LUNARAY
BLOCKCHAINSECURITY

SMART CONTRACT SECURITY AUDIT REPORT

For Roguex

19 April 2024

lunaray.co

Table of Contents

1. Overview

On April 4, 2024, the security team of Lunaray Technology received the security audit request of the **ROGUEX project**. The team completed the audit of the **ROGUEX smart contract** on April 19, 2024. During the audit process, the security audit experts of Lunaray Technology and the ROGUEX 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 ROGUEX 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 ROGUEX smart contract security audit: **Passed**

Audit Report Hash:
9D7EE7770A76840490FF8E421AC39FF327A3B5849CE038ADA094AEF7DD6C1146

2. Background

2.1 Project Description

Project name	RogueX
Contract type	Spot and perpetual trading
Code language	Solidity
Public chain	Blast
Project website	https://roguex.io
Introduction	RogueX innovates with an AMM that merges perpetual trading into liquidity pools, enhancing asset utilization and supporting leveraged trades for a diverse range of tokens, especially memecoins.
Contract file	BlastBase.sol BlastYield.sol FeeReward.sol HypervisorFactory.sol MasterChefFactory.sol Minter.sol RewardsDistributor.sol ROXToken.sol SwapMinningFactory.sol Voter.sol VotingEscrow.sol VotingRewardFactory.sol DeadLp.sol Lock.sol NFTArt.sol NoDelegateCall.sol NonfungiblePositionManager.sol PerpOrderbook.sol PerpRouter.sol PerpUtils.sol RoguexFactory.sol RoxPerpPool.sol RoxPerpPoolDeployer.sol RoxPosnPool.sol RoxPosnPoolDeployer.sol RoxSpotPool.sol RoxSpotPoolDeployer.sol RoxUtils.sol SwapRouter.sol

2.2 Audit Range

Smart contract file name and corresponding SHA256：

Name	SHA256
BlastBase.sol	29479146F017ED042FB2855B103176C2F9E8B66F57437A49B1DD3920EE3A4F64
BlastYield.sol	D9EBF8C876C40E6493CCC012FE90A290505DA92057572747C59E71B068114492
FeeReward.sol	1F7D24692358460BCA5615840AA90CDB7580AF631865ED9528951B7AC1932DE6
HypervisorFactory.sol	0B62633260552AEDE2946ACFED9F66CBAFB731788EDEA5701D2411ED1DFBB1A1
MasterChefFactory.sol	B1567F2BE340EB40B53D992B9B6A433884E9A963D795454608CCC34FD425B3B3
Minter.sol	A929504C214A0E818C888FC1A666B8048AD63F0F961AF2FA97750F4ECE5BB80D
RewardsDistributor.sol	23335358D976CD6AB4857055666B4DB128AEC5F24A339CF4726706C78C109EF7
ROXToken.sol	B457031F5910AD718D9AA745A07ECF912AAA03260B011EF94FB3E06A24B20366
SwapMinningFactory.sol	7A854A7887AB690A14A830BBE651ECF50F6EA0BE33FFCE7CCED3F226B2DEDE85
Voter.sol	D1B1B07ACE288777C636B27E673A5DD869032536A34147B8B9DC3A07AE385424
VotingEscrow.sol	36AEC476504E5963F95B360614EC213F208B35B772D04D492F4962D0AEF1B5DE
VotingRewardFactory.sol	796D8612D626E161E57356129B5EA5D6FB6193CBA0868EC5A4276AEA44B36B4A
DeadLp.sol	507025E484F5DE64408DDA582A8FAFA2D597F978E8F3025DED3E62565DC418F9
NonfungiblePositionManager.sol	BD2D548D7D9A495ADF7A4487FDCD16F984BA59EB3FC3B40A4B03225D8803B079
PerpOrderbook.sol	55EE3A4894C3B3B8FF45EB5E52375365E0E558E43F23E6DEFA5F55D861C23DFA

PerpRouter.sol	BDD0F447F5AA31BC87111D011210F7B752940FBA21A88DFA2FDCFB3341A2B71B
PerpUtils.sol	5808520C121EE19885CC9ABA253140D5E20B16A12AFA4E1C37A1130F2ADF68E2
RoguexFactory.sol	5C2F69101169916A44EF5CD178C4167D8BEDE95CFA342A719D80B5C8D885B01D
RoxPerpPool.sol	DAC9630AAD91B87BE371C93A17013B6471788DBBCD13AD1F3FE3997A3FAB7BB0
RoxPerpPoolDeployer.sol	448D4066861E8F495F5D377FF327B4200D05A58E3FD0741474BB66F6BFD5BBA4
RoxPosnPool.sol	5A96CD465B2E3F9046B4FFD926AAD9BCDFDFE2349EDDB83E555DF0B8E02354BA
RoxPosnPoolDeployer.sol	550607C304FF133D2B97232E0FF5BEA6B2EC4A9C18157A4A16A2F9184C6CA225
RoxSpotPool.sol	087FB9E235C7055C73A06B1E0B962192BAA10ED73C02AABB90D27E84AF97598A
RoxSpotPoolDeployer.sol	03149DAB257DB411EFA491BD00EE921210CEC43C3A718CC0C06C5E71297BA651
RoxUtils.sol	4E76DCDC764F3F7CC31D0E4CCB178C083282C76AE9F3C70A3DC233F5B0EDE24B
SwapRouter.sol	941045523E4D49E9C801751A43FFA54ED9F79199442B96358826A6E9E6AE6958

3. Project contract details

3.1 Contract Overview

SwapRouter Contract

The contract implements a routing contract for spot exchange that allows to perform precise input or output token exchange operations by invoking a liquidity pool contract, and supports cascading operations across multiple liquidity pools as well as multi-path exchange operations.It also inherits multiple pool contracts to handle transaction fees, multiple invocations, token auto-authorization, etc., and interacts with the spot pool through a callback mechanism.

RoxUtils Contract

The contract implements a series of necessary instrumental functions and parameter settings, such as calculating liquidity, prices, etc.

RoxSpotPoolDeployer Contract

This contract implements a factory contract for spot pools, and its main function is to generate new spot pools based on specified tokens and corresponding parameters.

RoxSpotPool Contract

The contract implements the Spot Trading Pool functionality, which is designed to provide liquidity pooling services for both tokens, allowing users to participate by exchanging, adding or removing liquidity. The contract defines several state variables to track the state of the pool, such as liquidity, price and rate. Through the `Slot0` structure, it stores important information such as the current price of the pool, the current Tick, and the index of the watch array. Various functions are implemented in the contract, including initializing the pool, adding or removing liquidity, performing swap operations, etc. By using various custom libraries such as `Tick`, `TickBitmap`, and `Oracle`, the contract is able to perform complex price calculations, manage Tick state changes, and provide observations of past prices. In addition, it also interacts with external `IRoxPerpPool`, `IRoxPosnPool` and `IRoxUtils` contracts for updating funding rates, managing positions and performing security checks.

RoxPosnPoolDeployer Contract

The contract implements factory contracts for liquidity positions, using custom parameters to create new liquidity pool contracts that are used to process specific token pairs and their trading logic.

RoxPosnPool Contract

This contract essentially implements a contract for liquidity positions, allowing the liquidity provider to increase or decrease liquidity within a specified price range and earn a transaction fee for doing so. Fees earned by the user for providing liquidity can be collected through the `collect` function. The contract is linked to a spot pool (`spotPool`) and a perpetual contract pool (`perpPool`), allowing the liquidity provider to earn fees from both pools.

RoxPerpPoolDeployer Contract

This contract implements a factory contract for perpetual trading pools, which deploys a perpetual pool of corresponding tokens by receiving some parameters.

RoxPerpPool Contract

The contract implements a pool of perpetual contracts, which is used to manage the position operations of the perpetual contracts and to adjust the funding rate and liquidation operations according to the market conditions. The contract also includes operations for increasing and decreasing positions, calculating funding rates, and performing liquidation. Through these functions, users can trade perpetual contracts in the contract and manage their positions and risks according to market conditions.

