SLOWMIST

Smart Contract Security Audit Report

Contents

1. Executive Summary

On Feb. 22, 2021, the SlowMist security team received the InsurAce team's security audit application for InsurAce, developed the audit plan according to the agreement of both parties and the characteristics of the project, and finally issued the security audit report.

The SlowMist security team adopts the strategy of "white box lead, black, grey box assists" to conduct a complete security test on the project in the way closest to the real attack.

SlowMist Smart Contract DeFi project test method:

Black box testing	Conduct security tests from an attacker's perspective externally.
Grey box testing	Conduct security testing on code module through the scripting tool, observing the internal running status, mining weaknesses.
White box testing	Based on the open source code, non-open source code, to detect whether there are vulnerabilities in programs such as nodes, SDK, etc.

SlowMist Smart Contract DeFi project risk level:

Critical vulnerabilities	Critical vulnerabilities will have a significant impact on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High-risk vulnerabilities	High-risk vulnerabilities will affect the normal operation of DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium-risk vulnerabilities	Medium vulnerability will affect the operation of DeFi project. It is recommended to fix medium-risk vulnerabilities.

Low-risk vulnerabilities	Low-risk vulnerabilities may affect the operation of DeFi project in certain scenarios. It is suggested that the project party should evaluate and consider whether these vulnerabilities need to be fixed.
Weaknesses	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.
Enhancement Suggestions	There are better practices for coding or architecture.

2. Audit Methodology

Our security audit process for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using public and in-house automated analysis tools.
- Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that was considered during the audit of the smart contract:

- Reentrancy attack and other Race Conditions
- Replay attack
- Reordering attack
- Short address attack
- Denial of service attack
- Transaction Ordering Dependence attack
- Conditional Completion attack
- Authority Control attack
- Integer Overflow and Underflow attack

- TimeStamp Dependence attack
- Gas Usage, Gas Limit and Loops
- Redundant fallback function
- Unsafe type Inference
- Explicit visibility of functions state variables
- Logic Flaws
- Uninitialized Storage Pointers
- Floating Points and Numerical Precision
- tx.origin Authentication
- "False top-up" Vulnerability
- Scoping and Declarations

3. Project Background

3.1 Project Introduction

InsurAce is a decentralized insurance protocol, aiming to provide reliable, robust, and carefree DeFi insurance services to DeFi users, with very low premiums and sustainable investment returns. We respect all the DeFi insurance pioneers and do not consider ourselves as a competitor to the existing players, but a necessary complementary role to the immense and expansive DeFi world.

Project website:

https://www.insurace.io

Audit version code:

smart-contracts-slowmist-review.zip(SHA256):

afb1bd268b8c2accbd5d82a6dedb4070d5bb26822f11f26c195cbf695e0cbe5f

Fixed version code:

smart-contracts-slowmist-review.zip(SHA256):

dd1cad7b409a6826b4ddb1d980cce1d915ff725b32651806c3123d7357d8f57f

4. Code Overview

4.1 Contracts Description

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

INSURToken			
Function Name	Visibility	Mutability	Modifiers
initializeINSUR	Public	Can Modify State	initializer
addSender	External	Can Modify State	onlyAdmin
getSenders	External	-	onlyAdmin
removeSender	External	Can Modify State	onlyAdmin
_beforeTokenTransfer	Internal	Can Modify State	-
_validSender	Private	-	-
delegate	External	Can Modify State	-
_delegate	Private	Can Modify State	-
_moveDelegates	Private	Can Modify State	-
_writeCheckpoint	Private	Can Modify State	-
getPriorVotes	Public	-	-

SecurityMatrix			
Function Name	Visibility	Mutability	Modifiers
initializeSecurityMatrix	Public	Can Modify State	initializer
pauseAll	External	Can Modify State	onlyOwner whenNotPaused
unPauseAll	External	Can Modify State	onlyOwner whenPaused

addAllowdCallersPerCallee	External	Can Modify State	onlyOwner
setAllowdCallersPerCallee	External	Can Modify State	onlyOwner
isAllowdCaller	External	-	whenNotPaused
getAllowedCallees	External	-	-
getAllowedCallersPerCallee	External	-	-

