LUNARAY
BLOCKCHAINSECURITY

Smart Contract Security Audit Report

For **Stabilizefi**

22 November 2021

https://lunaray.co

Table of Contents

1. Overview

On Oct 23, 2021, the security team of Lunaray Technology received the security audit request of the **Stabilizefi project**. The team completed the audit of the **Stabilizefi smart contract** on Nov 22, 2021. During the audit process, the security audit experts of Lunaray Technology and the Stabilizefi 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 Stabilizefi 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 Stabilizefi smart contract security audit: **passed**

Audit Report MD5：14A935894A33DA431D01A0080C36DF61

2. Background

2.1 Project Description

Project name	Stabilizefi
Contract type	Token, DeFi
Code language	Solidity
Public chain	Avalanche
Project address	https://stabilize.fi/
Contract file	LiquityBase.sol,ActivePool.sol ,BorrowerOperations.sol, CollSurplusPool.sol,CommunityIssuance.sol,DefaultPool.sol, HintHelpers.sol,MultiTroveGetter.sol,PriceFeed.sol, SortedTroves.sol,StabilityPool.sol,TroveManager.sol, USDToken.sol,StableToken.sol,Token.sol,TokenStaking.sol, TokenStakingV2.sol,VEToken.sol,Airdrop.sol,LPRewards.sol, TeamLock.sol,AddLiquidityWrapper.sol,ProxyHelper.sol, SafeToken.sol,Timelock.sol
Project Description	Stabilize is a stable decentralized borrowing protocol 2.0. Borrow $SUSD against Basic Assets & Interest-Bearing Asset at 0% interest.

2.2 Audit Range

Stabilizefi officially provides the smart contract address on the Avalanche chain：

Name	Address
ActivePool	0xc4392b160E1D57F41F52D1788aA765B6B8771885
BorrowerOperations	0xce0cB4088590021D44930726EA59e20d7a14bDCD
CollSurplusPool	0x9bCD350aa49FBe75Fda5B1F62076fe8797C6E278
DefaultPool	0x694771403876bE9B86C6563B90c12be9b0FBd476
SortedTroves	0x815f7906F39B46E233985fcd5A0e2FDFd7Bc203F
StabilityPool	0x153234eA0Fb1e114d28c1a173E12C67439E42609
TroveManager	0xEA4CC5C00bFb9C8a7011a8ddc693FbE691481c5B
CommunityIssuance	0x0Ae64330cFbf9F1bCE8B3E1B7217E7E76FCB1b38
PriceFeed	0x9167dbBCEB1e7bD3BaDbe08b8D4673918B564DF1
USDToken	0xc53A6EDa2C847ce9f10b5C8D51BC2a9Ed2Fe3d44
HintHelpers	0x6365452659D0F907512AfC18aF89AF764f10ccCE
MultiTroveGetter	0xc0f470539ea5fe2651e6A0B889764EF1a4917C80
TeamLock	0x875EbABA77b618a22200Cc94Be5d08d97EE843c7
TokenStakingV2	0x873b9DE8b936e9453a1A127bB0258f5B5FAb9464
Airdrop	0x6f35C00BcFEa009Fd0a06e7e027B92527e5AE4e2

Timelock	0x1750F78bAA285e88C3A04F3c541705693227 96E8
AddLiquidityWrapper	0x452f131757D3e079B30af35B8CF8308C548cE959
VEToken	0x05D63379D10f4363ADd3c4710b8C871Be3A91d4f
Token	0x37d87e316CB4e35163881fDb6c6Bc0CdBa91dc0A
StableToken	0xAafd2577Fb67366d3C89DB0d627C49D769ee2e5D

2.3 Findings Summary

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

3. Project Contract Details

3.1 Directory Structure

```
└─Stabilizefi
├──Dependencies
│     LiquityBase.sol
├──protocol
│   │   ActivePool.sol
│   │   BorrowerOperations.sol
│   │   CollSurplusPool.sol
│   │   CommunityIssuance.sol
│   │   DefaultPool.sol
│   │   HintHelpers.sol
│   │   MultiTroveGetter.sol
│   │   PriceFeed.sol
│   │   SortedTroves.sol
│   │   StabilityPool.sol
│   │   TroveManager.sol
│   │   USDToken.sol
│   └──global
```

```
│       │   StableToken.sol
│       │   Token.sol
│       │   TokenStaking.sol
│       │   TokenStakingV2.sol
│       │   VEToken.sol
│       └──rewards
│               Airdrop.sol
│               LPRewards.sol
│               TeamLock.sol
└──utils
    AddLiquidityWrapper.sol
    Timelock.sol
```

3.2 Contract Details

LiquityBase

Name	Parameter	Attributes
_getCompositeDebt	uint _debt	internal
_getNetDebt	uint _debt	internal
_getCollGasCompensation	uint _entireColl	internal
getEntireSystemColl	none	public
getEntireSystemDebt	none	public
_getTCR	uint _price	internal
_computeCR	uint _coll uint _debt uint _price	internal
_computeNominalCR	uint _coll uint _debt	internal
getCollDecimals	none	internal
collToDebt	uint _coll uint _price	public
debtToColl	uint _debt uint _price	public
_checkRecoveryMode	uint _price	internal
_requireUserAcceptsFee	uint _fee uint _amount uint _maxFeePercentage	internal