RoguexFactory Contract

This contract implements the factory contract of the Roguex protocol, which is responsible for deploying Spot pools, Perp pools and Position pools and managing the ownership and control of these pools. The main functions of the contract include creating pools, setting the pool deployer and instrument contract addresses, and setting the Hypervisor factory address. The data structures in the contract include storage charges, pool address mapping relationships, and so on. Through these data structures, the contract can track the status and information of various pools and perform necessary operations.

PerpUtils Contract

This contract is an instrumental contract used to support the perpetual pooling functionality of the Roguex protocol. The contract contains several internal functions and data structures that enable pool parameter setting, trade impact assessment, volume estimation, liquidity calculation and liquidity viewing.

PerpRouter Contract

The contract implements a routing contract to handle perpetual contract-related operations, mainly providing routing functions for perpetual pool transactions in the Roguex protocol. The contract implements pool operations, position management, trade execution and other functions, which support users to open, close, increase or decrease positions in the decentralized perpetual contract market, as well as perform automated stop-loss functions. And it supports the use of chain native tokens or ERC20 tokens for operation.

PerpOrderbook Contract

The contract implements a perpetual order book feature designed to provide the necessary functionality to support the perpetual pool of the Roguex protocol, allowing users to create, manage, and execute orders to increase or decrease perpetual contract positions on the blockchain.

NonfungiblePositionManager Contract

The contract implements an ERC721 compliant NFT Token for LP liquidity provider credentials. It also implements a set of functions for liquidity management, supporting users to add and destroy liquidity, and also allowing liquidity details to be queried.

DeadLp Contract

The contract implements a liquidity management tool that allows users to deposit ETH or ERC20 tokens, choosing the appropriate price range to optimize the liquidity input of their funds. The contract automatically handles token transfers and minting of liquidity tokens, supports dynamic price adjustments, and interacts with the liquidity pool through internal and external interfaces.

VotingRewardFactory Contract

This contract implements a voting reward contract factory and corresponding voting reward contracts. The main functions of the voting reward contract include reward management, reward distribution, reward collection, permission control and periodic reward release.A flexible reward management as well as cyclical reward release mechanism is implemented to incentivize participants to engage in voting and governance activities of the project.

VotingEscrow Contract

The contract implements an NFT Token called veNFT based on ERC721, which in addition to the standard ERC721, also implements additional functions including pledge locking and unpledge for ROX Token tokens, and supports the function of mandatory unpledge, but it is unpledge according to the proportion of the remaining pledge time, and the funds that have not yet arrived at the pledge period will be destroyed. The contract also records the voting points received by the user's pledge.

Voter Contract

The contract implements a voting governance system for liquidity pools, market makers and other protocols in the Roguex ecosystem. The contract allows for the creation of Voting Reward Pools and Swap Mining Reward Pools that are used to reward users for participating in the governance.These pools specify allowable reward tokens upon creation and allow users to deposit tokens for rewards in subsequent operations. The contract also implements a distribution mechanism for governance rewards. Through a distribution function in the contract, rewards can be distributed to users participating in governance according to certain rules to incentivize their participation.

SwapMinningFactory Contract

The contract consists of two main parts, one part implements a factory contract for creating SwapMinning contracts and the other part implements the SwapMinning contract logic. The factory contract is similar to the one in VotingRewardV2Factory.sol. The SwapMinning contract implements a reward distribution system that allows users to receive rewards for swapping operations. It

includes mechanisms to prevent re-entry attacks and a set of functions for managing reward tokens, calculating rewards, depositing and collecting rewards. The contract uses timestamps to identify the different rounds and calculates the reward for each token based on the total supply and reward rate of each round.

ROXToken Contract

The contract implements a standard ERC20 Token, which is a platform coin with the name ROX. The contract has an initialization logic for issuing tokens, but it does not remove the right to mint this Token, and the minter role address can mint any amount.

RewardsDistributor Contract

This contract implements a reward system for managing and distributing time-locked tokens. Allows users to collect rewards periodically based on the time and number of tokens locked in the VotingEscrow contract. The contract determines a user's rewards by tracking the number of tokens due to be distributed each week and basing their rewards on their share of locked tokens at a given point in time. Users can check the number of rewards they have available to them and can actively claim these rewards, which are then deposited into the VotingEscrow contract, increasing the user's locked share. The contract also implements a mechanism for periodically updating the total supply, ensuring that reward allocations are synchronized with the current total number of locked tokens.

Minter Contract

The contract implements the token minting functionality, which is responsible for periodically (weekly) generating tokens and distributing rewards. Embedded in the contract is a firing rate, which decreases over time and is used to calculate the number of tokens that should be minted each week. This minting process takes into account the total supply and circulation of tokens, as well as the number of tokens locked in the system through voting. The contract ensures that new tokens are minted at the beginning of each cycle if the token balance is insufficient to cover the required rewards and growth. A portion of the minted tokens are used for reward distribution and another portion is used to increase the total supply in the vote-locked system, thus keeping the interests of token holders undiluted. The contract ensures that rewards are distributed correctly by interacting with the reward distribution contract and the voting contract, and notifies the outside world via events at each cycle update.

MasterChefFactory Contract

This contract implements the MasterChef factory contract and the corresponding MasterChef contract, which implements a reward distribution system that allows users to make deposits and withdrawals in the contract and distribute reward tokens based on their deposit balances. The contract uses anti-re-entry locks, secure math, and defines events to notify the distribution and collection of rewards. The main logic of the contract is to manage the reward distribution cycle, calculation of rewards, deposits, withdrawals, and reward collection. It also allows the administrator to notify new reward amounts and adjust the reward rate based on the current time and remaining reward cycle.

HypervisorFactory Contract

The contract consists of two main parts, one part implements a factory contract that creates Hypervisor contracts and keeps a list of all Hypervisor addresses and their mapping to Token0 Token1, the other part implements Hypervisor contract logic. The other part implements the Hypervisor contract logic, which implements a liquidity management system that allows users to deposit funds and manage them according to the state of the liquidity pool, and supports rebalancing of liquidity based on price fluctuations.

FeeReward Contract

The contract implements the management and execution of expense allocations, ensuring that rewards are distributed in predetermined proportions by defining multiple allocation parameters and stakeholder shares. Stakeholders include team, treasury, voter, and chef and are managed through an interface, with each category having corresponding share ratios that can be adjusted by the owner of the contract.

3.2 Contract details

RoxPerpPool Contract

Name	Parameter	Attributes
increasePosition	address _account uint256 _sizeDelta bool _long0	external
decreasePosition	bytes32 _key uint256 _collateralDelta uint256 _sizeDelta address _feeRecipient bool _toETH	external
_increaseReserve	uint256 _delta bool _token0	private
_decreaseReserve	uint256 _delta bool _token0	private
_delPosition	bytes32 _key	private
_transferIn	bool _isToken0	private
_transferOut	bool is0 uint256 _amount address _recipient bool _toETH	private
balance	bool is0	private
settle	address _recipient bool _is0 bool _burn uint256 _tokenAmount uint256 _feeAmount	internal
updateFundingRate	none	public
tPid	bool l0	public
getPositionByKey	bytes32 _key	public
rgFeeSlot	none	external
clear	none	external

