LUNARAY
BLOCKCHAINSECURITY

# SMART CONTRACT SECURITY AUDIT REPORT

## For Rollup.Finance

20 September 2023

lunaray.co

# Table of Contents

# 1. Overview

On Sep 6, 2023, the security team of Lunaray Technology received the security audit request of the **ROLLUP.FINANCE project**. The team completed the audit of the **ROLLUP.FINANCE smart contract** on Sep 20, 2023. During the audit process, the security audit experts of Lunaray Technology and the ROLLUP.FINANCE project interface Personnel communicate and maintain symmetry of information, conduct security audits under controllable operational risks, and avoid risks to project generation and operations during the testing process.

Through communicat and feedback with ROLLUP.FINANCE project party, it is confirmed that the loopholes and risks found in the audit process have been repaired or within the acceptable range. The result of this ROLLUP.FINANCE smart contract security audit:

## Passed

Audit Report Hash:
C8D58259CA9E53905A7CC444C66C88B7B98C1EAACF0C68F088B39EC94729BF76

# 2. Background

## 2.1 Project Description

| | |
|---|---|
| **Project name** | Rollup.Finance |
| **Contract type** | Decentralized Perpetual Trading Platform |
| **Code language** | Solidity |
| **Public chain** | zkSync |
| **Project website** | https://rollup.finance |
| **Introduction** | Rollup.Finance is a decentralized derivatives exchange launched initially on zkSync, Layer 2 innovative solution for perpetual trading, creating a new model for on-chain perpetual trading and providing sustainable returns for users. |
| **Contract file** | FaucetToken.sol，SigVerify.sol，TimeVerify.sol，WETH9.sol，ReferralReader.sol，ReferralStorage.sol，FeeRewardDivider.sol，InsuranceFund.sol，ProfitRewardDivider.sol，RewardDivider.sol，RewardDistributor.sol，RewardRouter.sol，RewardTracker.sol，ChainUtils.sol，BasicRouter.sol，Config.sol，OrderBook.sol，PairsContract.sol，PositionManager.sol，Router.sol，StableToken.sol，StableVault.sol，Trading.sol，TradingExtension.sol |

## 2.2 Audit Range

**Smart contract file name and corresponding SHA256：**

| Name | SHA256 |
| --- | --- |
| FaucetToken.sol | 1D8A6E1028722D59B1CD48A2530E283082DDD11D33CC5A62D46997D16C1FE071 |
| SigVerify.sol | 45EC0614A4CC39D36D80BFD71905487E1898C42DC8517725D057411E7EF3392F |
| TimeVerify.sol | EFCD43EE8DCA3BB055DECDB30F4E1CA9B061E74CF3B28D5284061B1C72FE9198 |
| WETH9.sol | 839E5E751EA161494A12CCFB72F35FEB5CD030251248C169265036154896AD98 |
| ReferralReader.sol | 10B5D6BEE0B9F0C436770FFD9914B319F7656977F3C7C0AA40F5E39C84581D6F |
| ReferralStorage.sol | A2279254A5C5A14E8EE2A65B2E59F174D795663F6C5CD20B5A4B67A63609534F |
| FeeRewardDivider.sol | 47AF6E2FD0ABA7901D3FDF12B87933D403A8A210B75E91FF1664731919BE6DEA |
| InsuranceFund.sol | 48961EF3EB927D2ACB107E13C0E2D46AAC218DD5FC8B76DEB6E08F450F7BE47F |
| ProfitRewardDivider.sol | C84F8408373800548DAB9B6E17EC71A58A2E6A846EDDF493C2DCBAB8CDA60F04 |
| RewardDivider.sol | C55A91CAAE4D6BED078333192677A60335FFB9646D4CE7A484F9DF60D8984691 |

| | |
|---|---|
| RewardDistributor.sol | 411F82D3DD7C21170A871064A7885CA11E571B4462D0F55A0FD83434D6391378 |
| RewardRouter.sol | FAD2AAFF329F111DE3A5C4D79FC40BE9C0D2C730CCB7D51D48265B6904C8C573 |
| RewardTracker.sol | BF45A093AEC2C9A5E2119C406888F6E8E0CDA59BE89B032DE501FC4329C17535 |
| ChainUtils.sol | D37C26CD922B2913199FD34DAC117CB9A8160B2122BA161E74EDC9C611FA4322 |
| BasicRouter.sol | D9BF78928414B70E7B71835C53B504C8F680BB726F734A20555569134FFCDC03 |
| Config.sol | 76289ABB317F247078351F65A74828A0DBFFC9BFA3A12A5362D228B2707F8474 |
| OrderBook.sol | 29C8CF73980D7619A991184068E5F2B4BD9EA3962D21DEF9450505EE9638E787 |
| PairsContract.sol | 3737DBE6EACDAF7D60D5FF3C5CB41750AA951BC975D6FCAA6A566C0FCF48C719 |
| PositionManager.sol | 8C1ED4E7660087B55AC0CC09407B97B5729A7285695FDD54D082C1893AC87033 |
| Router.sol | 29C2767D15ECE920E95389B03358B00BD59D9C8815BED3CD897CF46E8C4A564D |
| StableToken.sol | AF86A593D63770A484C630B5CE624FF89E689C6CA21B11EC44F2EF3C43885550 |
| StableVault.sol | 15B224E123F24E9D6089A4D082D3D7DBF6C137CD5D20A9CB793183BAFA848938 |

| | |
|---|---|
| Trading.sol | 3B03F7A2F887E1A6B3ED85A81D0B5A268136D25C68523DE3155B7ECF5123EE32 |
| TradingExtension.sol | 75A05D30E341513E2F47838C58F8798E2C53B188106E007EF9AD88F587EBAB9D |

# 3. Project contract details

## 3.1 Contract Overview

### Config Contract

The contract implements the setup and access to global configuration information, as well as permission control over Vault and collateral tokens.

### StableVault Contract

The contract implements an upgradable ERC20 Token called RUSD by inheriting the ERC20Upgradeable contract, with deposit and withdrawal functionality that allows users to deposit stablecoins and receive a share of RUSD tokens, or withdraw stablecoins based on a share of RUSD tokens. The contract also provides a rewards feature, whereby the reward distributor can distribute rewards to users and allow them to collect them. The contract administrator can set the reward distributor, liquidity provider and withdrawal delay. The structure Withdraw in the contract defines the information related to the withdrawal request, including the address of the token to be withdrawn, the number of liquidity tokens, the number of tokens, the RUSD price at the time of the withdrawal request, the RUSD price at the time of the withdrawal execution, and the timestamp of the withdrawal release. The contract implements the core functions of the stablecoin vault through initialization functions, setup functions, deposit and withdrawal functions, reward functions, and management functions, which are managed and configured by the contract's administrator.

### StableToken Contract

This contract inherits the ERC20 contract from OpenZeppelin and implements a standard ERC20 Token with minting and destruction functions. The contract uses a whitelisting mechanism to manage mint privileges, and only addresses set to mint can call the mint and destroy functions. The owner of the contract can set the status of the mounter.

### TradingExtension Contract

The contract implements the auxiliary functions of the trading contract, including setting and managing the node address, the minimum position size, updating position information, verifying price signatures, calculating prices with spreads, obtaining the cumulative interest on assets and related parameters, verifying the validity of the transaction, and so on, a series of trading needs to be used in a series of parameters and the implementation of commonly used functions. The configuration parameters can only be modified by the administrator.

### Trading Contract

The contract implements a basic trading system that realizes trading-related functions. It includes operations such as increasing positions, decreasing positions, liquidating positions and updating take profit and stop loss prices.

### OrderBook Contract

This contract mainly realizes the order management related functions for managing the system to manage the orders to add and reduce positions, including the operations of creating, updating, canceling and executing orders. It realizes order execution and position management through interaction with other contracts.

### PairsContract Contract

The contract implements a series of functions for managing asset information, positions, funding rates, etc. of a trading pair. The contract specifically implements functions such as adding and updating asset information, updating funding rates, editing long and short positions, setting user position limits, setting maximum asset positions and user position limits, and suspending asset trading. However, the main logic in this contract is qualified by privileged roles, and the contract point functions are called by other contracts in the overall project.