TokenStakingV2

Name	Parameter	Attributes
initializeReentrancyGuard	none	onlyOwner
setVEToken	address _addr	onlyOwner
setBonusMultiplier	uint _bonusMultiplier	onlyOwner
setUnlockPeriod	uint _unlockPeriod	onlyOwner
lock	uint _amount	external
unlock	uint _amount	external
withdraw	none	external
unlockedToLock	uint _amount	external
stakedToLock	uint _amount	external
claimRewards	none	external
_userShare	address _user	internal
setAddresses	address _stakeToken	external
addNewAsset	address _borrowingFeeToken address _redeemingFeeToken address _troveManager address _borrowerOperation address _activePool	onlyOwner
stake	uint _amount	external
unstake	uint _amount	external
_sendRewards	none	internal

Name	Parameter	Attributes
increaseBorrowingFee	uint _fee	external
increaseRedeemingFee	uint _fee	external
increaseTransferFee	uint _fee	external
_increaseFee	uint256 _fee	internal
getPendingGain	address _token address _user	external
_getPendingGain	address _token address _user	internal
_updateUserSnapshots	address _user	internal
_transferOut	address _token address _user uint _amount	internal
_fillingDecimals	address token	internal
_requireCallerIsValidBorrowerOperations	none	internal
_requireCallerIsValidTroveManager	none	internal
_requireCallerIsValidActivePool	none	internal
_requireCallerIsStakeToken	none	internal
_requireNonZeroAmount	uint _amount	internal
_requireUserHasStake	uint currentStake	internal
_requireUserHasLocked	uint currentLocked	internal
_requireUserHasUnlocked	uint currentUnlocked	internal
_requireLockupPeriodHasExpired	none	internal

TokenStaking

Name	Parameter	Attributes
initializeReentrancyGuard	none	onlyOwner
setAddresses	address _stakeToken	external
addNewAsset	address _borrowingFeeToken address _redeemingFeeToken address _troveManager address _borrowerOperation address _activePool	onlyOwner
stake	uint _amount	external
unstake	uint _amount	external
_sendRewards	none	internal
increaseBorrowingFee	uint _fee	external
increaseRedeemingFee	uint _fee	external
increaseTransferFee	uint _fee	external
_increaseFee	uint256 _fee	internal
getPendingGain	address _token address _user	external
_getPendingGain	address _token address _user	internal
_updateUserSnapshots	address _user	internal

Name	Parameter	Attributes
_transferOut	address _token address _user uint _amount	internal
_fillingDecimals	address token	internal
_requireCallerIsValidBorrowerOperations	none	internal
_requireCallerIsValidTroveManager	none	internal
_requireCallerIsValidActivePool	none	internal
_requireCallerIsStakeToken	none	internal
_requireNonZeroAmount	uint _amount	internal
_requireUserHasStake	uint currentStake	internal

VEToken

Name	Parameter	Attributes
name	none	public
symbol	none	public
decimals	none	public
totalSupply	none	public
balanceOf	address account	public
mint	address account uint256 amount	onlyOwner
burn	address account uint256 amount	onlyOwner

TroveManager

Name	Parameter	Attributes
_getOffsetAnd RedistributionVals	uint _debt uint _coll uint _LUSDInStabPool	internal
_getCappedOffsetVals	uint _entireTroveDebt uint _entireTroveColl uint _price	internal
liquidateTroves	uint _n	external
getTotalsFromLiquidate TrovesSequence RecoveryMode	ContractsCache _contractsCache uint _price uint _LUSDInStabPool uint _n	internal
getTotalsFromLiquidate TrovesSequence NormalMode	IActivePool _activePool IDefaultPool _defaultPool uint _price uint _LUSDInStabPool uint _n	internal
_addLiquidationValues ToTotals	LiquidationTotals oldTotals LiquidationValues singleLiquidation	internal
_sendGasCompensation	IActivePool _activePool address _liquidator uint _LUSD uint _ETH	internal
_movePendingTrove RewardsToActivePool	IActivePool _activePool IDefaultPool _defaultPool uint _LUSD uint _ETH	internal
_redeemCollateral	ContractsCache _contractsCache	internal

Name	Parameter	Attributes
FromTrove	address _borrower uint _maxLUSDamount uint _price address _upperPartialRedemption address _lowerPartialRedemption uint _partialRedemptionHintNICR	
_redeemCloseTrove	ContractsCache _contractsCache address _borrower uint _LUSD uint _ETH	internal
_isValidFirst RedemptionHint	ISortedTroves _sortedTroves address _firstRedemptionHint uint _price	internal
redeemCollateral	uint _LUSDamount address _firstRedemptionHint address _upperPartialRedemption address _lowerPartialRedemption uint _partialRedemptionHintNICR uint _maxIterations uint _maxFeePercentage	external
getNominalICR	address _borrower	public
getCurrentICR	address _borrower uint _price	public
_getCurrentTrove Amounts	address _borrower	internal
applyPendingRewards	address _borrower	external