RoxUtils Contract

Name	Parameter	Attributes
updatePerpUtils	address _pU	onlyOwner
pUtils	none	external
setLiqManager	address _liqManager	onlyOwner
setGlobalSetting	uint8 _maxLeverage uint16 _spotThres uint16 _perpThres uint16 _setlThres uint32 _fdFeePerS uint32 _twapTime uint8 _countMin	onlyOwner
modifyPoolSetting	address _spotPool uint8 _maxLeverage uint16 _spotThres uint16 _perpThres uint16 _setlThres uint32 _fdFeePerS uint32 _twapTime uint8 _countMin bool _del	public
setFactor	uint256 _kMax uint256 _powF uint16 _timeSecDynamic	onlyOwner
spotThres	address _spotPool	public
perpThres	address _spotPool	public
setlThres	address _spotPool	public
fdFeePerS	address _spotPool	public
maxLeverage	address _spotPool	public
getSqrtTwapX96	address spotPool	public
getTwapTickUnsafe	address _spotPool uint32 _sec	public
getSqrtTwapX96Sec	address spotPool uint32 secAgo	public

getCountMin	address _spotPool	public
gOpenPrice	address _perpPool uint256 _sizeDelta bool _long0 bool _isSizeCor	public
getLiqArray	address spotPool bool isToken0 uint256 amount	public
getClosePrice	address _perpPool bool _long0 uint256 _sizeDelta bool _isCor	public
gClosePrice	address _perpPool uint256 _sizeDelta TradeData.TradePosition tP bool _isCor	public
countClose	address _perpPool bool long0 uint32 minC	public
_factor	address _spotPool uint256 _closePrice TradeData.TradePosition tP uint256 _delta	private
getDelta	address _spotPool uint256 _closePriceSqrtX96 TradeData.TradePosition tP	public
validPosition	uint256 collateral uint256 size address spotPool	public
collectPosFee	uint256 size address spotPool	public
gFdPs	address _spotPool address _posnPool uint256 _reserve0 uint256 _reserve1	public
estimate	address spotPool	public

	bool zeroForOne int256 amountSpecified uint24 fee	
nextInitializedTickWithin OneWord	address spotPool int24 tick bool lte int24 tickSpacing	external
availableReserve	address _spotPool bool _l0 bool _l1	public

PerpRouter Contract

Name	Parameter	Attributes
setSwapMining	address addr	public
setUtils	address _roguUtils address _perpOrderbook	external
increasePositionOrder	address _account address _perpPool uint256 _tokenAmount uint256 _sizeDelta bool _long0 uint160 _sqrtPriceX96	external
_increasePosition	address _account address _perpPool uint256 _sizeDelta uint256 _opPrice bool _long0	private
liquidatePosition	address _perpPool bytes32 _key	external
decreasePosition	address _account address _perpPool uint256 _collateralDelta uint256 _sizeDelta bool _long0	external

	bool _toETH uint160 _sqrtPriceX96	
execTakingProfitSet	address _perpPool bytes32 _posKey	external
getPositionKeys	address _account	public
_sender	none	private
unwrapWETH9	uint256 amountMinimum address recipient	public
sweepToken	address token uint256 amountMinimum address recipient	public
refundETH	none	external

PerpOrderbook Contract

Name	Parameter	Attributes
setUtils	address _roguUtils address _tradeRouter	external
getIncreaseOrder	uint256 _id	public
getPendingIncreaseOrdersKeys	address _account	public
getPendingIncreaseOrders	address _account	public
pendingIncreaseOrdersNum	address _account	public
isIncreaseOrderKeyAlive	uint256 _increaseKey	public
getDecreaseOrderByKey	uint256 _decreaseKey	public
getPendingDecreaseOrdersKeys	address _account	public
getPendingDecreaseOrders	address _account	public
pendingDecreaseOrdersNum	address _account	public
isDecreaseOrderKeyAlive	uint256 _decreaseKey	public

getPendingOrders	address _account	public
createIncreaseOrder	address _perpPool uint128 _tokenAmount uint128 _exeFee uint256 _sizeDelta uint160 _triggerPriceSqrtX96 bool _long0 bool _triggerAboveThreshold bool _shouldWarp	external
executeIncreaseOrder	uint256 _key address _feeReceipt	external
cancelIncreaseOrder	uint256 _orderIndex	public
updateIncreaseOrder	uint256 _key int256 _sizeDelta int256 _colDelta int256 _feeDelta uint160 _triggerPrice bool _triggerAboveThreshold	public
createDecreaseOrder	address _perpPool uint128 _exeFee uint128 _colDelta uint256 _sizeDelta uint160 _triggerPriceSqrtX96 bool _long0 bool _triggerAboveThreshold bool _shouldWarp	external
cancelDecreaseOrder	uint256 _orderIndex	public
executeDecreaseOrder	uint256 _key address _feeReceipt	external
updateDecreaseOrder	uint256 _key int256 _sizeDelta int256 _colDelta int256 _feeDelta uint160 _triggerPrice bool _triggerAboveThreshold	public
_transferInETH	none	private

_transferOutETH	uint256 _amountOut addresspayable _receiver	private
validatePositionOrderPrice	address _perpPool bool _triggerAboveThreshold uint256 _triggerPrice	public

RoxPosnPool Contract

Name	Parameter	Attributes
priceSlot	uint psId	public
timeSlot	uint psId	public
prInfo	uint256 timePr	external
writePriceSlot	uint16 _psId uint256 _priceSlot	external
updatePerpFee	uint256 curTime uint16 pr uint256 price uint256 liq uint256 feeDelta bool long0	external
updateSwapFee	int24 tick bool zeroForOne uint256 feeToken uint256 liquidity	external
encodeSlots	uint256 prStart uint256 prEnd bool isPrice	public
positions	bytes32 key	public
positionsSum	bytes32 key	public
burnLp	int24 tickLower int24 tickUpper uint128 liquidityDelta	public
lockLp	int24 tickLower int24 tickUpper	public

	uint256 releaseTime	
increaseLiquidity	address owner int24 tickLower int24 tickUpper uint128 liquidityDelta	onlySpotPool
collect	bytes32 _key uint128 _amount0Requested uint128 _amount1Requested	onlySpotPool
updateFee	bytes32 _key	public
decreaseLiquidity	bytes32 _key uint128 liquidityDelta int24 tick uint160 sqrtPriceX96	onlySpotPool
estimateDecreaseLiquidity	bytes32 _key uint128 liquidityDelta int24 tick uint160 sqrtPriceX96	external
pendingFee	bytes32 _key	external
updatePositionEntryPrice	uint256 entryLiq uint256 entryPrice uint256 newLiq uint256 curPrice	internal
liqTrans	uint128 entryLiq uint256 entryPrice uint256 curPrice	internal
_blockTimestamp	none	internal
initializeObserve	none	onlySpotPool
writeObserve	uint16 startObservationIndex uint32 blockTimestamp int24 tick uint128 liquidity uint16 startObservationCardinality uint16 observationCardinalityNext	onlySpotPool
observeSingle	uint32 time int24 tick uint16 observationIndex	external

	uint128 liquidity uint16 observationCardinality	

RoxSpotPoolDeployer Contract

Name	Parameter	Attributes
deploy	address factory address token0 address token1 uint24 fee address perpPool address posnPool address roxUtils	external

RoxPosnPoolDeployer Contract

Name	Parameter	Attributes
deploy	address factory address token0 address token1 uint24 fee address spotPool address perpPool	external

PerpUtils Contract

Name	Parameter	Attributes
setLiqManager	address _liqManager	onlyOwner
setFactor	uint256 _kMax uint256 _powF uint16 _timeSecDynamic uint16 _timeSecRange	onlyOwner
nextInitializedTickWithin OneWord	address spotPool int24 tick bool lte int24 tickSpacing	public
estimateImpact	address _spotPool uint256 _estiDelta uint256 _revtDelta bool _long0	external
estimate	address spotPool bool zeroForOne int256 amountSpecified uint24 fee	public
calLiqArray0	address spotPool uint256 amountSpecifiedRemaining	external
calLiqArray1	address spotPool uint256 amountSpecifiedRemaining	external
viewLiqArray0	address spotPool uint24 prs	external
viewLiqArray1	address spotPool uint24 prs	external

RoguexFactory Contract

Name	Parameter	Attributes
createHypervisor	address tokenA address tokenB uint24 fee	external
spotOwner	address _spotPool	public
transferOwner	address _pool address _new	external
transferCreator	address _pool address _new	external
setPerpRouter	address _router bool _status	onlyOwner
setNftRouter	address _router bool _status	onlyOwner
setSpotRouter	address _router bool _status	onlyOwner
setPoolDeployer	address _dep address _depTrade address _depPos	onlyOwner
setUtils	address _rUtils address _weth	onlyOwner
setHypervisorFactory	address _hypervisorFactory	onlyOwner
createPool	address tokenA address tokenB uint24 fee address poolOwner	external
transferOwnership	address _owner	external