### Router Contract

The contract implements a router contract that provides the user with the ability to increase positions, decrease positions and update take profit and stop loss prices. It also implements an interface in the form of a plug-in that provides additional front-end operations, which is restricted using a whitelist and the restrictions are managed by an administrator.

### BasicRouter Contract

This contract is an abstract contract that implements the functions of transaction information verification and price signature verification required for routing operations, and provides the functions of obtaining the maximum price and minimum price. The main functions are also accomplished through indirect calls to TradingExtension and other contracts.

### PositionManager Contract

The contract implements a function to manage the user's positions on the platform, including position liquidation, take-profit and stop-loss execution, and order execution. It also includes auxiliary functions such as generating unique identifiers for batch trades, obtaining maximum take-profit prices, and more.

### ReferralStorage Contract

This contract is a system used to manage the relationship between the referrer and the trader as well as their rewards, and it contains information such as the referrer's share of the discount, the referrer's rank, and the referral code. This contract is mainly used to process the rebates and discounts of the referrals and to manage the ownership of the referral codes. The administrator can set the referrer level and set the privileged role of Handler, which sets the trader's referral code.

### ReferralReader Contract

This contract is an implementation of a tool contract for fetching structured data on the chain for batch reading of referral code owner information from `ReferralStorage` contracts.

### RewardRouter Contract

The contract provides users with the ability to pledge LP tokens for fees, release LP tokens from pledge, redeem RUSD tokens, withdraw pledged tokens, and collect fee rewards. The contract also manages a privileged role, UnstakeKeeper, and a minimum execution fee parameter charged by that role, both controlled by the contract administrator.

### RewardTracker Contract

The contract is a contract for tracking and distributing rewards and mainly implements an ERC20 token-based collateralization and redemption mechanism, as well as the calculation and distribution of rewards. In addition to the main collateralization, redemption, transfer, authorization, reward collection and reward update functions, some setup functions are provided for some configuration.

### RewardDistributor Contract

This contract is mainly used for distributing rewards. This contract mainly takes rewards from the vault and distributes them to RewardTracker contracts at regular intervals.

### RewardDivider Contract

The contract is an abstract contract with an internal implementation of a list of special privileges and a list of reward recipients, through which the owner of the contract can manage the recipients and processors of rewards.

### FeeRewardDivider Contract

This contract is a specific implementation of the RewardDivider contract, and its logic is mainly to distribute rewards according to the reward distribution ratio set by the administrator. According to its code comments, we know that the rewards are distributed proportionally to the following roles: ROP repurchase, ROP holders, RUSD pledged LPs, fee-tracking LPs, rewards dispensers, and teams. However, the actual distribution needs to be handled according to the admin settings.

### InsuranceFund Contract

The contract is designed as an insurance fund, which can be initialized, the address of the insurance fund can be set, and compensation can be paid from the insurance fund. Called by a Trading contract.

### ProfitRewardDivider Contract

This contract is a concrete implementation of the RewardDivider contract, which mainly implements the distribution of rewards to three recipients based on the Tp Token price range of a given prophecy machine: the Insurance Fund, the ROP Buyback, and the LP Vault.

### ChainUtils Contract

The contract implements a simple library of tools that include functions such as getting on-chain block timestamps, block heights, and verifying signatures.

### FaucetToken Contract

The contract implements a basic ERC20 token contract with added tap functionality that can be used to issue and manage tokens and allow users to pick up tokens through the tap.

### SigVerify Contract

The contract implements a simple library of signature verification tools that provide multiple methods for verifying signatures and can be used to verify signatures on price data.

### TimeVerify Contract

The contract implements a simple library of time verification tools that allow users to get the current block time and can record and update time information by setting the latest time and delaying setting the latest time.

### WETH9 Contract

The contract is an implementation of an Ether wrapped Ether (Wrapped Ether, or WETH) contract that provides features such as deposits, withdrawals, authorizations, transfers, and support for authorization using signature verification.

## 3.2 Contract details

**StableToken Contract**

| Name | Parameter | Attributes |
|---|---|---|
| burnFrom | address account<br>uint256 amount | onlyMinter |
| mintFor | address account<br>uint256 amount | onlyMinter |
| setMinter | address _address<br>bool _status | onlyOwner |

**Config Contract**

| Name | Parameter | Attributes |
|---|---|---|
| initialize | address _weth | public |
| getAddr | AddressType _type | external |
| getUint | UintType _type | external |
| setAddr | AddressType _type<br>address _addr | onlyOwner |
| setUint | UintType _type<br>uint256 _value | onlyOwner |
| setVault | address _vault<br>bool _allowed | onlyOwner |
| isAddr | AddressType _type<br>address _addr | external |
| checkVaultAndMargin | address _vault<br>address _token | external |

**StableVault Contract**

| Name | Parameter | Attributes |
|---|---|---|
| initialize | address _cfg<br>address _stable | public |
| setRewardDivider | address _addr<br>bool _isOk | onlyOwner |
| getLpPrice | none | external |
| setLpPrice | uint256 _lpPrice | external |
| _setLpPrice | uint256 _lpPrice | private |
| setLpPriceUpdater | address _addr<br>bool _isOk | onlyOwner |
| setTokenWatcher | address _addr<br>bool _isOk | onlyOwner |
| setStakeFeeBasisPoints | uint256 _stakeFeeBasisPoints<br>uint256 _unstakeFeeBasisPoints | onlyOwner |
| setPrivateLiquidityMode | bool _isPrivateLiquidityMode | onlyOwner |
| setLiquidityRouter | address _addr<br>bool _isOk | onlyOwner |
| setWithdrawDelay | uint256 _withdrawDelay | onlyOwner |
| setMinUnderwaterLpPrice | uint256 _minUnderwaterLpPrice | onlyOwner |
| tokenAmountToStableAmount | address _token<br>uint256 _tAmount | public |
| stableAmountToTokenAmount | address _token<br>uint256 _uAmount | public |
| _withdraw | address _token<br>address _to<br>uint256 _amount | private |
| _defaultToken | none | private |
| transferOut | address _token<br>address _to<br>uint256 _uAmount | external |

| | | |
|---|---|---|
| _updateRewardBalance | address _addr<br>int256 _amount | private |
| transferIn | address _from<br>address _token<br>uint256 _tAmount<br>bool _toReward | public |
| stableRedeem | address _token<br>address _to<br>uint256 _uAmount | external |
| listToken | address _token | onlyOwner |
| delistToken | address _token | onlyOwner |
| pauseToken | address _token<br>bool _isPaused | external |
| toShare | uint256 _amount | public |
| fromShare | uint256 _share | public |
| _checkAndSetPrice | PriceData _priceData<br>bytes _signature | private |
| depositFor | PriceData _priceData<br>bytes _signature<br>address _from<br>address _token<br>uint256 _amount<br>address _receiver | public |
| _getDelay | none | internal |
| _checkLiqudityRouter | none | internal |
| requestWithdraw | PriceData _priceData<br>bytes _signature<br>uint256 _lpAmount<br>address _outputToken<br>address _receiver | external |
| _popUserWithdraw | address _account<br>uint256 _reqId | private |
| claimWithdraw | PriceData _priceData<br>bytes _signature<br>address _owner<br>uint256 _reqId | external |

| | | |
|---|---|---|
| payoutReward | address _to<br>uint256 _amount | external |
| paybackReward | address _from<br>uint256 _amount | external |
| claimReward | address _token<br>uint256 _amount | external |
| freeze | uint256 _amount | external |
| unfreeze | uint256 _amount | external |
| vaultValue | none | external |