Name	Parameter	Attributes
_applyPendingRewards	IActivePool _activePool IDefaultPool _defaultPool address _borrower	internal
updateTroveReward Snapshots	address _borrower	external
_updateTroveReward Snapshots	address _borrower	internal
getPendingETHReward	address _borrower	public
getPendingLUSD DebtReward	address _borrower	public
hasPendingRewards	address _borrower	public
getEntireDebtAndColl	address _borrower	public
removeStake	address _borrower	external
_removeStake	address _borrower	internal
updateStakeAnd TotalStakes	address _borrower	external
_updateStakeAnd TotalStakes	address _borrower	internal
_computeNewStake	uint _coll	internal
_redistributeDebtAndColl	IActivePool _activePool IDefaultPool _defaultPool uint _debt uint _coll	internal
closeTrove	address _borrower	external

Name	Parameter	Attributes
_closeTrove	address _borrower Status closedStatus	internal
_updateSystemSnapshots _excludeCollRemainder	IActivePool _activePool uint _collRemainder	internal
addTroveOwnerToArray	address _borrower	external
_addTroveOwnerToArray	address _borrower	internal
_removeTroveOwner	address _borrower uint TroveOwnersArrayLength	internal
getTCR	uint _price	external
checkRecoveryMode	uint _price	external
_checkPotentialRecoveryMode	uint _entireSystemColl uint _entireSystemDebt uint _price	internal
_updateBaseRateFrom Redemption	uint _ETHDrawn uint _price uint _totalLUSDSupply	internal
getRedemptionRate	none	public
getRedemptionRateWithDecay	none	public
_calcRedemptionRate	uint _baseRate	internal
_getRedemptionFee	uint _ETHDrawn	internal
getRedemptionFeeWithDecay	uint _ETHDrawn	external
_calcRedemptionFee	uint _redemptionRate uint _ETHDrawn	internal
getBorrowingRate	none	public

Name	Parameter	Attributes
getBorrowingRateWithDecay	none	public
_calcBorrowingRate	uint _baseRate	internal
getBorrowingFee	uint _LUSDDebt	external
getBorrowingFeeWithDecay	uint _LUSDDebt	external
_calcBorrowingFee	uint _borrowingRate uint _LUSDDebt	internal
decayBaseRateFromBorrowing	none	external
_updateLastFeeOpTime	none	internal
_calcDecayedBaseRate	none	internal
_minutesPassedSinceLastFeeOp	none	internal
_requireCallerIsBorrower Operations	none	internal
_requireTroveIsActive	address _borrower	internal
_requireLUSDBalanceCovers Redemption	IUSDToken _lusdToken address _redeemer uint _amount	internal
_requireMoreThanOneTrove InSystem	uint TroveOwnersArrayLength	internal
_requireAmountGreaterThanZero	uint _amount	internal
_requireTCRoverMCR	uint _price	internal
_requireAfterBootstrapPeriod	none	internal
_requireValidMaxFeePercentage	uint _maxFeePercentage	internal
getTroveStatus	address _borrower	external

Name	Parameter	Attributes
getTroveStake	address _borrower	external
getTroveDebt	address _borrower	external
getTroveColl	address _borrower	external
setTroveStatus	address _borrower uint _num	external
increaseTroveColl	address _borrower uint _collIncrease	external
decreaseTroveColl	address _borrower u int _collDecrease	external
increaseTroveDebt	address _borrower uint _debtIncrease	external
decreaseTroveDebt	address _borrower uint _debtDecrease	external

CommunityIssuance

Name	Parameter	Attributes
setAddresses	address _lqtyTokenAddress address _stabilityPoolAddress uint _LQTYSupplyCap	external
issueLQTY	none	external
_getCumulativeIssuanceFraction	none	internal
sendLQTY	address _account uint _LQTYamount	external
_requireCallerIsStabilityPool	none	internal

BorrowerOperations

Name	Parameter	Attributes
setAddresses	address _collTokenAddress	external
	address _troveManagerAddress	
	address _activePoolAddress	
	address _defaultPoolAddress	
	address _stabilityPoolAddress	
	address _gasPoolAddress	
	address _collSurplusPoolAddress	
	address _priceFeedAddress	
	address _sortedTrovesAddress	
	address _lusdTokenAddress	
	address _lqtyStakingAddress	
openTrove	uint _collAmount	external
	uint _maxFeePercentage	
	uint _LUSDAmount	
	address _upperHint	
	address _lowerHint	
addColl	uint _collAmount	external
	address _upperHint	
	address _lowerHint	
moveETHGainToTrove	uint _collAmount	external
	address _borrower	
	address _upperHint	
	address _lowerHint	

Name	Parameter	Attributes
withdrawColl	uint _collWithdrawal address _upperHint address _lowerHint	external
withdrawLUSD	uint _maxFeePercentage uint _LUSDAmount address _upperHint address _lowerHint	external
repayLUSD	uint _LUSDAmount address _upperHint address _lowerHint	external
adjustTrove	uint _collAmount uint _maxFeePercentage uint _collWithdrawal uint _LUSDChange bool _isDebtIncrease address _upperHint address _lowerHint	external
_adjustTrove	uint _collAmount address _borrower uint _collWithdrawal uint _LUSDChange bool _isDebtIncrease address _upperHint address _lowerHint	internal