SwapRouter Contract

Name	Parameter	Attributes
setSwapMining	address addr	onlyOwner
getPool	address tokenA address tokenB uint24 fee	private
swapCallback	int256 amount0Delta int256 amount1Delta bytes _data	external
exactInputInternal	uint256 amountIn address recipient uint160 sqrtPriceLimitX96 SwapCallbackData data	private
exactInputSingle	ExactInputSingleParams params	external
exactInput	ExactInputParams params	external
exactOutputInternal	uint256 amountOut address recipient uint160 sqrtPriceLimitX96 SwapCallbackData data	private
exactOutputSingle	ExactOutputSingleParams params	external
exactOutput	ExactOutputParams params	external

DeadLp Contract

Name	Parameter	Attributes
positionsSum	bytes32 key	external
estimateDecreaseLiquidity	bytes32 key uint128 liquidityDelta int24 tick uint160 sqrtPriceX96	external
mulDiv	uint256 a uint256 b	internal

	uint256 denominator	
mulDivRoundingUp	uint256 a uint256 b uint256 denominator	internal
divRoundingUp	uint256 x uint256 y	internal
getAmount0Delta	uint160 sqrtRatioAX96 uint160 sqrtRatioBX96 uint128 liquidity bool roundUp	internal
getAmount1Delta	uint160 sqrtRatioAX96 uint160 sqrtRatioBX96 uint128 liquidity bool roundUp	internal
totalSupply	none	external
balanceOf	address account	external
transfer	address recipient uint256 amount	external
allowance	address owner address spender	external
approve	address spender uint256 amount	external
transferFrom	address sender address recipient uint256 amount	external
isContract	address account	internal
sendValue	addresspayable recipient uint256 amount	internal
functionCall	address target bytes data string errorMessage	internal
functionCallWithValue	address target bytes data uint256 value string errorMessage	internal
functionStaticCall	address target bytes data	internal

	string errorMessage	
functionDelegateCall	address target bytes data string errorMessage	internal
_verifyCallResult	bool success bytes returndata string errorMessage	private
safeTransfer	IERC20 token address to uint256 value	internal
safeTransferFrom	IERC20 token address from address to uint256 value	internal
safeApprove	IERC20 token address spender uint256 value	internal
safeIncreaseAllowance	IERC20 token address spender uint256 value	internal
safeDecreaseAllowance	IERC20 token address spender uint256 value	internal
_callOptionalReturn	IERC20 token bytes data	private
collect	address recipient int24 tickLower int24 tickUpper uint128 amount0Requested uint128 amount1Requested	external
getTwapTickUnsafe	uint32 _sec	external
burnN	address owner int24 tickLower int24 tickUpper uint128 amount	external
mint	address recipient	external

	int24 tickLower int24 tickUpper uint128 amount bytes data	
token0	none	external
roxPosnPool	none	external
token1	none	external
tickSpacing	none	external
slot0	none	external
positions	bytes32 key	external
liquidity	none	external
liqdCheck	none	external
toUint128	uint256 x	private
getLiquidityForAmount0	uint160 sqrtRatioAX96 uint160 sqrtRatioBX96 uint256 amount0	internal
getLiquidityForAmount1	uint160 sqrtRatioAX96 uint160 sqrtRatioBX96 uint256 amount1	internal
getLiquidityForAmounts	uint160 sqrtRatioX96 uint160 sqrtRatioAX96 uint160 sqrtRatioBX96 uint256 amount0 uint256 amount1	internal
getAmount0ForLiquidity	uint160 sqrtRatioAX96 uint160 sqrtRatioBX96 uint128 liquidity	internal
getAmount1ForLiquidity	uint160 sqrtRatioAX96 uint160 sqrtRatioBX96 uint128 liquidity	internal
getAmountsForLiquidity	uint160 sqrtRatioX96 uint160 sqrtRatioAX96 uint160 sqrtRatioBX96 uint128 liquidity	internal
max	uint256 a uint256 b	internal

min	uint256 a uint256 b	internal
sqrt	uint y	internal
average	uint256 a uint256 b	internal
mintCallback	uint256 amount0Owed uint256 amount1Owed bytes data	external
getSqrtRatioAtTick	int24 tick	internal
getTickAtSqrtRatio	uint160 sqrtPriceX96	internal
getPriceByTick	int24 _tick	internal
leftBoundaryTickWithin	int24 tick	internal
getUpperAndLower	uint256 price int24 tick uint256 priceRange uint256 rangeSion	internal
deposit	none	external
deposit	address spotPool uint256 deposit0 uint256 deposit1 uint256 range	external
_depositDeadLp	address token0 address token1 uint256 priceRange address spotPool	internal
_mintLiquidity	int24 tickLower int24 tickUpper uint128 liquidity address payer address pool	internal
_liquidityForAmounts	int24 tickLower int24 tickUpper uint256 amount0 uint256 amount1 address pool	internal
currentTick	address pool	public
currentPrice	address pool	public

_uint128Safe	uint256 x	internal

NonfungiblePositionManager Contract

Name	Parameter	Attributes
positions	uint256 tokenId	external
cachePoolKey	address pool PoolAddress.PoolKey poolKey	private
mint	MintParams params	external
_mintLiq	MintParams params	private
createAndInitializePoolAndAddLiq	uint160 sqrtPriceX96 MintParams params uint8 _maxLeverage uint16 _spotThres uint16 _perpThres uint16 _setlThres uint32 _fdFeePerS uint32 _twapTime uint8 _countFrame	external
tokenURI	uint256 tokenId	public
increaseLiquidity	IncreaseLiquidityParams params	external
decreaseLiquidity	DecreaseLiquidityParams params	external
_collect	CollectParams params bool toETH	private
collect	CollectParams params	external
collectETH	CollectParams params	external
burn	uint256 tokenId	external
nftcompute	address owner int24 tickLower int24 tickUpper address spotPool	internal
baseURI	none	public

RoxSpotPool Contract

Name	Parameter	Attributes
_blockTimestamp	none	internal
balance0	none	public
balance1	none	public
initialize	uint160 sqrtPriceX96	external
_modifyPosition	PoolData.ModifyPositionParams params	private
_updateLiquidity	int24 tickLower int24 tickUpper int128 liquidityDelta int24 tick uint32 time	private
mint	address recipient int24 tickLower int24 tickUpper uint128 amount bytes data	external
_mint	address recipient int24 tickLower int24 tickUpper uint128 amount bytes data	private
collect	address recipient int24 tickLower int24 tickUpper uint128 amount0Requested uint128 amount1Requested	external
_collect	address owner address recipient int24 tickLower int24 tickUpper uint128 amount0Requested uint128 amount1Requested	private
burnN	address owner	external

	int24 tickLower int24 tickUpper uint128 amount	
_burn	address owner int24 tickLower int24 tickUpper uint128 amount	private
updatePnl	int24 tickLower int24 tickUpper int24 slot0tick int128 liquidityDelta	onlyPerpPool
perpSettle	uint256 amount bool is0 address recipient	onlyPerpPool
swap	address recipient bool zeroForOne int256 amountSpecified uint160 sqrtPriceLimitX96 bytes data	external
liqdCheck	none	public

RoxPerpPoolDeployer Contract

Name	Parameter	Attributes
deploy	address factory address token0 address token1 uint24 fee address spotPool address posnPool	external

BlastBase Contract

Name	Parameter	Attributes
claim	address recipient uint256 amount	external
configureContract	address contractAddress YieldMode _yield GasMode gasMode address governor	external
configure	YieldMode _yield GasMode gasMode address governor	external
configureClaimableYield	none	external
configureClaimableYieldOnBehalf	address contractAddress	external
configureAutomaticYield	none	external
configureAutomaticYieldOnBehalf	address contractAddress	external
configureVoidYield	none	external
configureVoidYieldOnBehalf	address contractAddress	external
configureClaimableGas	none	external
configureClaimableGasOnBehalf	address contractAddress	external
configureVoidGas	none	external
configureVoidGasOnBehalf	address contractAddress	external
configureGovernor	address _governor	external
configureGovernorOnBehalf	address _newGovernor address contractAddress	external
claimYield	address contractAddress address recipientOfYield uint256 amount	external
claimAllYield	address contractAddress address recipientOfYield	external

claimAllGas	address contractAddress address recipientOfGas	external
claimGasAtMinClaimRate	address contractAddress address recipientOfGas uint256 minClaimRateBips	external
claimMaxGas	address contractAddress address recipientOfGas	external
claimGas	address contractAddress address recipientOfGas uint256 gasToClaim uint256 gasSecondsToConsume	external
readClaimableYield	address contractAddress	external
readYieldConfiguration	address contractAddress	external
readGasParams	address contractAddress	external
configurePointsOperator	address operator	external
claimGas	address _recipient	external
claimYieldAll	address _recipient uint256 _amountWETH uint256 _amountUSDB	external