**PositionManager Contract**

| Name | Parameter | Attributes |
|---|---|---|
| initialize | address _cfg | public |
| setOrderKeeper | address _account<br>bool _isActive | onlyOwner |
| setPositionKeeper | address _account<br>bool _isActive | onlyOwner |
| uuid | address _account<br>uint256 _index | public |
| _liquidatePosition | PriceData _priceData<br>bytes _signature<br>address _account<br>uint256 _assetId<br>bool _isLong<br>address _feeReceiver<br>bytes32 _uuid | private |
| liquidatePosition | PriceData _priceData<br>bytes _signature<br>address _account<br>uint256 _assetId<br>bool _isLong | onlyPosition Keeper |

| | | |
|---|---|---|
| | address _feeReceiver | |
| _getMaxTpPrice | ITrading.Position _pos | internal |
| _executePositionTpSl | PriceData _priceData<br>bytes _signature<br>address _account<br>uint256 _assetId<br>bool _isLong<br>address _feeReceiver<br>bytes32 _uuid | private |
| executePositionTpSl | PriceData _priceData<br>bytes _signature<br>address _account<br>uint256 _assetId<br>bool _isLong<br>address _feeReceiver | onlyPosition Keeper |
| _executeIncreaseOrder | PriceData _priceData<br>bytes _signature<br>address _account<br>uint256 _orderIndex<br>address payable _feeReceiver<br>bytes32 _uuid | private |
| executeIncreaseOrder | PriceData _priceData<br>bytes _signature<br>address _account<br>uint256 _orderIndex<br>address payable _feeReceiver | onlyOrderKeeper |
| _executeDecreaseOrder | PriceData _priceData<br>bytes _signature<br>address _account<br>uint256 _orderIndex<br>address payable _feeReceiver<br>bytes32 _uuid | private |
| executeDecreaseOrder | PriceData _priceData<br>bytes _signature<br>address _account<br>uint256 _orderIndex<br>address payable _feeReceiver | onlyOrderKeeper |

**OrderBook Contract**

| Name | Parameter | Attributes |
|---|---|---|
| initialize | address _cfg<br>uint256 _minExecutionFee | public |
| setMinExecutionFee | uint256 _minExecutionFee | onlyOwner |
| validatePositionOrderPrice | bool _triggerAboveThreshold<br>uint256 _triggerPrice<br>uint256 _assetId<br>uint256 _indexPrice<br>bool _useMaxPrice<br>bool _raise | public |
| getDecreaseOrder | address _account<br>uint256 _orderIndex | public |
| getIncreaseOrder | address _account<br>uint256 _orderIndex | public |
| getTradeInfo | uint256 _tradeInfoId | public |
| _pairs | none | internal |
| _updateOi | address _account<br>uint256 _assetId<br>address _vault<br>bool _isLong<br>uint256 _sizeDelta<br>bool _isIncrease | private |
| createIncreaseOrder | TradeInfo _tradeInfo<br>uint256 _triggerPrice<br>bool _triggerAboveThreshold<br>bytes32 _referralCode | external |
| _createIncreaseOrder | address _account<br>uint256 _tradeInfoIndex<br>uint256 _triggerPrice<br>bool _triggerAboveThreshold<br>uint256 _executionFee | private |
| updateIncreaseOrder | uint256 _orderIndex<br>uint256 _sizeDelta<br>uint256 _triggerPrice | external |

| | | |
|---|---|---|
| | bool _triggerAboveThreshold | |
| cancelIncreaseOrder | uint256 _orderIndex | public |
| cancelIncreaseOrderByExecutor | address _account<br>uint256 _orderIndex<br>bytes32 _uuid | onlyExecutor |
| _cancelIncreaseOrder | address _account<br>uint256 _orderIndex<br>bytes32 _uuid | private |
| executeIncreaseOrder | uint256 _price<br>address _account<br>uint256 _orderIndex<br>address payable _feeReceiver<br>bytes32 _uuid | onlyExecutor |
| getPositionKey | address _account<br>uint256 _indexAsset<br>bool _isLong | public |
| createDecreaseOrder | TradeInfo _tradeInfo<br>uint256 _triggerPrice<br>bool _triggerAboveThreshold<br>address _outputToken | external |
| _addDecreaseOrderToList | address _account<br>uint256 _assetId<br>bool _isLong<br>uint256 _orderIndex | private |
| _delDecreaseOrderFromList | address _account<br>uint256 _assetId<br>bool _isLong<br>uint256 _orderIndex | private |
| _createDecreaseOrder | address _account<br>uint256 _tradeInfoIndex<br>uint256 _triggerPrice<br>bool _triggerAboveThreshold<br>address _outputToken<br>uint256 _executionFee | private |
| updateDecreaseOrder | uint256 _orderIndex<br>uint256 _marginDelta<br>uint256 _sizeDelta | external |

| | | |
|---|---|---|
| | uint256 _triggerPrice<br>bool _triggerAboveThreshold | |
| executeDecreaseOrder | uint256 _price<br>address _address<br>uint256 _orderIndex<br>address payable _feeReceiver<br>bytes32 _uuid | onlyExecutor |
| cancelDecreaseOrder | uint256 _orderIndex | public |
| cancelDecreaseOrderByExecutor | address _account<br>uint256 _orderIndex<br>bytes32 _uuid | onlyExecutor |
| _cancelDecreaseOrder | address _account<br>uint256 _orderIndex<br>bytes32 _uuid | internal |
| executeDecreasePosition | PosInfo _posInfo<br>address _outputToken<br>address _executor | onlyExecutor |
| cleanDecreaseOrderList | address _account<br>uint256 _assetId<br>bool _isLong<br>bytes32 _uuid | external |
| setExecutor | address _executor<br>bool _isExecutor | onlyOwner |
| _transferOutETH | uint256 _amountOut<br>address payable _receiver | private |

### BasicRouter Contract

| Name | Parameter | Attributes |
|---|---|---|
| __BasicRouter_init | address _cfg | internal |
| _validateAssetInfo | TradeInfo _tradeInfo | internal |
| _verifyPrice | uint256 _assetId<br>PriceData _priceData<br>bytes _signature | internal |
| _getMaxPrice | uint256 _assetId<br>uint256 _indexPrice | internal |
| _getMinPrice | uint256 _assetId<br>uint256 _indexPrice | internal |
| _tradeExt | none | internal |
| _trading | none | internal |
| _router | none | internal |
| _orderbook | none | internal |
| _referral | none | internal |

### Router Contract

| Name | Parameter | Attributes |
|---|---|---|
| initialize | address _cfg | public |
| addPlugin | address _plugin | onlyOwner |
| removePlugin | address _plugin | onlyOwner |
| approvePlugin | address _plugin | external |
| denyPlugin | address _plugin | external |
| pluginIncreasePosition | PosInfo _posInfo | external |
| pluginDecreasePosition | PosInfo _posInfo<br>address _outputToken<br>address _executor | external |
| increasePosition | TradeInfo _tradeInfo<br>PriceData _priceData<br>bytes _signature | external |

| | | |
|---|---|---|
| | bytes32 _referralCode | |
| decreasePosition | TradeInfo _tradeInfo<br>PriceData _priceData<br>bytes _signature<br>address _outputToken | external |
| _checkSl | ITrading.Position _pos<br>uint256 _slPrice<br>uint256 _price | internal |
| _checkTp | ITrading.Position _pos<br>uint256 _tpPrice<br>uint256 _price | internal |
| updateTpSl | TpSlInfo _tpSlInfo<br>PriceData _priceData<br>bytes _signature | external |
| _validatePlugin | address _account | private |

**PairsContract Contract**

| Name | Parameter | Attributes |
|---|---|---|
| initialize | address _cfg<br>uint256 _maxBaseFudingRate<br>uint256 _fundingInterval<br>uint256 _maxLoss<br>uint256 _maxProfit | public |
| idToAsset | uint256 _assetId | public |
| idToOi | uint256 _assetId<br>address _vault | public |
| getUserOrderOi | address _user<br>uint256 _assetId | public |
| getOiInfo | uint256 _assetId<br>address _vault | public |
| getUserOiInfo | uint256 _assetId<br>address _user | public |
| getAccInterest | uint256 _assetId | public |

| | | |
|---|---|---|
| | address _vault<br>bool _isLong | |
| getSpread | uint256 _assetId | public |
| setFundingInterval | uint256 _fundingInterval | onlyOwner |
| getUserOi | uint256 _assetId<br>address _user | external |
| updateAssetLeverage | uint256 _assetId<br>uint256 _minLeverage<br>uint256 _maxLeverage | onlyOwner |
| setAssetBaseFunding Rate | uint256 _assetId<br>uint256 _baseFundingRate | onlyOwner |
| updateSpread | uint256 _assetId<br>uint256 _spread | onlyOwner |
| updateMaxTpSl | uint256 _assetId<br>uint256 _maxProfit<br>uint256 _maxLoss | onlyOwner |
| pauseAsset | uint256 _assetId<br>bool _isPaused | onlyOwner |
| setMaxBaseFundingRa te | uint256 _maxBaseFundingRate | onlyOwner |
| setUserMaxMinOi | uint256 _assetId<br>address _user<br>uint256 _maxOi<br>uint256 _minOi | onlyOwner |
| updateOrderOi | address _account<br>uint256 _assetId<br>address _vault<br>bool _isLong<br>uint256 _sizeDelta<br>bool _isIncrease | onlyOrderbo ok |
| modifyLongOi | address _trader<br>uint256 _assetId<br>address _vault<br>bool _isIncrease<br>uint256 _amount | onlyTradeExt |
| modifyShortOi | address _trader | onlyTradeExt |

| | | |
|---|---|---|
| | uint256 _assetId<br>address _vault<br>bool _isIncrease<br>uint256 _amount | |
| updateFundingFee | uint256 _assetId<br>address _vault | external |

