can set the keeper role by calling the setKeeper function. The keeper role is mainly responsible for executing the trading signed by users. If these roles are set to the EOA addresses and their permissions are compromised, it could affect the normal operation of the project.

Code Location:

src/core/TrustedForwarder.sol

```
    function setKeeper(address keeper, bool enable) external onlyOwner {
        ...
    }
```

9.In the PriceRouter contract, the owner role can set the signer address by calling the setSignerAddress function, and the operator role can set the price source of tokens. If these roles are set to the EOA addresses and their permissions are compromised, it could affect the normal operation of the project.

Code Location:

src/core/PriceRouter.sol

```
    function setChainLinkOracleSource(uint tokenId, address token) external
onlyOperator {
        ...
    }

    function setPythOracleSource(uint tokenId, bytes32 pythId) external onlyOperator
{
        ...
    }

    function setKiloSignatureOracleSource(uint tokenId) external onlyOperator {
        ...
    }

    function setSignerAddress(address _signerAddress) external onlyOwner {
        ...
    }
```

10.In the VUSD contract, the owner role can call the setHybridVault function to set the hybridVault. And the hybridVault can call the mint and burn functions to mint or burn VUSD. If this role is set to an EOA address and its permission is compromised, it could affect the normal operation of the project.

Code Location:

src/hybridvault/VUSD.sol

```
    function setHybridVault(address _hybridVault) external onlyOwner {
        ...
    }
```

11.In the CommonReward and KolRewardDistributor contracts, the owner role can directly call the withdrawEmergency function to withdraw any reward tokens in the contract. If this role is set to an EOA address and its permission is compromised, it could affect the normal operation of the project.

Code Location:

src/peripherals/CommonReward.sol

src/peripherals/KolRewardDistributor.sol

```
    function withdrawEmergency(uint256 _rewardId) external onlyOwner {
         ...
     }
```

12.In the ListaDaoWbnbStrategy, AaveV3Strategy and VenusVTokenStrategy contracts, the owner role can call the setSideVault function to set the sideVault. And the sideVault can call the withdraw function to obtain the tokens in the deposit. If this role is set to an EOA address and its permission is compromised, it could affect the normal operation of the project.

Code Location:

src/sidevault/ListaDaoWbnbStrategy.sol

src/sidevault/AaveV3Strategy.sol

src/sidevault/VenusVTokenStrategy.sol

```
    function setSideVault(address _sideVault) external onlyOwner {
         ...
     }
```

13.In the XKiloDividends contract, the owner role can directly call the emergencyWithdraw function to withdraw any tokens in the contract. If this role is set to an EOA address and its permission is compromised, it could affect the normal operation of the project.

Code Location:

src/tokens/XKiloDividends.sol

```
    function emergencyWithdraw(IERC20Upgradeable token) public nonReentrant onlyOwner
{
         ...
     }
```

14.In the ProtocolReward contract, the owner role can call the claimProtocolReward function to withdraw rewards from the pendingReward contract and then use the withdrawReward function to transfer them. Additionally, the owner role can call the setTradeRewardDistributor and setAffiliateRewardDistributorAddr functions to set the addresses of the tradeRewardDistributor and affiliateRewardDistributor.

Code Location:

src/tradereward/ProtocolReward.sol

```
    function claimProtocolReward() external onlyOwner {
        ...
    }

    function setProtocolRewardReceiver(address _receiver) external onlyOwner {
        ...
    }

    function setTradeRewardDistributor(address _tradeRewardDistributor) external
onlyOwner {
        ...
    }

    function setAffiliateRewardDistributorAddr(address
_affiliateRewardDistributorAddr) external onlyOwner {
        ...
    }

    function withdrawReward(uint256 _amount) external onlyOwner {
        ...
    }
```

Solution

In the short term, transferring the ownership of core roles 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. The authority involving user funds should be managed by the community, and the authority involving emergency contract suspension can be managed by the EOA address. This ensures both a quick response to threats and the safety of user funds.

Status

Fixed; The project team responded: Currently, the permissions of key roles related to funds have been transferred to

multi-signature, and permission management will be further strengthened to achieve permission separation.management to achieve authority separation in the future.

Update: The permissions of the core roles have been transferred to the timelock contract for control. Here is the address of the timelock contract on the mainnet:

B2 chain: 0x939E8c7631381218a33577F7426CfdbB62382A90

Base: 0x54e8742a73d63d9681084eec4e9c3aa430d34aed

Bsc: 0xdfF3c7cE7a57f808c897d2e1D93f4c188644CfE1

Manta: 0x9108b3c4e6a9d7519c369a4b37f9b403534cd38d

OpBNB: 0x4b1dbc4b0ba9f9b0b29fc956946d27afe1cc9dc7

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002506060001	SlowMist Security Team	2025.04.21 - 2025.06.06	Low Risk

Summary conclusion: The SlowMist security team uses a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 1 critical, 2 high, 11 medium, 5 low risks, 9 suggestion and 2 information. All the findings were fixed or acknowledged. The code has been deployed to the mainnet.

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