BlastYield Contract

Name	Parameter	Attributes
configureContract	address contractAddress YieldMode _yield GasMode gasMode address governor	external
configure	YieldMode _yield GasMode gasMode address governor	external
configureClaimableYield	none	external
configureClaimableYieldOnBehalf	address contractAddress	external

configureAutomaticYield	none	external
configureAutomaticYieldOnBehalf	address contractAddress	external
configureVoidYield	none	external
configureVoidYieldOnBehalf	address contractAddress	external
configureClaimableGas	none	external
configureClaimableGasOnBehalf	address contractAddress	external
configureVoidGas	none	external
configureVoidGasOnBehalf	address contractAddress	external
configureGovernor	address _governor	external
configureGovernorOnBehalf	address _newGovernor address contractAddress	external
claimYield	address contractAddress address recipientOfYield uint256 amount	external
claimAllYield	address contractAddress address recipientOfYield	external
claimAllGas	address contractAddress address recipientOfGas	external
claimGasAtMinClaimRate	address contractAddress address recipientOfGas uint256 minClaimRateBips	external
claimMaxGas	address contractAddress address recipientOfGas	external
claimGas	address contractAddress address recipientOfGas uint256 gasToClaim uint256 gasSecondsToConsume	external
readClaimableYield	address contractAddress	external
readYieldConfiguration	address contractAddress	external
readGasParams	address contractAddress	external
claimGas	address _recipient	external

claimYieldAll	address _recipient uint256 _amountWETH uint256 _amountUSDB	external
setHandler	address _handler bool _state	onlyOwner
claimAllYield	address contractAddress address recipientOfYield	onlyHandler
claimAllGas	address contractAddress address recipientOfGas	onlyHandler
configureClaimableYieldOnBehalf	address contractAddress	onlyHandler
configureAutomaticYieldOnBehalf	address contractAddress	onlyHandler
configureVoidYieldOnBehalf	address contractAddress	onlyHandler
configureClaimableGasOnBehalf	address contractAddress	onlyHandler
configureVoidGasOnBehalf	address contractAddress	onlyHandler
configureGovernor	address _governor	onlyHandler
configureGovernorOnBehalf	address _newGovernor address contractAddress	onlyHandler
claimGas	address _contract address _recipient	onlyHandler
claimYieldAll	address _contract address _recipient uint256 _amountWETH uint256 _amountUSDB	onlyHandler
executeCall	address target bytes data	onlyOwner

DisFeeReward Contract

Name	Parameter	Attributes
totalSupply	none	external
balanceOf	address account	external
transfer	address recipient uint256 amount	external
allowance	address owner address spender	external
approve	address spender uint256 amount	external
transferFrom	address sender address recipient uint256 amount	external
notifyFeeAmount	address poolAddrss uint amount0 uint amount1	external
createGauge	address _pool address _hypervisor	external
deposit	uint256 _amount address _recipient	external
withdraw	uint256 _amount address _recipient	external
notifyRewardAmount	address token uint256 _amount	external
setParams	address _team address _treasury address _voter	onlyOwner
setShares	uint256 _teamShare uint256 _treasuryShare uint256 _voterShare uint256 _chefShare	onlyOwner
setFlag	bool _f	onlyOwner
withdrawToken	address _token uint _amount	onlyOwner

disFee	address token address pool address chef uint256 amount bool isToken0	external
transferOwnership	address newOwner	onlyOwner

ROXToken Contract

Name	Parameter	Attributes
manager	none	external
renounceManagement	none	external
pushManagement	address newOwner_	external
pullManagement	none	external
manager	none	public
renounceManagement	none	onlyManager
pushManagement	address newOwner_	onlyManager
pullManagement	none	public
setMinter	address _minter bool _status	onlyManager
initialMint	address _recipient	external
approve	address _spender uint _value	external
_mint	address _to uint _amount	internal
_transfer	address _from address _to uint _value	internal
transfer	address _to uint _value	external

transferFrom	address _from address _to uint _value	external
burn	uint256 amount	external
mint	address account uint amount	external

Minter Contract

Name	Parameter	Attributes
max	uint a uint b	internal
min	uint a uint b	internal
sqrt	uint y	internal
cbrt	uint256 n	internal
checkpoint_token	none	external
checkpoint_total_supply	none	external
totalSupply	none	external
approve	address spender uint value	external
_ve	none	external
notifyRewardAmount	uint amount	external
token	none	external
totalSupply	none	external
initialize	none	external
calculate_emission	none	public
weekly_emission	none	public
circulating_emission	none	public
calculate_growth	uint _minted	public
update_period	none	external

RewardsDistributor Contract

Name	Parameter	Attributes
max	uint a uint b	internal
min	uint a uint b	internal
sqrt	uint y	internal
cbrt	uint256 n	internal
totalSupply	none	external
transfer	address recipient uint amount	external
decimals	none	external
symbol	none	external
transferFrom	address sender address recipient uint amount	external
allowance	address owner address spender	external
approve	address spender uint value	external
token	none	external
epoch	none	external
point_history	uint loc	external
user_point_history	uint tokenId uint loc	external
user_point_epoch	uint tokenId	external
checkpoint	none	external
deposit_for	uint tokenId uint value	external
totalSupply	none	external
timestamp	none	external
_checkpoint_token	none	internal
checkpoint_token	none	external

_find_timestamp_epoch	address ve uint _timestamp	internal
_find_timestamp_user_epoch	address ve uint tokenId uint _timestamp uint max_user_epoch	internal
ve_for_at	uint _tokenId uint _timestamp	external
_checkpoint_total_supply	none	internal
checkpoint_total_supply	none	external
_claim	uint _tokenId address ve uint _last_token_time	internal
_claimable	uint _tokenId address ve uint _last_token_time	internal
claimable	uint _tokenId	external
claim	uint _tokenId	external
setDepositor	address _depositor	external

HypervisorFactory Contract

Name	Parameter	Attributes
positionsSum	bytes32 key	external
estimateDecreaseLiquidity	bytes32 key uint128 liquidityDelta int24 tick uint160 sqrtPriceX96	external
mulDiv	uint256 a uint256 b uint256 denominator	internal
mulDivRoundingUp	uint256 a	internal

	uint256 b uint256 denominator	
divRoundingUp	uint256 x uint256 y	internal
getAmount0Delta	uint160 sqrtRatioAX96 uint160 sqrtRatioBX96 uint128 liquidity bool roundUp	internal
getAmount1Delta	uint160 sqrtRatioAX96 uint160 sqrtRatioBX96 uint128 liquidity bool roundUp	internal
_msgSender	none	internal
_msgData	none	internal
recover	bytes32 hash uint8 v bytes32 r bytes32 s	internal
toEthSignedMessageHash	bytes32 hash	internal
totalSupply	none	external
balanceOf	address account	external
transfer	address recipient uint256 amount	external
allowance	address owner address spender	external
approve	address spender uint256 amount	external
transferFrom	address sender address recipient uint256 amount	external
name	none	public
symbol	none	public
decimals	none	public
totalSupply	none	public
balanceOf	address account	public