**Trading Contract**

| Name | Parameter | Attributes |
|---|---|---|
| initialize | address _cfg<br>uint256 _openingFeeBasisPoints<br>uint256 _closingFeeBasisPoints<br>uint256 _hourlyFeeBasisPoints<br>uint256 _liquidationFeeUsd | public |
| setInPrivateLiquidationMode | bool _inPrivateLiquidationMode | onlyOwner |
| setLiquidator | address _liquidator<br>bool _isActive | onlyOwner |
| setFees | uint256 _openingFeeBasisPoints<br>uint256 _closingFeeBasisPoints<br>uint256 _hourlyFeeBasisPoints<br>uint256 _liquidationFeeUsd<br>uint256 _minProfitTime | onlyOwner |
| addRouter | address _router | external |
| removeRouter | address _router | external |
| _getPrice | PosInfo _posInfo<br>bool _increased | internal |
| _getPosKey | PosInfo _posInfo | internal |
| getOpeningFeeBasisPoints | uint256 _indexAsset | public |
| getClosingFeeBasisPoints | uint256 _indexAsset | public |
| getHourlyFeeBasisPoints | uint256 _indexAsset | public |

| | | |
|---|---|---|
| _collectIncreasePositionFee | Position _pos<br>PosInfo _posInfo | private |
| _updateFundingFee | uint256 _assetId<br>address _vault | private |
| increasePosition | PosInfo _posInfo | external |
| _emitIncreasePosition | bytes32 _key<br>PosInfo _posInfo<br>uint256 _price<br>uint256 _fee<br>uint256 _oriMargin | internal |
| decreasePosition | PosInfo _posInfo<br>address _outputToken<br>address _executor | external |
| _emitDecreasePosition | bytes32 _key<br>PosInfo _posInfo<br>uint256 _price<br>uint256 _fee<br>uint256 _oriMargin | internal |
| _updateOi | PosInfo _posInfo<br>bool _isIncrease | private |
| _decreasePosition | PosInfo _posInfo<br>address _outputToken<br>address _executor<br>bool _isLiquidate | internal |
| _emitLiquidatePosition | bytes32 _key<br>PosInfo _posInfo<br>uint256 _price<br>Position _oriPosition<br>DePosOut _outInfo | internal |
| liquidatePosition | PosInfo _posInfo<br>address _feeReceiver | external |
| validateLiquidation | PosInfo _posInfo<br>bool _raise | public |
| updateTpSl | TpSlInfo _tpSlInfo<br>address _account | external |
| getMaxPrice | uint256 _indexAsset | public |

| | | |
|---|---|---|
| | uint256 _indexPrice | |
| getMinPrice | uint256 _indexAsset<br>uint256 _indexPrice | public |
| getPosition | address _account<br>uint256 _indexAsset<br>bool _isLong | public |
| getPositionKey | address _account<br>uint256 _indexAsset<br>bool _isLong | public |
| getPositionLeverage | address _account<br>uint256 _indexAsset<br>bool _isLong | public |
| getNextAveragePrice | Position _pos<br>uint256 _nextPrice<br>uint256 _sizeDelta | public |
| getPositionPnl | address _account<br>uint256 _indexAsset<br>uint256 _indexPrice<br>bool _isLong | public |
| getFundingFee | Position _pos | public |
| getOpeningFee | uint256 _assetId<br>uint256 _sizeDelta<br>uint256 _discount | public |
| getClosingFee | uint256 _assetId<br>uint256 _sizeDelta<br>uint256 _discount | public |
| getHourlyFee | Position _pos | public |
| getPositionInfo | address _account<br>uint256 _indexAsset<br>bool _isLong | external |
| _distributeDecreasePositionFee | PosInfo _posInfo<br>uint256 _fee | private |
| getPnl | Position _pos<br>uint256 _indexPrice | public |
| _getPnl | Position _pos<br>uint256 _indexPrice | private |

| | | |
|---|---|---|
| | bool _includeHourly | |
| getDelta | Position _pos<br>uint256 _indexPrice | public |
| _getAdjustedDelta | Position _pos<br>PosInfo _posInfo | private |
| _reduceMargin | PosInfo _posInfo<br>address _executor | private |
| _emitUpdatePnl | bytes32 _key<br>int256 adjustDelta<br>bytes32 _uuid | private |
| _validatePosition | uint256 _size<br>uint256 _margin | private |
| _validateRouter | address _account | private |
| _transferOut | address _vault<br>address _token<br>uint256 _amount<br>address _receiver | private |
| _tradeExt | none | internal |
| _referral | none | internal |
| _orderbook | none | internal |
| _feeDivider | none | internal |
| _profitDivider | none | internal |
| _insuranceFund | none | internal |
| _routerAddr | none | internal |

**TradingExtension Contract**

| Name | Parameter | Attributes |
|---|---|---|
| initialize | address _cfg<br>uint256 _validSignatureTime | public |
| minPos | address _vault | external |
| modifyShortOi | address _trader<br>uint256 _asset<br>address _vault<br>bool _isIncrease<br>uint256 _size | onlyTrade |
| modifyLongOi | address _trader<br>uint256 _asset<br>address _vault<br>bool _isIncrease<br>uint256 _size | onlyTrade |
| setFutureTimeDelta | uint256 _time | onlyOwner |
| verifyPrice | uint256 _assetId<br>PriceData _priceData<br>bytes _signature | public |
| verifyTest | uint256 _assetId<br>PriceData _priceData<br>bytes _signature | external |
| getBlockTimestamp | none | external |
| getMaxPrice | uint256 _assetId<br>uint256 _indexPrice | external |
| getMinPrice | uint256 _assetId<br>uint256 _indexPrice | external |
| getAccInterest | uint256 _assetId<br>address _vault<br>bool _isLong | external |
| getMaxLeverage | uint256 _assetId | external |
| getMaxLoss | uint256 _assetId | external |
| getMaxProfit | uint256 _assetId | external |
| getFeeBasisPoints | uint256 _assetId | external |

| | | |
|---|---|---|
| isPaused | none | external |
| validateTrade | uint256 _asset | external |
| setValidSignatureTime | uint256 _time | onlyOwner |
| setNode | address _node<br>bool _isNode | onlyOwner |
| setMinPositionSize | address _vault<br>uint256 _min | onlyOwner |
| setPaused | bool _paused | onlyOwner |
| _pairs | none | internal |

**ChainUtils Contract**

| Name | Parameter | Attributes |
|---|---|---|
| getTime | none | internal |
| getBlock | none | internal |
| isValidSignature | address _address<br>bytes32 _hash<br>bytes _signature | internal |

**RewardRouter Contract**

| Name | Parameter | Attributes |
|---|---|---|
| initialize | address _cfg<br>address _rusd<br>address _feeLpTracker | external |
| setUnstakeKeeper | address _keeper<br>bool _isKeeper | onlyOwner |

| | | |
|---|---|---|
| setMinUnstakeFee | uint256 _minUnstakeFee | onlyOwner |
| _feeDivider | none | internal |
| mintAndStakeLp | PriceData _priceData<br>bytes _signature<br>address _token<br>uint256 _amount<br>uint256 _minLp | external |
| unstakeAndRedeemLp | PriceData _priceData<br>bytes _signature<br>address _outputToken<br>uint256 _feeLpAmount | external |
| claimWithdraw | PriceData _priceData<br>bytes _signature<br>uint256 _reqId<br>address _account | external |
| claim | none | external |
| claimReward | address _outputToken | external |
| _transferOutETH | uint256 _amountOut<br>address payable _receiver | private |