FixedVesting			
Function Name	Visibility	Mutability	Modifiers
initializeFixedVesting	public	Can Modify State	-
pauseAll	external	Can Modify State	onlyOwner whenNotPaused
unPauseAll	external	Can Modify State	onlyOwner whenPaused
startVesting	external	Can Modify State	onlyOwner
setInsurTokenAddress	external	Can Modify State	onlyOwner
setupVestors	external	Can Modify State	OnlyVestor
viewWithdrawableRewardPV	Public	-	-
withdrawRewardPV	external	Can Modify State	OnlyVestor

StakeOps			
Function Name	Visibility	Mutability	Modifiers
initializeStakeOps	Public	Can Modify State	initializer
setup	External	Can Modify State	onlyOwner
pauseAll	External	Can Modify State	onlyOwner whenNotPaused
unPauseAll	External	Can Modify State	onlyOwner whenPaused
_reCalcPerStaker	Private	Can Modify State	-
getTVL	External	-	-
getStakeSettings	External	-	-
stakeTokens	External	-	whenNotPaused nonReentrant

StakersData			
Function Name	Visibility	Mutability	Modifiers
setup	External	Can Modify State	onlyOwner
setAccuWdableAmt	External	Can Modify State	allowedCaller
setUnstakeLockTotalAmt	External	Can Modify State	allowedCaller
setAccuWdableAmtPS	External	Can Modify State	allowedCaller
setUnstakeLkTtAmtPS	External	Can Modify State	allowedCaller
pushUnstkLkArrAmtPS	External	Can Modify State	allowedCaller
purgeUnstkLkAmtBlkPS	External	Can Modify State	allowedCaller
getUnstkLkArrAmtPS	External	-	-
getTVL	External	-	-
getUnstkLkArrBlkPS	External	-	-
pushUnstkLkArrBlkPS	External	Can Modify State	allowedCaller
setAccuRwHvAmt	External	Can Modify State	allowedCaller
setAccuRwHvAmtPS	External	Can Modify State	allowedCaller
setAccuRwAmt	External	Can Modify State	allowedCaller
setAccuRwAmtPS	External	Can Modify State	allowedCaller
setStakedAmtAccumulated	External	Can Modify State	allowedCaller
setStkAmtPS	External	Can Modify State	allowedCaller
getRewardToken	External	-	-
getStakedToken	External	-	-
getStakersArray	External	-	-
pushStakersArray	External	Can Modify State	allowedCaller
setLastCalcBlockPS	External	Can Modify State	allowedCaller

RewardOps			
Function Name	Visibility	Mutability	Modifiers
initializeRewardOps	Public	Can Modify State	initializer
setup	External	Can Modify State	onlyOwner
pauseAll	External	Can Modify State	onlyOwner whenNotPaused

unPauseAll	External	Can Modify State	onlyOwner whenPaused
_reCalcPerStaker	Private	Can Modify State	-
getRewardAmount	External	-	-
getWdAmtAftFee	External	-	-
harvestRewardToken	External	Can Modify State	nonReentrant onlyStaker whenNotPaused

ScheduledMiningProgram			
Function Name	Visibility	Mutability	Modifiers
initializeScheduledMiningProgram	Public	Can Modify State	initializer
setup	External	Can Modify State	onlyOwner
getMiningProgramSettings	External	-	allowedCaller
setMiningProgramSettings	External	Can Modify State	onlyOwner
setGRewardAmtPerBlock	External	Can Modify State	allowedCaller
canWithdrawTokens	External	-	allowedCaller
canStake	External	-	allowedCaller
canProposeUnstake	External	-	allowedCaller
showRewardTokenRatePerStakedTokenByBlock	External	-	onlyOwner
showRewardTokenRatePerStakedToken	External	-	onlyOwner
wdAmtAfterFee	External	-	allowedCaller
reCalcAPY	External	Can Modify State	allowedCaller
_getDeltaAccumulativeRewardsWithFixRatePerStaker	Private	-	-
_getDelWdableAmtPS	Private	-	-
getDelWdableAmtPS	External	-	allowedCaller
_getDelAccuRwAmtPS	Private	-	-
getDelAccuRwAmtPS	External	-	allowedCaller