transfer	address recipient uint256 amount	public
allowance	address owner address spender	public
approve	address spender uint256 amount	public
transferFrom	address sender address recipient uint256 amount	public
increaseAllowance	address spender uint256 addedValue	public
decreaseAllowance	address spender uint256 subtractedValue	public
_transfer	address sender address recipient uint256 amount	internal
_mint	address account uint256 amount	internal
_burn	address account uint256 amount	internal
_approve	address owner address spender uint256 amount	internal
_setupDecimals	uint8 decimals_	internal
_beforeTokenTransfer	address from address to uint256 amount	internal
current	Counter counter	internal
increment	Counter counter	internal
decrement	Counter counter	internal
isContract	address account	internal
sendValue	Address payable recipient uint256 amount	internal
functionCall	address target bytes data	internal

	string errorMessage	
functionCallWithValue	address target bytes data uint256 value string errorMessage	internal
functionStaticCall	address target bytes data string errorMessage	internal
functionDelegateCall	address target bytes data string errorMessage	internal
_verifyCallResult	bool success bytes returndata string errorMessage	private
safeTransfer	IERC20 token address to uint256 value	internal
safeTransferFrom	IERC20 token address from address to uint256 value	internal
safeApprove	IERC20 token address spender uint256 value	internal
safeIncreaseAllowance	IERC20 token address spender uint256 value	internal
safeDecreaseAllowance	IERC20 token address spender uint256 value	internal
_callOptionalReturn	IERC20 token bytes data	private
collect	address recipient int24 tickLower int24 tickUpper uint128 amount0Requested	external

	uint128 amount1Requested	
getTwapTickUnsafe	uint32 _sec	external
burnN	address owner int24 tickLower int24 tickUpper uint128 amount	external
mint	address recipient int24 tickLower int24 tickUpper uint128 amount bytes data	external
token0	none	external
roxPosnPool	none	external
token1	none	external
roxUtils	none	external
tickSpacing	none	external
slot0	none	external
positions	bytes32 key	external
liquidity	none	external
liqdCheck	none	external
toUint128	uint256 x	private
getLiquidityForAmount0	uint160 sqrtRatioAX96 uint160 sqrtRatioBX96 uint256 amount0	internal
getLiquidityForAmount1	uint160 sqrtRatioAX96 uint160 sqrtRatioBX96 uint256 amount1	internal
getLiquidityForAmounts	uint160 sqrtRatioX96 uint160 sqrtRatioAX96 uint160 sqrtRatioBX96 uint256 amount0 uint256 amount1	internal
getAmount0ForLiquidity	uint160 sqrtRatioAX96 uint160 sqrtRatioBX96 uint128 liquidity	internal

getAmount1ForLiquidity	uint160 sqrtRatioAX96 uint160 sqrtRatioBX96 uint128 liquidity	internal
getAmountsForLiquidity	uint160 sqrtRatioX96 uint160 sqrtRatioAX96 uint160 sqrtRatioBX96 uint128 liquidity	internal
max	uint256 a uint256 b	internal
min	uint256 a uint256 b	internal
sqrt	uint y	internal
average	uint256 a uint256 b	internal
mul	int256 a int256 b	internal
div	int256 a int256 b	internal
sub	int256 a int256 b	internal
add	int256 a int256 b	internal
mintCallback	uint256 amount0Owed uint256 amount1Owed bytes data	external
getSqrtRatioAtTick	int24 tick	internal
getTickAtSqrtRatio	uint160 sqrtPriceX96	internal
notifyFeeAmount	address poolAddrss uint amount0 uint amount1	external
createGauge	address _pool address _hypervisor	external
deposit	uint256 _amount address _recipient	external
withdraw	uint256 _amount address _recipient	external

notifyRewardAmount	address token uint256 _amount	external
getPriceByTick	int24 _tick	internal
leftBoundaryTickWithin	int24 tick	internal
getUpperAndLower	uint256 price int24 tick uint256 priceRange uint256 rangeSion	internal
getSpli	uint256 price int24 tick uint256 priceRange uint256 rangeSion uint128 liquidity	internal
disFee	address token address pool address chef uint256 amount bool isToken0	external
deposit	none	external
getTwapTickUnsafe	address _spotPool uint32 _sec	external
deposit	uint256 deposit0 uint256 deposit1 address to	external
withdraw	uint256 shares address to address from	external
_reb	none	internal
_zeroBurn	int24 tickLower int24 tickUpper	internal
zeroBurn	none	internal
rebalanceTime	none	public
_rebalance	none	internal
_compound	none	internal
_beforeTokenTransfer	address from address to	internal

	uint256 amount	
_mintLiquidity	int24 tickLower int24 tickUpper uint128 liquidity address payer	internal
_burnLiquidity	int24 tickLower int24 tickUpper uint128 liquidity address to bool collectAll	internal
_liquidityForShares	int24 tickLower int24 tickUpper uint256 shares	internal
_position	int24 tickLower int24 tickUpper	internal
getTotalAmounts	none	public
getUserAmounts	address account	public
getBasePosition	none	public
getLimitPosition	none	public
_liquidityForAmounts	int24 tickLower int24 tickUpper uint256 amount0 uint256 amount1	internal
currentTick	none	public
getDepositAmountRatio	none	public
currentPrice	none	public
_uint128Safe	uint256 x	internal
feeAmountTickSpacing	uint24 fee	external
getPool	address tokenA address tokenB uint24 fee	external
rangeDelta	uint256 _range uint256 _timeRecords	external
decreaseRange	uint256 _range	external
decP	none	external

timeSlot	none	external
timeSlot	none	public
decreaseRange	uint256 _range	external
rangeDelta	uint256 _range uint256 _timeRecords	external
createHypervisor	address tokenA address tokenB uint24 fee	external
setFeeReward	address _feeReward	onlyOwner
transferOwnership	address newOwner	onlyOwner
setParams	uint256 _MAX uint256 _MIN uint256 _Adjust	onlyOwner
setTimeSlot	uint256 _incP uint256 _decP int24 _tkRange uint32 _twapSec uint32 _reblanceGapSec uint8 _increaseCountThres	onlyOwner

MasterChefFactory Contract

Name	Parameter	Attributes
manager	none	external
renounceManagement	none	external
pushManagement	address newOwner_	external
pullManagement	none	external
manager	none	public
renounceManagement	none	onlyManager
pushManagement	address newOwner_	onlyManager
pullManagement	none	public

max	uint a uint b	internal
min	uint a uint b	internal
sqrt	uint y	internal
cbrt	uint256 n	internal
totalSupply	none	external
transfer	address recipient uint amount	external
decimals	none	external
symbol	none	external
transferFrom	address sender address recipient uint amount	external
allowance	address owner address spender	external
approve	address spender uint value	external
distribute	address _gauge	external
rewardPerToken	address token	public
getRewardRateAtNow	address token	public
lastTimeRewardApplicable	address token	public
getRewardList	none	public
getReward	address _account	external
_getReward	address _account	internal
earned	address token address _account	public
deposit	uint256 _amount address _recipient	external
withdraw	uint256 _amount address _recipient	external
_updateRewards	address _account	internal
notifyRewardAmount	address token uint256 _amount	external

epochNext	uint256 timestamp	internal
setVoter	address _voter	onlyManager

SwapMinningFactory Contract

Name	Parameter	Attributes
manager	none	external
renounceManagement	none	external
pushManagement	address newOwner_	external
pullManagement	none	external
manager	none	public
renounceManagement	none	onlyManager
pushManagement	address newOwner_	onlyManager
pullManagement	none	public
max	uint a uint b	internal
min	uint a uint b	internal
sqrt	uint y	internal
cbrt	uint256 n	internal
totalSupply	none	external
transfer	address recipient uint amount	external
decimals	none	external
symbol	none	external
transferFrom	address sender address recipient uint amount	external
allowance	address owner address spender	external
approve	address spender	external

	uint value	
distribute	address _gauge	external
rewardPerToken	address token uint round	public
lastTimeRewardApplicable	address token uint round	public
getRewardList	none	public
getReward	address _account uint round	external
earned	address token address _account uint round	public
deposit	uint256 _amount address _recipient	external
_deposit	uint256 _amount address _recipient	internal
_updateRewards	address _account uint round	internal
notifyRewardAmount	address token uint256 _amount	external
getCurrRound	none	public
epochNext	uint256 timestamp	internal
setVoter	address _voter	onlyManager