Name	Parameter	Attributes
	uint _maxFeePercentage	
closeTrove	none	external
claimCollateral	none	external
_triggerBorrowingFee	ITroveManager _troveManager IUSDToken _lusdToken uint _LUSDAmount uint _maxFeePercentage	internal
_getUSDValue	uint _coll uint _price	internal
_getCollChange	uint _collReceived uint _requestedCollWithdrawal	internal
_updateTroveFrom Adjustment	ITroveManager _troveManager address _borrower uint _collChange bool _isCollIncrease uint _debtChange bool _isDebtIncrease	internal
_moveTokensAndETHfrom Adjustment	IActivePool _activePool IUSDToken _lusdToken address _borrower uint _collChange bool _isCollIncrease uint _LUSDChange bool _isDebtIncrease uint _netDebtChange	internal

Name	Parameter	Attributes
_transferTokenToActivePool	IActivePool _activePool uint256 _amount	internal
_activePoolAddColl	IActivePool _activePool uint _amount	internal
_withdrawLUSD	IActivePool _activePool IUSDToken _lusdToken address _account uint _LUSDAmount uint _netDebtIncrease	internal
_repayLUSD	IActivePool _activePool IUSDToken _lusdToken address _account uint _LUSD	internal
_requireSingularCollChange	uint _collAmount uint _collWithdrawal	internal
_requireCallerIsBorrower	address _borrower	internal
_requireNonZeroAdjustment	uint amount uint _collWithdrawal uint _LUSDChange	internal
_requireTroveisActive	ITroveManager _troveManager address _borrower	internal
_requireTroveisNotActive	ITroveManager _troveManager address _borrower	internal
_requireNonZeroDebtChange	uint _LUSDChange	internal
_requireNotInRecoveryMode	uint _price	internal
_requireNoCollWithdrawal	uint _collWithdrawal	internal
_requireValidAdjustment	bool _isRecoveryMode	internal

Name	Parameter	Attributes
InCurrentMode	uint _collWithdrawal bool _isDebtIncrease LocalVariables_adjustTrove _vars	
_requireICRisAboveMCR	uint _newICR	internal
_requireICRisAboveCCR	uint _newICR	internal
_requireNewICRisAboveOldICR	uint _newICR uint _oldICR	internal
_requireNewTCRisAboveCCR	uint _newTCR	internal
_requireAtLeastMinNetDebt	uint _netDebt	internal
_requireValidLUSDRepayment	uint _currentDebt uint _debtRepayment	internal
_requireCallerIsStabilityPool	none	internal
_requireSufficientLUSDBalance	IUSDToken _lusdToken address _borrower uint _debtRepayment	internal
_requireValidMaxFeePercentage	uint _maxFeePercentage bool _isRecoveryMode	internal
_getNewNominalICR FromTroveChange	uint _coll uint _debt uint _collChange bool _isCollIncrease uint _debtChange bool _isDebtIncrease	internal
_getNewICRFromTroveChange	uint _coll uint _debt uint _collChange bool _isCollIncrease uint _debtChange bool _isDebtIncrease uint _price	internal

Name	Parameter	Attributes
_getNewTroveAmounts	uint _coll uint _debt uint _collChange bool _isCollIncrease uint _debtChange bool _isDebtIncrease	internal
_getNewTCRFromTroveChange	uint _collChange bool _isCollIncrease uint _debtChange bool _isDebtIncrease uint _price	internal
getCompositeDebt	uint _debt	external

HintHelpers

Name	Parameter	Attributes
setAddresses	address _sortedTrovesAddress address _troveManagerAddress address _activePoolAddress	external
getRedemptionHints	uint _LUSDamount uint _price uint _maxIterations	external
getApproxHint	uint _CR uint _numTrials uint _inputRandomSeed	external
computeNominalCR	uint _coll uint _debt	external
computeCR	uint _coll uint _debt uint _price	external

CollSurplusPool

Name	Parameter	Attributes
setAddresses	address _collTokenAddress address _borrowerOperationsAddress address _troveManagerAddress address _activePoolAddress	external
getETH	none	external
getCollateral	address _account	external
accountSurplus	address _account uint _amount	external
claimColl	address _account	external
increaseColl	uint256 _amount	external
_requireCallerIsBorrowerOperations	none	internal
_requireCallerIsTroveManager	none	internal
_requireCallerIsActivePool	none	internal

ActivePool

Name	Parameter	Attributes
setAddresses	address _collTokenAddress address _borrowerOperationsAddress address _troveManagerAddress address _stabilityPoolAddress address _defaultPoolAddress address _collSurplusPoolAddress	external
getETH	none	external
getLUSDDebt	none	external
sendETH	address _account uint _amount	external
_increasePoolColl	address _account uint _amount	internal
increaseLUSDDebt	uint _amount	external
decreaseLUSDDebt	uint _amount	external
increaseColl	uint256 _amount	external
_requireCallerIsBorrower OperationsOrDefaultPool	none	internal
_requireCallerIsBOorTroveMorSP	none	internal
_requireCallerIsBOorTroveM	none	internal

DefaultPool

Name	Parameter	Attributes
setAddresses	address _collTokenAddress address _troveManagerAddress address _activePoolAddress	external
getETH	none	external
getLUSDDebt	none	external
sendETHToActivePool	uint _amount	external
_increasePoolColl	address _account uint _amount	internal
increaseLUSDDebt	uint _amount	external
decreaseLUSDDebt	uint _amount	external
increaseColl	uint256 _amount	external
_requireCallerIsActivePool	none	internal
_requireCallerIsTroveManager	none	internal