**RewardTracker Contract**

| Name | Parameter | Attributes |
|---|---|---|
| initialize | string _name<br>string _symbol | external |
| setDepositToken | address _depositToken<br>bool _isDepositToken | onlyOwner |
| setDistributor | address _distributor | onlyOwner |
| setInPrivateTransferMode | bool _inPrivateTransferMode | onlyOwner |

| | | |
|---|---|---|
| setInPrivateStakingMode | bool _inPrivateStakingMode | onlyOwner |
| setInPrivateClaimingMode | bool _inPrivateClaimingMode | onlyOwner |
| setHandler | address _handler<br>bool _isActive | onlyOwner |
| balanceOf | address _account | external |
| stake | address _depositToken<br>uint256 _amount | external |
| stakeForAccount | address _fundingAccount<br>address _account<br>address _depositToken<br>uint256 _amount | external |
| unstake | address _depositToken<br>uint256 _amount | external |
| unstakeForAccount | address _account<br>address _depositToken<br>uint256 _amount<br>address _receiver | external |
| transfer | address _recipient<br>uint256 _amount | external |
| allowance | address _owner<br>address _spender | external |
| approve | address _spender<br>uint256 _amount | external |
| transferFrom | address _sender<br>address _recipient<br>uint256 _amount | external |
| tokensPerInterval | none | external |
| updateRewards | none | external |
| claim | address _receiver | external |
| claimForAccount | address _account<br>address _receiver | external |
| claimable | address _account | public |
| rewardToken | none | public |
| _claim | address _account | private |

| | | |
|---|---|---|
| | address _receiver | |
| _mint | address _account<br>uint256 _amount | internal |
| _burn | address _account<br>uint256 _amount | internal |
| _transfer | address _sender<br>address _recipient<br>uint256 _amount | private |
| _approve | address _owner<br>address _spender<br>uint256 _amount | private |
| _validateHandler | none | private |
| _stake | address _fundingAccount<br>address _account<br>address _depositToken<br>uint256 _amount | private |
| _unstake | address _account<br>address _depositToken<br>uint256 _amount<br>address _receiver | private |
| _updateRewards | address _account | private |

**RewardDistributor Contract**

| Name | Parameter | Attributes |
|---|---|---|
| initialize | address _vault<br>address _rewardTracker<br>address _rewardDivider | external |
| rewardToken | none | public |
| setRewardInfo | address _rewardDivider<br>uint256 _rewardFetchInterval | onlyOwner |
| _setTokensPerInterval | uint256 _amount | private |

| | | |
|---|---|---|
| pendingRewards | none | public |
| fetchRewardManual | none | external |
| distribute | none | external |

**ReferralStorage Contract**

| Name | Parameter | Attributes |
|---|---|---|
| initialize | address _cfg<br>uint256 _defaultRebate<br>uint256 _defaultDiscountShare | external |
| setPrivateMode | bool _isPrivateMode | onlyOwner |
| setHandler | address _handler<br>bool _isActive | onlyOwner |
| addTier | uint256 _totalRebate<br>uint256 _discountShare<br>uint256 _subRate | onlyOwner |
| setTier | uint256 _tierId<br>uint256 _totalRebate<br>uint256 _discountShare<br>uint256 _subRate | onlyOwner |
| setReferrerTier | address _referrer<br>uint256 _tierId | onlyOwner |
| setReferrerDiscountShareFromWl | bytes _signature<br>address _whitelistSigner<br>uint256 _discountShare | external |
| setReferrerDiscountShare | uint256 _discountShare | external |
| setTraderReferralCode | address _account<br>bytes32 _code | external |
| registerCode | bytes32 _code | external |
| registerCodeFromWl | bytes _signature<br>address _whitelistSigner | external |

| | | |
|---|---|---|
| | bytes32 _code | |
| acceptReferral | bytes32 _code | external |
| setCodeOwner | bytes32 _code<br>address _newAccount | external |
| getTraderReferralInfo | address _account | external |
| _setTraderReferralCode | address _account<br>bytes32 _code | private |
| distribute | address _referrer<br>address _trader<br>address _vault<br>uint256 _amount<br>bytes32 _uuid | external |

**ReferralReader Contract**

| Name | Parameter | Attributes |
|---|---|---|
| getCodeOwners | IReferralStorage _referralStorage<br>bytes32[] _codes | public |

**FeeRewardDivider Contract**

| Name | Parameter | Attributes |
|---|---|---|
| initialize | address _cfg | external |
| _referral | none | internal |
| distribute | address _account<br>address _vault<br>uint256 _amount<br>bytes32 _uuid | onlyHandler |

### ProfitRewardDivider Contract

| Name | Parameter | Attributes |
|---|---|---|
| initialize | address _cfg<br>address _oracle | external |
| _getPercents | none | private |
| distribute | address _account<br>address _vault<br>uint256 _amount<br>bytes32 _uuid | onlyHandler |

### RewardDivider Contract

| Name | Parameter | Attributes |
|---|---|---|
| __RewardDivider_init | address _cfg | internal |
| setHandler | address _handler<br>bool _enable | onlyOwner |
| addRecipient | address _account<br>uint256 _percent | onlyOwner |
| delRecipient | address _account | onlyOwner |

### InsuranceFund Contract

| Name | Parameter | Attributes |
|---|---|---|
| initialize | address _cfg<br>address _insuranceFund | external |
| setInsuranceFund | address _insuranceFund | onlyOwner |
| compensate | address _vault<br>uint256 _amount | external |

### TimeVerify Contract

| Name | Parameter | Attributes |
|---|---|---|
| getTime | none | public |
| setTime | none | external |
| setTimeWithDelay | uint256 _delay | external |

### SigVerify Contract

| Name | Parameter | Attributes |
|---|---|---|
| verify | PriceData _info<br>bytes _signature | public |
| verify1 | address _address<br>bytes32 _hash<br>bytes _signature | public |
| verify2 | address _address<br>bytes32 _hash<br>bytes _signature | public |
| verify3 | address _address<br>bytes32 _hash<br>bytes _signature | public |
| verify4 | PriceData _info<br>bytes _signature | public |
| verifyTest3 | address _address<br>bytes32 _hash<br>bytes _signature | external |
| verifyTest4 | PriceData _info<br>bytes _signature | external |

**FaucetToken Contract**

| Name | Parameter | Attributes |
|---|---|---|
| mint | address account<br>uint256 amount | public |
| enableFaucet | none | public |
| disableFaucet | none | public |
| setDropletAmount | uint256 dropletAmount | public |
| claimDroplet | none | public |
| name | none | public |
| symbol | none | public |
| decimals | none | public |
| totalSupply | none | public |
| balanceOf | address account | public |
| transfer | address recipient<br>uint256 amount | public |
| allowance | address owner<br>address spender | public |
| approve | address spender<br>uint256 amount | public |
| transferFrom | address sender<br>address recipient<br>uint256 amount | public |
| increaseAllowance | address spender<br>uint256 addedValue | public |
| decreaseAllowance | address spender<br>uint256 subtractedValue | public |
| _transfer | address sender<br>address recipient<br>uint256 amount | internal |
| _mint | address account<br>uint256 amount | internal |
| _burn | address account | internal |

| | | |
|---|---|---|
| | uint256 amount | |
| _approve | address owner<br>address spender<br>uint256 amount | internal |
| _beforeTokenTransfer | address from<br>address to<br>uint256 amount | internal |
| _msgSender | none | internal |

**WETH Contract**

| Name | Parameter | Attributes |
|---|---|---|
| supportsInterface | bytes4 interfaceID | external |
| DOMAIN_SEPARATOR | none | public |
| _computeDomainSeparator | none | private |
| deposit | none | public |
| withdraw | uint value | external |
| totalSupply | none | external |
| approve | address spender<br>uint value | external |
| transfer | address to<br>uint value | external |
| transferFrom | address from<br>address to<br>uint value | external |
| permit | address owner<br>address spender<br>uint value<br>uint deadline<br>uint8 v<br>bytes32 r<br>bytes32 s | external |
| permit2 | address owner | external |

| | | |
|---|---|---|
| | address spender<br>uint value<br>uint deadline<br>bytes signature | |
| _checkSignature | address signer<br>bytes32 hash<br>bytes signature | private |
| _recover | bytes32 hash<br>bytes signature | private |