VotingEscrow Contract

Name	Parameter	Attributes
supportsInterface	bytes4 interfaceId	external
balanceOf	address owner	external
ownerOf	uint256 tokenId	external
safeTransferFrom	address from	external

	address to uint256 tokenId	
transferFrom	address from address to uint256 tokenId	external
approve	address to uint256 tokenId	external
setApprovalForAll	address operator bool approved	external
getApproved	uint256 tokenId	external
isApprovedForAll	address owner address operator	external
name	none	external
symbol	none	external
tokenURI	uint256 tokenId	external
onERC721Received	address operator address from uint256 tokenId bytes data	external
totalSupply	none	external
transfer	address recipient uint amount	external
decimals	none	external
symbol	none	external
transferFrom	address sender address recipient uint amount	external
allowance	address owner address spender	external
approve	address spender uint value	external
burn	uint256 amount	external
_tokenURI	uint _tokenId uint _balanceOf uint _locked_end	external

	uint _value	
resetNoVoted	uint _tokenId	external
setTeam	address _team	external
setArtProxy	address _proxy	external
tokenURI	uint _tokenId	external
ownerOf	uint _tokenId	public
_balance	address _owner	internal
balanceOf	address _owner	external
getApproved	uint _tokenId	external
isApprovedForAll	address _owner address _operator	external
approve	address _approved uint _tokenId	public
setApprovalForAll	address _operator bool _approved	external
_clearApproval	address _owner uint _tokenId	internal
_isApprovedOrOwner	address _spender uint _tokenId	internal
isApprovedOrOwner	address _spender uint _tokenId	external
_transferFrom	address _from address _to uint _tokenId address _sender	internal
transferFrom	address _from address _to uint _tokenId	external
safeTransferFrom	address _from address _to uint _tokenId bytes _data	public
_isContract	address account	internal
supportsInterface	bytes4 _interfaceID	external
tokenOfOwnerByIndex	address _owner	external

	uint _tokenIndex	
_addTokenToOwnerList	address _to uint _tokenId	internal
_addTokenTo	address _to uint _tokenId	internal
_mint	address _to uint _tokenId	internal
_removeTokenFromOwnerList	address _from uint _tokenId	internal
_removeTokenFrom	address _from uint _tokenId	internal
_burn	uint _tokenId	internal
get_last_user_slope	uint _tokenId	external
user_point_history__ts	uint _tokenId uint _idx	external
locked__end	uint _tokenId	external
_checkpoint	uint _tokenId LockedBalance old_locked LockedBalance new_locked	internal
_deposit_for	uint _tokenId uint _value uint unlock_time LockedBalance locked_balance DepositType deposit_type	internal
block_number	none	external
checkpoint	none	external
deposit_for	uint _tokenId uint _value	external
_create_lock	uint _value uint _lock_duration address _to	internal
create_lock	uint _value uint _lock_duration	external
create_lock_for	uint _value uint _lock_duration address _to	external

increase_amount	uint _tokenId uint _value	external
increase_unlock_time	uint _tokenId uint _lock_duration	external
withdraw	uint _tokenId	external
withdrawForce	uint _tokenId	external
withdrawForceCalculate	uint _tokenId	public
split	uint256 _from uint256 _amount	external
_createSplitNFT	address _to LockedBalance _newLocked uint originCreateLockTime	private
_find_block_epoch	uint _block uint max_epoch	internal
_balanceOfNFT	uint _tokenId uint _t	internal
balanceOfNFT	uint _tokenId	external
balanceOfNFTAt	uint _tokenId uint _t	external
_balanceOfAtNFT	uint _tokenId uint _block	internal
balanceOfAtNFT	uint _tokenId uint _block	external
totalSupplyAt	uint _block	external
_supply_at	Point point uint t	internal
totalSupply	none	external
totalSupplyAtT	uint t	public
setVoter	address _voter	external
voting	uint _tokenId	external
abstain	uint _tokenId	external
merge	uint _from uint _to	external
encode	bytes data	internal
toString	uint value	internal

_tokenURI	uint _tokenId uint _balanceOf uint _locked_end uint _value	external

VotingRewardFactory Contract

Name	Parameter	Attributes
manager	none	external
renounceManagement	none	external
pushManagement	address newOwner_	external
pullManagement	none	external
manager	none	public
renounceManagement	none	onlyManager
pushManagement	address newOwner_	onlyManager
pullManagement	none	public
max	uint a uint b	internal
min	uint a uint b	internal
sqrt	uint y	internal
cbrt	uint256 n	internal
totalSupply	none	external
transfer	address recipient uint amount	external
decimals	none	external
symbol	none	external
transferFrom	address sender address recipient uint amount	external
allowance	address owner	external

	address spender	
approve	address spender uint value	external
distribute	address _gauge	external
rewardPerToken	address token uint round	public
lastTimeRewardApplicable	address token uint round	public
getRewardList	none	public
getReward	address _account uint round	external
earned	address token address _account uint round	public
deposit	uint256 _amount address _recipient	external
_deposit	uint256 _amount address _recipient	internal
_updateRewards	address _account uint round	internal
notifyRewardAmount	address token uint256 _amount	external
getCurrRound	none	public
epochNext	uint256 timestamp	internal
setVoter	address _voter	onlyManager

Voter Contract

Name	Parameter	Attributes
manager	none	external
renounceManagement	none	external
pushManagement	address newOwner_	external
pullManagement	none	external

manager	none	public
renounceManagement	none	onlyManager
pushManagement	address newOwner_	onlyManager
pullManagement	none	public
max	uint a uint b	internal
min	uint a uint b	internal
sqrt	uint y	internal
cbrt	uint256 n	internal
totalSupply	none	external
transfer	address recipient uint amount	external
decimals	none	external
symbol	none	external
transferFrom	address sender address recipient uint amount	external
allowance	address owner address spender	external
approve	address spender uint value	external
getReward	address _account uint round	external
deposit	uint256 _amount address _recipient	external
notifyRewardAmount	address token uint amount	external
active_period	none	external
update_period	none	external
token	none	external
team	none	external
voting	uint tokenId	external
abstain	uint tokenId	external

totalSupply	none	external
token0	none	external
token1	none	external
notifyRewardAmount	address token uint256 _amount	external
getReward	address _account uint round	external
deposit	uint256 _amount address _recipient	external
notifyRewardAmount	address token uint256 _amount	external
getReward	address _account	external
setHypervisorFactory	address _f	onlyManager
setMinter	address _minter	onlyManager
reset	uint _tokenId	onlyNewEpoch
resetNoVoted	uint _tokenId	external
_reset	uint _tokenId	internal
poke	uint _tokenId	external
notifyFeeAmount	address poolAddrss uint amount0 uint amount1	external
createGauge	address _pool address _hypervisor	external
killGauge	address _gauge	onlyManager
reviveGauge	address _gauge	onlyManager
_epochTimestamp	none	public
weights	address _pool	public
weightsAt	address _pool uint256 _time	public
totalWeight	none	public
totalWeightAt	uint256 _time	public
getPools	none	external
depositSwap	address _pool uint256 _amount	external

	address _recipient	
notifyRewardAmount	uint amount	external
_updateFor	address _gauge	private
_updateForFee	address _gauge	internal
_distribute	address _gauge	internal
_safeTransferFrom	address token address from address to uint256 value	internal

4. Audit details

4.1 Findings Summary

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

4.2 Risk distribution

Name	Risk level	Repair status
Cancel Increase Order Reentry Attack	High	Resolved
Cancel Decrease Order Reentry Attack	High	Resolved
Unrestricted fund withdrawals	Low	Acknowledged
Undiscarded mint rights	Info	Acknowledged
Logic Design Flaw	No	normal
Price Control	No	normal
Floating Point and Numeric Precision	No	normal
Variables are updated	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
Integer Overflow	No	normal