PriceFeed

Name	Parameter	Attributes
setAddresses	address _priceAggregatorAddress address _bandCallerAddress string _collTokenName	external
addAddresses	address _priceAggregatorAddress address _bandCallerAddress string _collTokenName	onlyOwner
_addAddresses	address _priceAggregatorAddress address _bandCallerAddress string _collTokenName	internal
fetchPrice	none	external
_chainlinkIsBroken	ChainlinkResponse _currentResponse ChainlinkResponse _prevResponse	internal
_badChainlinkResponse	ChainlinkResponse _response	internal
_chainlinkIsFrozen	ChainlinkResponse _response	internal
_chainlinkIsEmpty	none	internal
_chainlinkPriceChange AboveMax	ChainlinkResponse _currentResponse ChainlinkResponse _prevResponse	internal
_bandIsBroken	IStdReference.ReferenceData _response	internal
_bandIsFrozen	IStdReference.ReferenceData _response	internal
_bandIsEmpty	none	internal
_bothOraclesLiveAndUnbroken AndSimilarPrice	ChainlinkResponse _chainlinkResponse ChainlinkResponse _prevChainlinkResponse IStdReference.ReferenceData _bandResponse	internal

Name	Parameter	Attributes
_bandLive	IStdReference.ReferenceData _bandResponse	internal
_chainlinkLive	ChainlinkResponse _chainlinkResponse ChainlinkResponse _prevChainlinkResponse	internal
_bothOraclesSimilarPrice	ChainlinkResponse _chainlinkResponse uint256 _bandPrice	internal
_scaleChainlinkPriceByDigits	uint _price uint _answerDigits	internal
_changeStatus	Status _status	internal
_storePrice	uint _currentPrice	internal
_storeBandPrice	uint256 _bandPrice	internal
_storeChainlinkPrice	ChainlinkResponse _chainlinkResponse	internal
_getCurrentBandResponse	none	internal
_getCurrentChainlinkResponse	none	internal
_getPrevChainlinkResponse	uint80 _currentRoundId uint8 _currentDecimals	internal

MultiTroveGetter

Name	Parameter	Attributes
getMultipleSortedTroves	int _startIdx uint _count	external
_getMultipleSortedTrovesFromHead	uint _startIdx uint _count	internal
_getMultipleSortedTrovesFromTail	uint _startIdx uint _count	internal

StabilityPool

Name	Parameter	Attributes
setAddresses	address _collTokenAddress address _borrowerOperationsAddress address _troveManagerAddress address _activePoolAddress address _lusdTokenAddress address _sortedTrovesAddress address _priceFeedAddress address _communityIssuanceAddress	external
getETH	none	external
getTotalLUSDDeposits	none	external
provideToSP	uint _amount address _frontEndTag	external
withdrawFromSP	uint _amount	external
withdrawETHGainToTrove	address _upperHint address _lowerHint	external
_triggerLQTYIssuance	ICommunityIssuance _communityIssuance	internal
_updateG	uint _LQTYIssuance	internal
_computeLQTYPerUnitStaked	uint _LQTYIssuance uint _totalLUSDDeposits	internal
offset	uint _debtToOffset	external

Name	Parameter	Attributes
	uint _collToAdd	
_computeRewardsPerUnitStaked	uint _collToAdd uint _debtToOffset uint _totalLUSDDeposits	internal
_updateRewardSumAndProduct	uint _ETHGainPerUnitStaked uint _LUSDLossPerUnitStaked	internal
_moveOffsetCollAndDebt	uint _collToAdd uint _debtToOffset	internal
_decreaseLUSD	uint _amount	internal
getDepositorETHGain	address _depositor	public
_getETHGainFromSnapshots	uint initialDeposit Snapshots snapshots	internal
getDepositorLQTYGain	address _depositor	public
getFrontEndLQTYGain	address _frontEnd	public
_getLQTYGainFromSnapshots	uint initialStake Snapshots snapshots	internal
getCompoundedLUSDDeposit	address _depositor	public
getCompoundedFrontEndStake	address _frontEnd	public
_getCompoundedStakeFromSnapshots	uint initialStake Snapshots snapshots	internal
_sendLUSDtoStabilityPool	address _address uint _amount	internal
_sendETHGainToDepositor	uint _amount	internal

Name	Parameter	Attributes
_sendLUSDToDepositor	address _depositor uint LUSDWithdrawal	internal
registerFrontEnd	uint _kickbackRate	external
_setFrontEndTag	address _depositor address _frontEndTag	internal
_updateDepositAndSnapshots	address _depositor uint _newValue	internal
_updateFrontEndStakeAndSnapshots	address _frontEnd uint _newValue	internal
_payOutLQTYGains	ICommunityIssuance _communityIssuance address _depositor address _frontEnd	internal
_requireCallerIsActivePool	none	internal
_requireCallerIsTroveManager	none	internal
_requireNoUnderCollateralizedTroves	none	internal
_requireUserHasDeposit	uint _initialDeposit	internal
_requireUserHasNoDeposit	address _address	internal
_requireNonZeroAmount	uint _amount	internal
_requireUserHasTrove	address _depositor	internal
_requireUserHasETHGain	address _depositor	internal
_requireFrontEndNotRegistered	address _address	internal
_requireFrontEndIsRegisteredOrZero	address _address	internal