# 4. Audit details

## 4.1 Findings Summary

| Severity | Found | Resolved | Acknowledged |
|---|---|---|---|
| High | 0 | **0** | 0 |
| Medium | 2 | 1 | 1 |
| Low | 1 | 0 | 1 |
| Info | 3 | 0 | 3 |

## 4.2 Risk distribution

| Name | Risk level | Repair status |
|---|---|---|
| Signature Replay | Medium | Resolved |
| Price Control | Medium | Acknowledged |
| Administrator Permissions | Low | Acknowledged |
| Redundant codes | Info | Acknowledged |
| External Expansion Security | Info | Acknowledged |
| Logic Design Flaw | Info | Acknowledged |
| Variables are updated | No | normal |
| Floating Point and Numeric Precision | No | normal |
| Default visibility | No | normal |
| tx.origin authentication | No | normal |
| Faulty constructor | No | normal |
| Unverified return value | No | normal |
| Insecure random numbers | No | normal |
| Timestamp Dependent | No | normal |
| Transaction order dependency | No | normal |
| Delegatecall | No | normal |
| Denial of Service | No | normal |
| Fake recharge vulnerability | No | normal |
| Short address attack Vulnerability | No | normal |
| Uninitialized storage pointer | No | normal |
| Frozen account bypass | No | normal |

| | | |
|---|---|---|
| Uninitialized | No | normal |
| Reentry attack | No | normal |
| Integer Overflow | No | normal |

## 4.3 Risk audit details

### 4.3.1 Signature Replay

- **Risk description**

Signature replay risk is the risk that under certain circumstances, an attacker can use a valid signature that has been obtained to repeat the execution of a signed transaction or message to perform an unauthorized operation. Without proper protection mechanisms in a contract, an attacker can exploit this vulnerability to perform repeated operations, which may lead to undesirable consequences.

According to the implementation of the contract, the contract relies heavily on the time tolerance mechanism to protect against signature replay attacks, but there is no one-time use restriction on the signature data, and it is still possible for an attacker to replay the signature data before it expires within a short period of time; no obvious points of exploitation have been identified, but the risk is recommended to be fixed.

```
function verifyPrice(
    uint256 _assetId,
    PriceData calldata _priceData,
    bytes calldata _signature
)
    public override view returns (string memory)
{
    uint256 _time = ChainUtils.getTime();
    if (!ChainUtils.isValidSignature(
        _priceData.provider,
        keccak256(abi.encodePacked(_priceData.provider, _priceData.asset,
_priceData.price, _priceData.timestamp)),
        _signature
    )) return "TradingExtension: bad sig";
    if (!isNode[_priceData.provider]) return "TradingExtension: !node";
    if (_assetId != _priceData.asset) return "TradingExtension: bad
asset";
    // _priceData.timestamp < _time - validSignatureTime
    if (_priceData.timestamp + validSignatureTime < _time) return
"TradingExtension: expired sig";
```

```
    // _priceData.timestamp > _time + futureTimeDelta
    if (futureTimeDelta > 0) {
        if (_priceData.timestamp > _time + futureTimeDelta) return
"TradingExtension: bad timestamp";
    }
    if (_priceData.price == 0) return "TradingExtension: bad price";
    return "";
}
```

- **Safety advice**

It is recommended that the following mechanism be added to the signature verificati on mechanism here:

Use nonce: Introduce an incremental nonce value in each transaction or operation to ensure that each transaction or operation can only be executed once. Maintain a no nce counter in the contract and verify that the nonce is correct in each transaction or operation, and add 1 to the in-contract nonce upon successful verification to preven t malicious replay of privileged signatures.

- **Repair Status**

ROLLUP.FINANCE has Resolved.

### 4.3.2 Price Control

- **Risk description**

All locations in the program that involve prices use price data signed with administrator privileges and the source of the data is unknown, so there may be a risk of price manipulation.

```
function increasePosition(
    TradeInfo calldata _tradeInfo,
    PriceData calldata _priceData,
    bytes calldata _signature,
    bytes32 _referralCode
)
    external override
{
    address _account = _msgSender();
    _validateAssetInfo(_tradeInfo);
    // check price
    string memory _error = _verifyPrice(_tradeInfo.assetId, _priceData,
_signature);
    if (bytes(_error).length > 0) {
        emit PriceVerify(_error);
        revert(_error);
    }
    // set referral code
    if (_referralCode != bytes32(0)) {
        if (_referral() != address(0)) {
            IReferralStorage(_referral()).setTraderReferralCode(_account,
_referralCode);
        }
    }
    // margin to this
    IERC20(_tradeInfo.marginAsset).safeTransferFrom(
        _account,
        address(this),
        _tradeInfo.marginDelta
    );
    if (!_isApproved[_tradeInfo.marginAsset]) {
        IERC20(_tradeInfo.marginAsset).approve(_tradeInfo.vault,
type(uint256).max);
        _isApproved[_tradeInfo.marginAsset] = true;
    }
    // then to vault
    uint256 _marginDelta = IStableVault(_tradeInfo.vault).transferIn(
```

```
        _account,
        _tradeInfo.marginAsset,
        _tradeInfo.marginDelta,
        false
    );
    _trading().increasePosition(
        PosInfo(
            _account,
            _tradeInfo.assetId,
            _priceData.price, // still use priceData price
            _marginDelta,
            _tradeInfo.sizeDelta,
            _tradeInfo.isLong,
            _tradeInfo.vault,
            bytes32(0)
        )
    );
}
function decreasePosition(
    TradeInfo calldata _tradeInfo,
    PriceData calldata _priceData,
    bytes calldata _signature,
    address _outputToken
)
    external override
{
    _validateAssetInfo(_tradeInfo);
    // check price
    string memory _error = _verifyPrice(_tradeInfo.assetId, _priceData,
_signature);
    if (bytes(_error).length > 0) {
        emit PriceVerify(_error);
        revert(_error);
    }
    _trading().decreasePosition(
        PosInfo(
            _msgSender(),
            _tradeInfo.assetId,
            _priceData.price,
            _tradeInfo.marginDelta,
            _tradeInfo.sizeDelta,
            _tradeInfo.isLong,
```

```
            _tradeInfo.vault,
            bytes32(0)
        ),
        _outputToken,
        address(0) // non executor
    );
}
```

- **Safety advice**

The source of prices is as transparent as possible and price data is available through trusted sources on the chain wherever possible.

- **Repair Status**

ROLLUP.FINANCE has Acknowledged.

### 4.3.3 Administrator Permissions

- **Risk description**

Contract administrator privileges are people who can change the status of a contract or perform certain sensitive operations. If administrator privileges are abused or hacked, the contract may be subject to several risks. First, a hacker may be able to obtain the administrator's private key, thus hijacking the administrator's account and thus having full control of the contract. The hacker can change the status of the contract or perform illegal operations at will, causing a serious loss of contract assets. Second, the administrator may intentionally or unintentionally disclose his or her private key, causing others to gain access. Hackers may obtain the administrator's private key through social engineering or trickery, and obtain a large amount of sensitive contract information, which can then be used to carry out attacks. Third, administrators may use their privileges to perform improper operations. For example, an administrator may change the status of a contract for financial gain, or change the execution rules of a contract to gain additional control, and so on. These actions may also lead to loss of contract assets or illegitimate domination.

The administrator address in the project has parameters that can arbitrarily restrict whether the main business can operate normally, such as: whether to enable the private clearing mode, etc., as well as the whitelist of the clearer, which may cause an impact on the main business when the administrator address is an EOA address and the administrator address is leaked by hackers' phishing or other means.

```
function setInPrivateLiquidationMode(bool _inPrivateLiquidationMode)
external onlyOwner {
    inPrivateLiquidationMode = _inPrivateLiquidationMode;
}
function setLiquidator(address _liquidator, bool _isActive) external
onlyOwner {
    isLiquidator[_liquidator] = _isActive;
}
function liquidatePosition(
    PosInfo calldata _posInfo,
    address _feeReceiver
) external override nonReentrant {
```

```
    if (inPrivateLiquidationMode) {
        if (!isLiquidator[_msgSender()]) revert("Trading: not
liquidator");
    }
    bytes32 key = _getPosKey(_posInfo);
    Position memory position = positions[key];
    if (position.size == 0) revert("Trading: position not found");
    if (position.size != _posInfo.sizeDelta) revert("Trading: liquidate
size not full");
    bool isLiquidatable = validateLiquidation(_posInfo, false);
    if (isLiquidatable == false)  revert("Trading: position not
liquidatable");
    // decrease position now
    DePosOut memory _outInfo = _decreasePosition(_posInfo, address(0),
_feeReceiver, true);
    uint256 markPrice = _getPrice(_posInfo, false);
    _emitLiquidatePosition(key, _posInfo, markPrice, position, _outInfo);
    delete positions[key];
}
```

- **Safety advice**

Avoid using the EOA address as the administrator privilege owner when the contract goes live, and use multi-signature contracts or multi-signature wallets to manage the relevant special privilege operations.