4.3 Risk audit details

4.3.1 Cancel Increase Order 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.

In RoxPerpPool contracts, since the transfer function uses the safeTransfer transfer function, which uses a call function, there may be a call to an external contract address that could result in a contract reentry. However, in the current version of the code audit, the contract called the function before the correct update of the position data, so there is no re-entry point of utilization, just a reminder.

```
function createIncreaseOrder(
    address _perpPool,
    uint128 _tokenAmount,
    uint128 _exeFee,
    uint256 _sizeDelta,
    uint160 _triggerPriceSqrtX96,
    bool _long0,
    bool _triggerAboveThreshold,
    bool _shouldWarp
) external payable {
    require(_tokenAmount > _exeFee, "Fee>Col");
    address _account = msg.sender;
    address token0 = IRoxPerpPool(_perpPool).token0();
    address token1 = IRoxPerpPool(_perpPool).token1();
    address _colToken = _long0 ? token0 : token1;
    if (_shouldWarp) {
        require(_tokenAmount <= msg.value);
```

```
        _transferInETH();
    } else {
        TransferHelper.safeTransferFrom(
            _colToken,
            _account,
            address(this),
            _tokenAmount);
    }
    uint256 _orderIndex = (increaseOrdersIndex += 1);
    // bytes32 _key = getRequestKey(_account, _orderIndex,
"increase");
    increaseOrders[_orderIndex] = OrderData.IncreaseOrder({
        spotpool: IRoxPerpPool(_perpPool).spotPool(),
        account: _account,
        perpPool: _perpPool,
        token0: token0,
        token1: token1,
        key: _orderIndex,
        collateralIn: _tokenAmount - _exeFee,
        sizeDelta: _sizeDelta,
        executionFee: _exeFee,
        long0: _long0,
        triggerAboveThreshold: _triggerAboveThreshold,
        shouldWarp: _shouldWarp,
        triggerPrice: _triggerPriceSqrtX96
    });
    increaseOrderKeysAlive[address(0)].add(_orderIndex);
    increaseOrderKeysAlive[_account].add(_orderIndex);
    emit CreateIncreaseOrder(increaseOrders[_orderIndex]);
}

function cancelIncreaseOrder(uint256 _orderIndex) public {
    OrderData.IncreaseOrder memory order =
increaseOrders[_orderIndex];
    require(isIncreaseOrderKeyAlive(_orderIndex), "no-order");
    require(order.account == msg.sender, "OnlyOwner");
    address _colToken = order.long0
        ? IRoxPerpPool(order.perpPool).token0()
        : IRoxPerpPool(order.perpPool).token1();
    uint256 tokenBack = order.collateralIn + order.executionFee;
```

```
    if (order.shouldWarp) {
        _transferOutETH(tokenBack, payable(msg.sender));
    } else {
        // IERC20(_colToken).safeTransfer(msg.sender, tokenBack);
        TransferHelper.safeTransfer(
            _colToken,
            msg.sender,
            tokenBack);
    }
    emit CancelIncreaseOrder(order);
    increaseOrderKeysAlive[order.account].remove(_orderIndex);
    increaseOrderKeysAlive[address(0)].remove(_orderIndex);
    delete increaseOrders[_orderIndex];
}
```

- **Safety advice**

Three ways to mitigate this problem: 1. Ensure that all transfer operations are executed after a state change so that they do not satisfy the reentry condition; 2. Add reentry locks to external functions in the contract that use both functions; and 3. Limit the relevant contract addresses, prohibit contract calls, and add a whitelist of non-contract addresses.

- **Repair Status**

ROGUEX has Resolved.

4.3.2 Cancel Decrease Order Increase

- **Risk description**

In the PerpOrderbook contract, there is a re-entry for canceling a fill order, and the contract does not add a re-entry restriction to the cancelDecreaseOrder function, which allows the caller to fake the callback function to call the cancelDecreaseOrder function repeatedly, thus maliciously transferring out funds

```
function createDecreaseOrder(
    address _perpPool,
    uint128 _exeFee,
    uint128 _colDelta,
    uint256 _sizeDelta,
    uint160 _triggerPriceSqrtX96,
    bool _long0,
    bool _triggerAboveThreshold,
    bool _shouldWarp
) external payable {
    address _account = msg.sender;
    address token0 = IRoxPerpPool(_perpPool).token0();
    address token1 = IRoxPerpPool(_perpPool).token1();
    address _colToken = _long0 ? token0 : token1;
    if (_shouldWarp) {
        require(_exeFee == msg.value);
        _transferInETH();
    } else {
        TransferHelper.safeTransferFrom(
            _colToken,
            _account,
            address(this),
            _exeFee
        );
    }
    uint256 _orderIndex = (decreaseOrdersIndex++);
    decreaseOrders[_orderIndex] = OrderData.DecreaseOrder({
        spotpool: IRoxPerpPool(_perpPool).spotPool(),
        account: _account,
        perpPool: _perpPool,
        token0: token0,
```

```
        token1: token1,
        key: _orderIndex,
        sizeDelta: _sizeDelta,
        collateralDelta: _colDelta,
        executionFee: _exeFee,
        long0: _long0,
        triggerAboveThreshold: _triggerAboveThreshold,
        shouldWarp: _shouldWarp,
        triggerPrice: _triggerPriceSqrtX96
    });
    decreaseOrderKeysAlive[address(0)].add(_orderIndex);
    decreaseOrderKeysAlive[_account].add(_orderIndex);
    emit CreateDecreaseOrder(decreaseOrders[_orderIndex]);
}

function cancelDecreaseOrder(uint256 _orderIndex) public {
    OrderData.DecreaseOrder memory order =
decreaseOrders[_orderIndex];
    require(isDecreaseOrderKeyAlive(_orderIndex), "no-order");
    require(order.account == msg.sender, "OnlyOwner");
    address _colToken = order.long0
        ? IRoxPerpPool(order.perpPool).token0()
        : IRoxPerpPool(order.perpPool).token1();
    uint256 tokenBack = order.executionFee;
    if (order.shouldWarp) {
        _transferOutETH(tokenBack, payable(msg.sender));
    } else {
        TransferHelper.safeTransfer(_colToken, msg.sender,
tokenBack);
    }
    emit CancelDecreaseOrder(order);
    decreaseOrderKeysAlive[order.account].remove(_orderIndex);
    decreaseOrderKeysAlive[address(0)].remove(_orderIndex);
    delete decreaseOrders[_orderIndex];
}
```

- **Safety advice**

Administrator addresses avoid using a single EOA address for control, and try to use more decentralized permissions management, such as multi-signature wallets or governance voting.

- **Repair Status**

ROGUEX has Resolved.

4.3.3 Unrestricted fund withdrawals

- **Risk description**

Any contract that inherits the abstract contract PeripheryPayments can be withdrawn by any role to fund the native tokens in the current contract.

```
function refundETH() external payable override {
    if (address(this).balance > 0)
TransferHelper.safeTransferETH(msg.sender, address(this).balance);
}
```

- **Safety advice**

Suggested by adding a restriction on the withdrawal of funds to avoid large sums of money being raised by unexpected people all the time

- **Repair Status**

ROGUEX has Acknowledged.

4.3.4 Undiscarded mint rights

- **Risk description**

In the ROXToken contract, the designated administrator role can mint tokens, and the contract does not set the maximum issuance amount and other related parameters, so the Token has not given up the right to mint tokens, and there may be a malicious issuance of tokens due to the leakage of the private key.

```
function mint(address account, uint amount) external returns (bool)
{
    require(isMinter[msg.sender], "not allowed");
    _mint(account, amount);
    return true;
}
```

- **Safety advice**

It is recommended to decentralize the Minter role, such as multi-signing contracts, etc. Try to avoid using the EOA address to hold this privilege to avoid private key leakage or phishing attacks;

- **Repair Status**

ROGUEX has Acknowledged.

4.3.5 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.

- **Audit Results : Passed**

4.3.6 Price Control

- **Risk description**

Price manipulation usually refers to the practice of some large investors or traders of buying or selling large quantities of a particular asset to influence the price of that asset and then capitalizing on the price change to make a profit. This behavior undermines the fairness of the market and creates an uneven playing field for smaller investors.

- **Audit Results : Passed**

4.3.7 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**

4.3.8 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**

4.3.9 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 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_co

http://t.me/lunaraySec