Name	Parameter	Attributes
_requireValidKickbackRate	uint _kickbackRate	internal
increaseColl	uint256 _amount	external

SortedTroves

Name	Parameter	Attributes
setParams	uint256 _size address _troveManagerAddress address _borrowerOperationsAddress	external
insert	address _id uint256 _NICR address _prevId address _nextId	external
_insert	ITroveManager _troveManager address _id uint256 _NICR address _prevId address _nextId	internal
remove	address _id	external
_remove	address _id	internal
reInsert	address _id uint256 _newNICR address _prevId address _nextId	external
contains	address _id	public
isFull	none	public
isEmpty	none	public
getSize	none	external
getMaxSize	none	external
getFirst	none	external

Name	Parameter	Attributes
getLast	none	external
getNext	address _id	external
getPrev	address _id	external
validInsertPosition	uint256 _NICR address _prevId address _nextId	external
_validInsertPosition	ITroveManager _troveManager uint256 _NICR address _prevId address _nextId	internal
_descendList	ITroveManager _troveManager uint256 _NICR address _startId	internal
_ascendList	ITroveManager _troveManager uint256 _NICR address _startId	internal
findInsertPosition	uint256 _NICR address _prevId address _nextId	external
_findInsertPosition	ITroveManager _troveManager uint256 _NICR address _prevId address _nextId	internal
_requireCallerIsTroveManager	none	internal
_requireCallerIsBOorTroveM	ITroveManager _troveManager	internal

USDToken

Name	Parameter	Attributes
mint	address _account uint256 _amount	external
burn	address _account uint256 _amount	external
sendToPool	address _sender address _poolAddress uint256 _amount	external
returnFromPool	address _poolAddress address _receiver uint256 _amount	external
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
increaseAllowance	address spender uint256 addedValue	external
decreaseAllowance	address spender uint256 subtractedValue	external
domainSeparator	none	public

Name	Parameter	Attributes
permit	address owner address spender uint amount uint deadline uint8 v bytes32 r bytes32 s	external
nonces	address owner	external
_chainID	none	private
_buildDomainSeparator	bytes32 typeHash bytes32 name bytes32 version	private
_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
_requireValidRecipient	address _recipient	internal
_requireCallerIsBorrowerOperations	none	internal
_requireCallerIsBOorTroveMorSP	none	internal
_requireCallerIsStabilityPool	none	internal
_requireCallerIsTroveMorSP	none	internal
name	none	external
symbol	none	external
decimals	none	external
version	none	external
permitTypeHash	none	external

TokenStaking

Name	Parameter	Attributes
setAddresses	address _stakeToken	external
addNewAsset	address _borrowingFeeToken address _redeemingFeeToken address _troveManager address _borrowerOperation address _activePool	onlyOwner
stake	uint _amount	external
unstake	uint _amount	external
_sendRewards	none	internal
increaseBorrowingFee	uint _fee	external
increaseRedeemingFee	uint _fee	external
increaseTransferFee	uint _fee	external
_increaseFee	uint256 _fee	internal
getPendingGain	address _token address _user	external
_getPendingGain	address _token address _user	internal
_updateUserSnapshots	address _user	internal
_transferOut	address _token address _user uint _amount	internal
_fillingDecimals	address token	internal
_requireCallerIsValid BorrowerOperations	none	internal
_requireCallerIsValidTroveManager	none	internal

Name	Parameter	Attributes
_requireCallerIsValidActivePool	none	internal
_requireCallerIsStakeToken	none	internal
_requireNonZeroAmount	uint _amount	internal
_requireUserHasStake	uint currentStake	internal

StableToken

Name	Parameter	Attributes
decimals	none	external
addUnderlyingToken	address token	onlyOwner
setPermissionToWrap	address token bool enable	onlyOwner
setPermissionToUnwrap	address token bool enable	onlyOwner
wrap	address tokenAddress uint wrapAmount address receiver	external
unwrap	address tokenAddress uint unwrapAmount address receiver	external

Token

Name	Parameter	Attributes
setExcludedFromFeeForSending	address _addr bool _enable	onlyOwner
setExcludedFromFeeForReceiving	address _addr bool _enable	onlyOwner
setTransferFeeRatio	uint256 _transferFeeRatio	onlyOwner
totalSupply	none	external
balanceOf	address account	external
getDeploymentStartTime	none	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
increaseAllowance	address spender uint256 addedValue	external
decreaseAllowance	address spender uint256 subtractedValue	external
sendToTokenStaking	address _sender uint256 _amount	external
domainSeparator	none	public
permit	address owner address spender uint amount uint deadline uint8 v bytes32 r bytes32 s	external
nonces	address owner	external
_chainID	none	private