Schedules			
Function Name	Visibility	Mutability	Modifiers
initSchedules	Internal	Can Modify State	initializer
pushMiningSchedule	External	Can Modify State	onlyOwner nonReentrant

getCurrentMiningScheduleCounter	Public	-	-
popMiningSchedule	External	Can Modify State	onlyOwner nonReentrant
showMiningScheduleByCounter	External	-	-

StakersAdminOps			
Function Name	Visibility	Mutability	Modifiers
initializeStakersAdminOps	Public	Can Modify State	initializer
setup	External	Can Modify State	onlyOwner
pauseAll	External	Can Modify State	onlyOwner whenNotPaused
unPauseAll	External	Can Modify State	onlyOwner whenPaused
getStakers	External	-	onlyOwner
getUnstakeLockArrPS	External	-	onlyOwner
setPerBlkReward	External	Can Modify State	onlyOwner
clearStakersDelta	External	Can Modify State	onlyOwner
reCalcPerStaker	Public	Can Modify State	onlyOwner

UnstakeOps			
Function Name	Visibility	Mutability	Modifiers
initializeUnstakeOps	Public	Can Modify State	initializer
setup	External	Can Modify State	onlyOwner
pauseAll	External	Can Modify State	onlyOwner whenNotPaused
unPauseAll	External	Can Modify State	onlyOwner whenPaused
_reCalcPerStaker	Private	Can Modify State	-
getStakedAmount	External	-	-
proposeUnstake	External	Can Modify State	nonReentrant onlyStaker whenNotPaused
getWithdrawableAmount	External	-	-
getUnstakeLockArrPS	External	-	-
withdrawTokens	External	Can Modify State	nonReentrant onlyStaker whenNotPaused

CapitalPool			
Function Name	Visibility	Mutability	Modifiers
initializeCapitalPool	Public	Can Modify State	initializer
setup	External	Can Modify State	onlyOwner
setData	External	Can Modify State	onlyOwner
addStakersPoolData	External	Can Modify State	onlyOwner
_getTokenToBase	Private	-	-
getStakingPercentageX10000	External	-	allowedCaller
getTVLinBaseToken	External	-	-
_getCapInBaseToken	Private	-	-
_getDeltaCoverAmtInBaseToken	Private	-	-
getCapacityInfo	External	-	-
_getFreeCapacity	Private	-	-
_getCoverAmtPPinBaseToken	Private	-	-
canBuyCoverPerProduct	External	-	-
canBuyCover	External	-	-
buyCoverPerProduct	External	Can Modify State	allowedCaller
_getExactToken2PaymentToken	Private	-	-
_settleExactPayoutFromStakers	Private	Can Modify State	-
preparePaymentforClaim	External	Can Modify State	allowedCaller

Cover			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
setup	External	Can Modify State	onlyOwner
pauseAll	External	Can Modify State	onlyOwner whenNotPaused
unPauseAll	External	Can Modify State	onlyOwner whenPaused
buyCover	External	Payable	whenNotPaused nonReentrant
depositReward	External	Payable	whenNotPaused nonReentrant
withdrawReward	External	Can Modify State	allowedCaller

			whenNotPaused nonReentrant
harvestReward	External	Can Modify State	whenNotPaused nonReentrant
getInsurRewardAmount	Public	-	-
getTokenToInsurToken	Public	-	-
getTokenToToken	Public	-	-
getTokenToMiddleToToken	Public	-	-

CoverQuotation			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
setup	External	Can Modify State	onlyOwner
getPremium	External	-	-
calculateStakingBasedCost	Internal	-	-
calculateSumOfCoverAmount	Internal	-	-
calculatePortfolioBasedPremium	Internal	-	-
calculateTotalUnitCost	Internal	-	-
calculateTotalRiskMargin	Internal	-	-
calculateWeightedAvgOfCoverPeriod	Internal	-	-
calculateNetPremium	Internal	-	-
calculateGrossPremium	Internal	-	-
calculateFinalPremium	Internal	-	-
calculateDiscountedPremium	Internal	-	-