- **Repair Status**

ROLLUP.FINANCE has Acknowledged.

### 4.3.4 Redundant codes

- **Risk description**

Code unrelated to the business may result in unnecessary processing fees.

The existence of a secondary wrapper function for global configuration acquisition that is not used in the contract may result in unnecessary contract deployment fees.

```
function _referral() internal view returns (address) {
    return cfg.getAddr(AddressType.Referral);
}
```

- **Safety advice**

If more code redundancy does exist, it is recommended that it be removed

- **Repair Status**

ROLLUP.FINANCE has Acknowledged.

### 4.3.5 External Expansion Security

- **Risk description**

There is an external expansion interface in the contract, which allows pluginIncreasePosition and pluginDecreasePosition functions to make external contract calls, so the plugins added in the future need to strictly carry out security checks to avoid mutual calls at the contract level, which may lead to exploitation points.

```
function addPlugin(address _plugin) external onlyOwner {
    plugins[_plugin] = true;
}
function removePlugin(address _plugin) external onlyOwner {
    plugins[_plugin] = false;
}
function approvePlugin(address _plugin) external {
    approvedPlugins[msg.sender][_plugin] = true;
}
function denyPlugin(address _plugin) external {
    approvedPlugins[msg.sender][_plugin] = false;
```

```
}
/**
 * @dev plugin increase position
 * redirectly to trading with checking plugin approved
 */
function pluginIncreasePosition(
    PosInfo calldata _posInfo
) external override {
    _validatePlugin(_posInfo.account);
    _trading().increasePosition(_posInfo);
}
/**
 * @dev plugin decrease position
 * redirectly to trading with checking plugin approved
 */
function pluginDecreasePosition(
    PosInfo calldata _posInfo,
    address _outputToken,
    address _executor
) external override returns (uint256) {
    _validatePlugin(_posInfo.account);
    return _trading().decreasePosition(
        _posInfo,
        _outputToken,
        _executor
    );
}
```

- **Safety advice**

This issue requires additional security checks on the plugin along with the project as a whole to avoid suspicious or backdoor plugins being added to the whitelist, and administrator privileges here should also be managed using multi-signature wallets or multi-signature contracts, avoiding the use of EOA addresses for management.

- **Repair Status**

ROLLUP.FINANCE has Acknowledged.

### 4.3.6 Logic Design Flaw

- **Risk Description**

In smart contracts, developers design special features for their contracts intended to stabilize the market value of tokens or the life of the project and increase the highlight of the project, however, the more complex the system, the more likely it is to have the possibility of errors. It is in these logic and functions that a minor mistake can lead to serious depasstions from the whole logic and expectations, leaving fatal hidden dangers, such as errors in logic judgment, functional implementation and design and so on.

The data is not verified when creating the cashout request, so that once you have the correct price signature, you can forge a cashout request from any user. Moreover, there is a problem that in the requestWithdraw and claimWithdraw functions, since requestWithdraw can be created by any user for any other user, it is possible to execute a malicious withdrawal request created by another user, and thus the executor spends Lp funds but does not withdraw them to the address of the user who maliciously initiated the request. The executor spends Lp funds but does not withdraw them to the address of the user who maliciously initiated the request.

```
function requestWithdraw(
    PriceData calldata _priceData,
    bytes calldata _signature,
    uint256 _lpAmount,
    address _outputToken,
    address _receiver
) external override returns (uint256) {
    _checkLiqudityRouter();
    // verify price data, RUSD asset id is 0
    _checkAndSetPrice(_priceData, _signature);
    // check output token is allowed token
    if (!allowed[_outputToken]) revert("StableVault: token not listed");
    uint256 _reqId = allWithdraws.length;
    allWithdraws.push(
        Withdraw({
            outputToken: _outputToken,
            lpAmount: _lpAmount,
            amount: 0,
            priceAtReq: _priceData.price,
            releasedAt:  ChainUtils.getTime() + _getDelay(),
```

```
            priceAtClaim: 0
        })
    );
    userWithdraws[_receiver].push(_reqId);
    emit WithdrawRequest(_receiver, _reqId, _outputToken, _lpAmount,
allWithdraws[_reqId].releasedAt);
    return _reqId;
}
function claimWithdraw(
    PriceData calldata _priceData,
    bytes calldata _signature,
    address _owner,
    uint256 _reqId
) external override returns (uint256, uint256) {
    _checkLiqudityRouter();
    // verify price data, RUSD asset id is 0
    _checkAndSetPrice(_priceData, _signature);
    Withdraw storage withdraw = allWithdraws[_reqId];
    if (withdraw.lpAmount == 0) revert("StableVault: no withdraw");
    if (withdraw.releasedAt > ChainUtils.getTime()) revert("StableVault:
withdraw not released");
    // check if this reqId's owner is account, and pop it from
userWithdraws
    _popUserWithdraw(_owner, _reqId);
    // burn lp token from sender, burn when claim withdraw
    _burn(_msgSender(), withdraw.lpAmount);
    uint256 _amount = fromShare(withdraw.lpAmount);
    uint256 _fee = _amount * stakeFeeBasisPoints[1] / PRECISION;
    uint256 _output = stableAmountToTokenAmount(withdraw.outputToken,
_amount - _fee);
    if (_output > 0) {
        _withdraw(withdraw.outputToken, _owner, _output);
        withdraw.amount = _output;
        withdraw.priceAtClaim = _priceData.price;
    }
    emit RemoveLiquidity(_owner, _reqId, withdraw.outputToken, _output,
withdraw.lpAmount, totalSupply(), lpPrice);
    return (_output, _fee);
}
```

- **Safety advice**

Since this function is called in the RewardRouter contract, it is impossible to check whether the calling address is the same as the withdrawing address, so it is recommended to adjust the permission of requestWithdraw function and add whitelisting calling restrictions to prevent malicious users from creating malicious LP withdrawing requests. This needs to be adjusted in conjunction with the needs of the project.

- **Repair Status**

ROLLUP.FINANCE has Acknowledged.

### 4.3.7 Variables are updated

- **Risk description**

When there is a contract logic to obtain rewards or transfer funds, the coder mistakenly updates the value of the variable that sends the funds, so that the user can use the value of the variable that is not updated to obtain funds, thus affecting the normal operation of the project.

- **Audit Results : Passed**

### 4.3.8 Floating Point and Numeric Precision

- **Risk Description**

In Solidity, the floating-point type is not supported, and the fixed-length floating-point type is not fully supported. The result of the division operation will be rounded off, and if there is a decimal number, the part after the decimal point will be discarded and only the integer part will be taken, for example, dividing 5 pass 2 directly will result in 2. If the result of the operation is less than 1 in the token operation, for example, 4.9 tokens will be approximately equal to 4, bringing a certain degree of The tokens are not only the tokens of the same size, but also the tokens of the same size. Due to the economic properties of tokens, the loss of precision is equivalent to the loss of assets, so this is a cumulative problem in tokens that are frequently traded.

- **Audit Results : Passed**

### 4.3.9 Default Visibility

- **Risk description**

In Solidity, the visibility of contract functions is public pass default. therefore, functions that do not specify any visibility can be called externally pass the user. This can lead to serious vulnerabilities when developers incorrectly ignore visibility specifiers for functions that should be private, or visibility specifiers that can only be called from within the contract itself. One of the first hacks on Parity's multi-signature wallet was the failure to set the visibility of a function, which defaults to public, leading to the theft of a large amount of money.

- **Audit Results : Passed**

### 4.3.10 tx.origin authentication

- **Risk Description**

tx.origin is a global variable in Solidity that traverses the entire call stack and returns the address of the account that originally sent the call (or transaction). Using this variable for authentication in a smart contract can make the contract vulnerable to phishing-like attacks.