Name	Parameter	Attributes
_buildDomainSeparator	bytes32 typeHash bytes32 name bytes32 version	private
isTxExcludedFromFee	address sender address recipient	internal
_transfer	address sender address recipient uint256 amount	internal
_transferWithFee	address sender address recipient uint256 amount	internal
_transferStandard	address sender address recipient uint256 amount	internal
_transferFeeToTokenStaking	address sender uint256 amount	internal
_mint	address account uint256 amount	internal
_burn	address account uint256 amount	internal
_approve	address owner address spender uint256 amount	internal
_requireCallerIsTokenStaking	none	internal
_requireValidTransferFeeRatio	uint256 _ratio	internal
name	none	external
symbol	none	external
decimals	none	external
version	none	external
permitTypeHash	none	external

4. Audit Details

4.1 Risk Distribution

Name	Risk level	Status
Re-entry Attack	Medium	Resolved
Contract Initialization Risks	Info	Acknowledged
Administrator Permissions	Info	Resolved
Unused Local Variables	No	Passed
Gas Consumption Due to for Loops	No	Passed
Self-transfer Issues	No	Passed
Floating Point and Numeric Precision	No	Passed
Default Visibility	No	Passed
tx.origin Authentication	No	Passed
Wrong constructor	No	Passed
Unverified Return Value	No	Passed
Insecure Random Numbers	No	Passed
Timestamp Dependent	No	Passed
Transaction order Dependence	No	Passed
Delegatecall	No	Passed
Call	No	Passed
Denial of Service	No	Passed
Logical Design Flaw	No	Passed

Fake Recharge Vulnerability	No	Passed
Short Address Attack	No	Passed
Uninitialized Storage Pointer	No	Passed
Frozen Account Bypass	No	Passed
Uninitialized	No	Passed
Integer Overflow	No	Passed

4.2 Risk Audit Details

4.2.1 Re-entry Attack

- **Risk description**

Tokenstaking.sol contract, the _transferOut method has the following code:

```
function _transferOut(address _token, address _user, uint _amount) inte
rnal {
        if (_amount > 0) {
            if (_token == GAS_TOKEN_ADDR) {
                SafeToken.safeTransferETH(_user, _amount);
            } else {
                SafeToken.safeTransfer(_token, _user, _amount);
            }
        }
    }
```

Trace down to the safeTransferETH function:

```
  function safeTransferETH(address to, uint256 value) internal {
    // solhint-disable-next-line no-call-value
    (bool success, ) = to.call{value: value}(new bytes(0));
    require(success, "!safeTransferETH");
  }
```

There is a re-entry vulnerability here, and to determine if the vulnerability can be exploited, trace the function call upwards:

```
    function _sendRewards() internal {
        uint feeTokenCounts = feeTokens.length;
        for (uint i = 0 ; i < feeTokenCounts; i++) {
            address feeToken = feeTokens[i];
            uint tokenGain = _getPendingGain(feeToken, msg.sender);
            _transferOut(feeToken, msg.sender, tokenGain);
            emit StakingGainsWithdrawn(msg.sender, feeToken, tokenGai
n);
        }
    }
```

Further upward tracking:

```
function stake(uint _amount) external override {
        _requireNonZeroAmount(_amount);

        uint currentStake = stakes[msg.sender];
```

```
        if (currentStake != 0) {
            _sendRewards();
        }

        _updateUserSnapshots(msg.sender);

        uint newStake = currentStake.add(_amount);
        stakes[msg.sender] = newStake;
        totalTokenStaked = totalTokenStaked.add(_amount);
        emit totalTokenStakedUpdated(totalTokenStaked);

        IToken(stakeToken).sendToTokenStaking(msg.sender, _amount);
        emit StakeChanged(msg.sender, newStake);
    }
```

It can be seen that the reentry attack is achieved through malicious constructs.

Also the unstake function on line 107 has a similar re-entry problem.

- **Safety advice**

1. add nonReentrant modifiers to the take/unstake functions.

```
function stake(uint _amount) external override nonReentrant {}
function unstake(uint _amount) external override nonReentrant {}
```

2. update state variables before transferring them
3. can add the modifier lock to restrict some functions from calling each other.

- **Repair Status**

By communicating with Stabilizefi officials, the nonReentrant modifier has been added to prevent the risk of re-entry attack.

4.2.2 Contract Initialization Risks

- **Risk description**

In several contracts such as StabilityPool, BorrowerOperations, CommunityIssuance, TroveManager, etc. the initialization methods are all modified by initializer, and the method attribute is external external call, if the contract initialization method is not called in time after successful deployment on the contract chain If the contract initialization method is not called in time after successful deployment on the contract chain, unexpected risks or loss of funds may occur, as shown in the following code:

```
    function setAddresses(
        address _collTokenAddress,
        address _borrowerOperationsAddress,
        address _troveManagerAddress,
        address _activePoolAddress,
        address _lusdTokenAddress,
        address _sortedTrovesAddress,
        address _priceFeedAddress,
        address _communityIssuanceAddress
    )
        external
        override
        initializer /// noneage
    {

        P = DECIMAL_PRECISION;

        collTokenAddress = _collTokenAddress;
        borrowerOperations = IBorrowerOperations(_borrowerOperationsAdd
ress);
        troveManager = ITroveManager(_troveManagerAddress);
        activePool = IActivePool(_activePoolAddress);
        lusdToken = IUSDToken(_lusdTokenAddress);
        sortedTroves = ISortedTroves(_sortedTrovesAddress);
        priceFeed = IPriceFeed(_priceFeedAddress);
        communityIssuance = ICommunityIssuance(_communityIssuanceAddres
s);

        emit CollTokenAddressChanged(_collTokenAddress);
        emit BorrowerOperationsAddressChanged(_borrowerOperationsAddres
s);
        emit TroveManagerAddressChanged(_troveManagerAddress);
        emit ActivePoolAddressChanged(_activePoolAddress);
        emit LUSDTokenAddressChanged(_lusdTokenAddress);
        emit SortedTrovesAddressChanged(_sortedTrovesAddress);
```

```
        emit PriceFeedAddressChanged(_priceFeedAddress);
        emit CommunityIssuanceAddressChanged(_communityIssuanceAddres
s);

        SafeToken.safeApprove(_collTokenAddress, _borrowerOperationsAdd
ress, uint(-1));
    }
```