CoverQuotationData			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
setup	External	Can Modify State	onlyOwner
getUnitCost	Public	-	-
updateAllUnitCost	External	Can Modify State	allowedCaller
updateUnitCostOfOneProduct	External	Can Modify State	allowedCaller
getCorrelation	Public	-	-

updateAllCorrelationMatrix	Public	Can Modify State	allowedCaller
updateCorrelationOfOneProduct	Public	Can Modify State	allowedCaller
updateCorrelationValue	Public	Can Modify State	allowedCaller
getTheta1Percent	Public	-	-
getTheta2Percent	Public	-	-
getRiskMarginPercent	Public	-	-
getExpenseMarginPercent	Public	-	-
getPremiumDiscountPercentX10000	Public	-	-
setTheta1Percent	Public	Can Modify State	allowedCaller
setTheta2Percent	Public	Can Modify State	allowedCaller
setRiskMarginPercent	Public	Can Modify State	allowedCaller
setExpenseMarginPercent	Public	Can Modify State	allowedCaller
setPremiumDiscountPercentX10000	Public	Can Modify State	allowedCaller

CoverData			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
setup	External	Can Modify State	onlyOwner
getAllCoverOwnerCount	Public	-	-
hasCoverOwner	Public	-	-
addCoverOwner	Public	Can Modify State	-
getAllCoverOwnerList	Public	-	-
getCoverCount	Public	-	validAddress
increaseCoverCount	External	Can Modify State	allowedCaller validAddress
getCoverBeginTimestamp	Public	-	validCoverId
setCoverBeginTimestamp	External	Can Modify State	allowedCaller validCoverId
getCoverEndTimestamp	Public	-	validCoverId
setCoverEndTimestamp	External	Can Modify State	allowedCaller validCoverId
getCoverProductId	External	-	validCoverId
setCoverProductId	External	Can Modify State	allowedCaller validCoverId
getCoverDurationInDays	External	-	validCoverId
setCoverDurationInDays	External	Can Modify State	allowedCaller validCoverId
getCoverCurrency	Public	-	validCoverId

setCoverCurrency	External	Can Modify State	allowedCaller validCoverId
getCoverAmount	Public	-	validCoverId
setCoverAmount	External	Can Modify State	allowedCaller validCoverId
getCoverStatus	Public	-	validCoverId
setCoverStatus	External	Can Modify State	allowedCaller validCoverId
getTotalInsurTokenEarned	External	-	validAddress
increaseTotalInsurTokenEarned	External	Can Modify State	allowedCaller validAddress
decreaseTotalInsurTokenEarned	External	Can Modify State	allowedCaller validAddress
getTotalInsurTokenRewardAmount	External	-	-
increaseTotalInsurTokenRewardAmount	External	Can Modify State	allowedCaller
decreaseTotalInsurTokenRewardAmount	External	Can Modify State	allowedCaller

CoverConfig			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
setup	External	Can Modify State	onlyOwner
addCurrency	Public	Can Modify State	allowedCaller
getAllValidCurrencyArray	External	-	-
isValidCurrency	External	-	-
removeCurrency	External	Can Modify State	allowedCaller
getMinDurationInDays	External	-	-
getMaxDurationInDays	External	-	-
setMinDurationInDays	Public	Can Modify State	allowedCaller
setMaxDurationInDays	Public	Can Modify State	allowedCaller
getMinAmountOfCurrency	External	-	-
getMaxAmountOfCurrency	External	-	-
setMinAmountOfCurrency	Public	Can Modify State	allowedCaller
setMaxAmountOfCurrency	Public	Can Modify State	allowedCaller
getMaxClaimDurationInDaysAfterExpired	External	-	-
setMaxClaimDurationInDaysAfterExpired	Public	Can Modify State	allowedCaller
getInsurTokenRewardPercentX10000	External	-	-

setInsurTokenRewardPercentX10000	External	Can Modify State	allowedCaller

CoverQuery			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
setup	External	Can Modify State	onlyOwner
getCoverDetails	External	-	-
getTotalCoverAmount	External	-	-
getTotalCoverCount	External	-	-
canCoverBeClaimed	Public	-	-