- **Audit Results : Passed**

### 4.3.11 Faulty constructor

- **Risk description**

Prior to version 0.4.22 in solidity smart contracts, all contracts and constructors had the same name. When writing a contract, if the constructor name and the contract name are not the same, the contract will add a default constructor and the constructor you set up will be treated as a normal function, resulting in your original contract settings not being executed as expected, which can lead to terrible consequences, especially if the constructor is performing a privileged operation.

- **Audit Results : Passed**

### 4.3.12 Unverified return value

- **Risk description**

Three methods exist in Solidity for sending tokens to an address: transfer(), send(), call.value(). The difference between them is that the transfer function throws an exception throw when sending fails, rolls back the transaction state, and costs 2300gas; the send function returns false when sending fails and costs 2300gas; the call.value method returns false when sending fails and costs all gas to call, which will lead to the risk of reentrant attacks. If the send or call.value method is used in the contract code to send tokens without checking the return value of the method, if an error occurs, the contract will continue to execute the code later, which will lead to the thought result.

- **Audit Results : Passed**

### 4.3.13 Insecure random numbers

- **Risk Description**

All transactions on the blockchain are deterministic state transition operations with no uncertainty, which ultimately means that there is no source of entropy or randomness within the blockchain ecosystem. Therefore, there is no random number function like rand() in Solidity. Many developers use future block variables such as block hashes, timestamps, block highs and lows or Gas caps to generate random numbers. These quantities are controlled pass the miners who mine them and are therefore not truly random, so using past or present block variables to generate random numbers could lead to a destructive vulnerability.

- **Audit Results : Passed**

### 4.3.14 Timestamp Dependency

- **Risk description**

In blockchains, data block timestamps (block.timestamp) are used in a variety of applications, such as functions for random numbers, locking funds for a period of time, and conditional statements for various time-related state changes. Miners have the ability to adjust the timestamp as needed, for example block.timestamp or the alias now can be manipulated pass the miner. This can lead to serious vulnerabilities if the wrong block timestamp is used in a smart contract. This may not be necessary if the contract is not particularly concerned with miner manipulation of block timestamps, but care should be taken when developing the contract.

- **Audit Results : Passed**

### 4.3.15 Transaction order dependency

- **Risk description**

In a blockchain, the miner chooses which transactions from that pool will be included in the block, which is usually determined pass the gasPrice transaction, and the miner will choose the transaction with the highest transaction fee to pack into the block. Since the information about the transactions in the block is publicly available, an attacker can watch the transaction pool for transactions that may contain problematic solutions, modify or revoke the attacker's privileges or change the state of the contract to the attacker's detriment. The attacker can then take data from this transaction and create a higher-level transaction gasPrice and include its transactions in a block before the original, which will preempt the original transaction solution.

- **Audit Results : Passed**

### 4.3.16 Delegatecall

- **Risk Description**

In Solidity, the delegatecall function is the standard message call method, but the code in the target address runs in the context of the calling contract, i.e., keeping msg.sender and msg.value unchanged. This feature supports implementation libraries, where developers can create reusable code for future contracts. The code in the library itself can be secure and bug-free, but when run in another application's environment, new vulnerabilities may arise, so using the delegatecall function may lead to unexpected code execution.

- **Audit Results : Passed**

### 4.3.17 Denial of Service

- **Risk Description**

Denial of service attacks have a broad category of causes and are designed to keep the user from making the contract work properly for a period of time or permanently in certain situations, including malicious behavior while acting as the recipient of a transaction, artificially increasing the gas required to compute a function causing gas exhaustion (such as controlling the size of variables in a for loop), misuse of access control to access the private component of the contract, in which the Owners with privileges are modified, progress state based on external calls, use of obfuscation and oversight, etc. can lead to denial of service attacks.

- **Audit Results : Passed**

### 4.3.18 Fake recharge vulnerability

- **Risk Description**

The success or failure (true or false) status of a token transaction depends on whether an exception is thrown during the execution of the transaction (e.g., using mechanisms such as require/assert/revert/throw). When a user calls the transfer function of a token contract to transfer funds, if the transfer function runs normally without throwing an exception, the transaction will be successful or not, and the status of the transaction will be true. When balances[msg.sender] < _value goes to the else logic and returns false, no exception is thrown, but the transaction acknowledgement is successful, then we believe that a mild if/else judgment is an undisciplined way of coding in sensitive function scenarios like transfer, which will lead to Fake top-up vulnerability in centralized exchanges, centralized wallets, and token contracts.

- **Audit Results : Passed**

### 4.3.19 Short Address Attack Vulnerability

- **Risk Description**

In Solidity smart contracts, when passing parameters to a smart contract, the parameters are encoded according to the ABI specification. the EVM runs the attacker to send encoded parameters that are shorter than the expected parameter length. For example, when transferring money on an exchange or wallet, you need to send the transfer address address and the transfer amount value. The attacker could send a 19-passte address instead of the standard 20-passte address, in which case the EVM would fill in the 0 at the end of the encoded parameter to make up the expected length, which would result in an overflow of the final transfer amount parameter value, thus changing the original transfer amount.

- **Audit Results : Passed**

### 4.3.20 Uninitialized storage pointer

- **Risk description**

EVM uses both storage and memory to store variables. Local variables within functions are stored in storage or memory pass default, depending on their type. uninitialized local storage variables could point to other unexpected storage variables in the contract, leading to intentional or unintentional vulnerabilities.

- **Audit Results : Passed**

### 4.3.21 Frozen Account bypass

- **Risk Description**

In the transfer operation code in the contract, detect the risk that the logical functionality to check the freeze status of the transfer account exists in the contract code and can be passpassed if the transfer account has been frozen.

- **Audit Results : Passed**

### 4.3.22 Uninitialized

- **Risk description**

The initialize function in the contract can be called pass another attacker before the owner, thus initializing the administrator address.

- **Audit Results : Passed**

### 4.3.23 Reentry Attack

- **Risk Description**

An attacker constructs a contract containing malicious code at an external address in the Fallback function When the contract sends tokens to this address, it will call the malicious code. The call.value() function in Solidity will consume all the gas he receives when it is used to send tokens, so a re-entry attack will occur when the call to the call.value() function to send tokens occurs before the actual reduction of the sender's account balance. The re-entry vulnerability led to the famous The DAO attack.

- **Audit Results : Passed**

### 4.3.24 Integer Overflow

- **Risk Description**

Integer overflows are generally classified as overflows and underflows. The types of integer overflows that occur in smart contracts include three types: multiplicative overflows, additive overflows, and subtractive overflows. In Solidity language, variables support integer types in steps of 8, from uint8 to uint256, and int8 to int256, integers specify fixed size data types and are unsigned, for example, a uint8 type , can only be stored in the range 0 to 2^8-1, that is, [0,255] numbers, a uint256 type can only store numbers in the range 0 to 2^256-1. This means that an integer variable can only have a certain range of numbers represented, and cannot exceed this formulated range. Exceeding the range of values expressed pass the variable type will result in an integer overflow vulnerability.

- **Audit Results : Passed**

## 1. Security Audit Tool

| Tool name | Tool Features |
| --- | --- |
| Oyente | Can be used to detect common bugs in smart contracts |
| securify | Common types of smart contracts that can be verified |
| MAIAN | Multiple smart contract vulnerabilities can be found and classified |
| Lunaray Toolkit | self-developed toolkit |

## Disclaimer：

Lunaray Technology only issues a report and assumes corresponding responsibilities for the facts that occurred or existed before the issuance of this report, Since the facts that occurred after the issuance of the report cannot determine the security status of the smart contract, it is not responsible for this.

Lunaray Technology conducts security audits on the security audit items in the project agreement, and is not responsible for the project background and other circumstances, The subsequent on-chain deployment and operation methods of the project party are beyond the scope of this audit.

This report only conducts a security audit based on the information provided by the information provider to Lunaray at the time the report is issued, If the information of this project is concealed or the situation reflected is inconsistent with the actual situation, Lunaray Technology shall not be liable for any losses and adverse effects caused thereby.

There are risks in the market, and investment needs to be cautious. This report only conducts security audits and results announcements on smart contract codes, and does not make investment recommendations and basis.


https://lunaray.co

https://github.com/lunaraySec

https://twitter.com/lunaray_Sec

http://t.me/lunaraySec