- **Safety advice**

It is recommended that the initialization method be called promptly after the contract is deployed.

- **Repair Status**

The risk has been officially confirmed through communication with Stabilizefi officials.

4.2.3 Administrator Permissions

- **Risk description**

Tokenstaking contracts, LPRewards and other contracts, the administrator can set address permissions and other sensitive operations, if the administrator's private key is controlled by a malicious person, it may lead to the loss of abnormal funds and shake the stability of the market, as shown in the following code:

```
    function setVEToken(address _addr) external onlyOwner {
        VEToken = IVEToken(_addr);
        emit VETokenChanged(_addr);
    }

    function setBonusMultiplier(uint _bonusMultiplier) external onlyOwn
er {
        bonusMultiplier = _bonusMultiplier;
        emit BonusMultiplierChanged(_bonusMultiplier);
    }

    function setUnlockPeriod(uint _unlockPeriod) external onlyOwner {
        unlockPeriod = _unlockPeriod;
        emit UnlockPeriodChanged(_unlockPeriod);
    }
    function addNewAsset(
        address _borrowingFeeToken,
        address _redeemingFeeToken,
        address _troveManager,
        address _borrowerOperation,
        address _activePool
    )
        external
        onlyOwner
        override
    {
        require(!isFeeToken[_borrowingFeeToken], "fee token has been ad
ded!");
        require(!isFeeToken[_redeemingFeeToken], "fee token has been ad
ded!");

        isTM[_troveManager] = true;
        isBO[_borrowerOperation] = true;

        if (_redeemingFeeToken == GAS_TOKEN_ADDR) {
            isAP[_activePool] = true;
        }
```

```
        feeTokenMap[_troveManager] = _redeemingFeeToken;
        feeTokens.push(_redeemingFeeToken);
        isFeeToken[_redeemingFeeToken] = true;

        feeTokenMap[_borrowerOperation] = _borrowingFeeToken;
        feeTokens.push(_borrowingFeeToken);
        isFeeToken[_borrowingFeeToken] = true;

        emit NewAssetTokenAddress(
            _troveManager, _borrowerOperation, _activePool, _redeemingF
eeToken, _borrowingFeeToken
        );
    }
```

- **Safety advice**

It is recommended to set TimeLock time lock to time bound the administrator operation; it is recommended to store this administrator key securely.

- **Repair Status**

After verification, Stabilizefi has officially transferred the administrator authority to the TimeLock contract.

4.2.4. Unused local variables

- **Risk Description**

In Solidity contracts, there may be operations that assign values to partial variables in multiple methods due to coders' negligence, but there is no operation on the variable in the logic afterwards. If this issue occurs, it may increase the gas consumption for contract deployment and operation.

- **Audit Results : Passed**

4.2.5. Gas Consumption Due to for Loops

- **Risk Description**

In Solidity contract, due to some logic functions need, may use for loop for processing, if there is no strict judgment on for loop, may make the loop keep running, consume gas or affect the contract logic, if this problem occurs, may increase the contract running gas consumption, serious can cause the contract can not enter the normal logic.

- **Audit Results : Passed**

4.2.6 Self-transfer issues

- **Risk Description**

In Solidity contracts, there will be transfer function, most contracts use ERC20 transfer method to handle the transfer, but some of the transfer may require developers to write the transfer logic due to the need to fit the project logic function, if there are rewards or other means to obtain funds, self-transfer can be made to obtain a large number of fees normal funds acquisition.

- **Audit Results : Passed**

4.2.7 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 Findings : Passed**

4.2.8 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**

4.2.9 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**

4.2.10 Wrong 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**

4.2.11 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**

4.2.12 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**

4.2.13 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**

4.2.14 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**

4.2.15 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**

4.2.16 Call

- **Risk Description**

The call function is similar to the delegatecall function in that it is an underlying function provided pass Solidity, a smart contract writing language, to interact with external contracts or libraries, but when the call function method is used to handle an external Standard Message Call to a contract, the code runs in the environment of the external contract/function The call function is used to interact with an external contract or library. The use of such functions requires a determination of the security of the call parameters, and caution is recommended. An attacker could easily borrow the identity of the current contract to perform other malicious operations, leading to serious vulnerabilities.

- **Audit results : Passed**

4.2.17 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**

4.2.18 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**

4.2.19 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**

4.2.20 Short Address Attack

- **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**

4.2.21 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 Findings : Passed**

4.2.22 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**

4.2.23 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**

4.2.24 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**

5. 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