Claim			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
setup	External	Can Modify State	onlyOwner
pauseAll	External	Can Modify State	onlyOwner whenNotPaused
unPauseAll	External	Can Modify State	onlyOwner whenPaused
<Receive Ether>	External	Payable	-
stake	External	Payable	whenNotPaused nonReentrant
unstake	External	Payable	whenNotPaused nonReentrant
getClaimFeeAmount	Public	-	-
getAdjustedClaimStatus	Public	-	-
getClaimDetails	External	-	-
getClaimVotingDetails	External	-	-
getAssessorVotingDetails	External	-	-
claim	External	Payable	whenNotPaused nonReentrant
vote	External	Can Modify State	whenNotPaused nonReentrant
getComplainFeeAmount	Public	-	-

complain	External	Payable	whenNotPaused nonReentrant
deposit	External	Payable	whenNotPaused nonReentrant
withdraw	External	Can Modify State	whenNotPaused nonReentrant
harvest	External	Can Modify State	whenNotPaused nonReentrant

ClaimManager			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
setup	External	Can Modify State	onlyOwner
switchFromSubmittedToInvestigating	Public	Can Modify State	allowedCaller nonReentrant
switchFromInvestigatingToPrepareForVoting	Public	Payable	allowedCaller nonReentrant
switchFromPrepareForVotingToVoting	Public	Can Modify State	allowedCaller nonReentrant
switchFromVotingToComplaining	Public	Can Modify State	allowedCaller nonReentrant
switchFromComplainingToVerdict	Public	Can Modify State	allowedCaller nonReentrant
_switchToABDiscretion	Internal	Can Modify State	-
switchFromABDiscretionToVerdict	Public	Can Modify State	allowedCaller nonReentrant
_switchToVerdict	Internal	Can Modify State	-
switchToPayout	Public	Payable	allowedCaller nonReentrant
switchToPaid	Public	Can Modify State	allowedCaller nonReentrant

ClaimAdminOps			
Function Name	Visibility	Mutability	Modifiers

initialize	Public	Can Modify State	initializer
setup	External	Can Modify State	onlyOwner
getRewardAmount	External	-	-
depositReward	External	Payable	nonReentrant
withdrawReward	External	Can Modify State	allowedCaller nonReentrant
setCurrentProcessingClaimId	External	Can Modify State	allowedCaller nonReentrant
setMoveToNextClaimFlag	External	Can Modify State	allowedCaller nonReentrant
findNextPendingClaimId	External	-	-
switchToInvestigating	External	Can Modify State	allowedCaller nonReentrant
switchToPrepareForVoting	External	Can Modify State	allowedCaller nonReentrant
switchToVoting	External	Can Modify State	allowedCaller nonReentrant
switchToComplaining	External	Can Modify State	allowedCaller nonReentrant
switchFromComplainingToVerdict	External	Can Modify State	allowedCaller nonReentrant
switchFromABDiscretionToVerdict	External	Can Modify State	allowedCaller nonReentrant
preparePaymentForPayout	External	Can Modify State	allowedCaller nonReentrant
switchToPaid	External	Can Modify State	allowedCaller nonReentrant

ClaimReward			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
setup	External	Can Modify State	onlyOwner
recalculateAssessor	External	Can Modify State	allowedCaller
getTotalWithdrawableINSURRewardAmount	External	-	-

getClaimINSURRewardAmount	Public	-	-
_calculateAssessor	Internal	-	-

ClaimRewardData			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
setup	External	Can Modify State	onlyOwner
getVotedClaimIdArrayCount	External	-	-
getVotedClaimIdByIndex	External	-	-
getVotedClaimIdArray	External	-	-
addVotedClaimId	External	Can Modify State	allowedCaller
getLastCalculatedClaimIdPosition	External	-	-
setLastCalculatedClaimIdPosition	External	Can Modify State	allowedCaller
getTotalWithdrawedRewardAmount	External	-	-
addTotalWithdrawedRewardAmount	External	Can Modify State	allowedCaller
getWithdrawableRewardAmount	External	-	-
addWithdrawableRewardAmount	External	Can Modify State	allowedCaller
substractWithdrawableRewardAmount	External	Can Modify State	allowedCaller

ClaimData			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
setup	External	Can Modify State	onlyOwner
getClaimCount	External	-	-
increaseClaimCount	External	Can Modify State	allowedCaller
getCoverId	External	-	-
setCoverId	External	Can Modify State	allowedCaller
getCoverOwner	External	-	-
setCoverOwner	External	Can Modify State	allowedCaller
getLossAmount	External	-	-
setLossAmount	External	Can Modify State	allowedCaller
getLossEventTime	External	-	-

setLossEventTime	External	Can Modify State	allowedCaller
getClaimAmount	External	-	-
setClaimAmount	External	Can Modify State	allowedCaller
getOtherClaimInfo	External	-	-
setOtherClaimInfo	External	Can Modify State	allowedCaller
getClaimStatus	External	-	-
setClaimStatus	External	Can Modify State	allowedCaller
getClaimJudgementCount	External	-	-
getClaimJudgement	External	-	-
addClaimJudgement	External	Can Modify State	allowedCaller
getClaimINSURRewardAmount	External	-	-
setClaimINSURRewardAmount	External	Can Modify State	allowedCaller
getClaimPayoutAmount	External	-	-
addClaimPayoutAmount	External	Can Modify State	allowedCaller
getClaimIdCount	External	-	-
getClaimIdByIndex	External	-	-
getClaimIdList	External	-	-
addClaimId	External	Can Modify State	allowedCaller
getAccumulatedPayoutAmount	External	-	-
addAccumulatedPayoutAmount	External	Can Modify State	allowedCaller
getWithdrawablePayoutAmount	External	-	-
addWithdrawablePayoutAmount	External	Can Modify State	allowedCaller
substractWithdrawablePayoutAmount	External	Can Modify State	allowedCaller

ClaimVotingData			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
setup	External	Can Modify State	onlyOwner
getClaimAssessorCount	External	-	-
increaseClaimAssessorCount	External	Can Modify State	allowedCaller
getClaimForVoteCount	External	-	-
increaseClaimForVoteCount	External	Can Modify State	allowedCaller
getClaimAgainstVoteCount	External	-	-

increaseClaimAgainstVoteCount	External	Can Modify State	allowedCaller
getClaimStartTimestamp	External	-	-
setClaimStartTimestamp	External	Can Modify State	allowedCaller
getClaimEndTimestamp	External	-	-
setClaimEndTimestamp	External	Can Modify State	allowedCaller
getClaimEndTimestampExtended	External	-	-
setClaimEndTimestampExtended	External	Can Modify State	allowedCaller
getClaimComplainStartTimestamp	External	-	-
setClaimComplainStartTimestamp	External	Can Modify State	allowedCaller
getClaimComplainEndTimestamp	External	-	-
setClaimComplainEndTimestamp	External	Can Modify State	allowedCaller
getClaimComplainCount	External	-	-
getClaimComplain	External	-	-
addClaimComplain	External	Can Modify State	allowedCaller
getClaimAssessorArray	External	-	-
addClaimAssessor	External	Can Modify State	allowedCaller
getClaimAssessorForOrAgainstFlag	External	-	-
setClaimAssessorForOrAgainstFlag	External	Can Modify State	allowedCaller
getClaimAssessorNumOfVotes	External	-	-
setClaimAssessorNumOfVotes	External	Can Modify State	allowedCaller
getClaimStartBlockNumber	External	-	-
setClaimStartBlockNumber	External	Can Modify State	allowedCaller

ClaimConfig			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
setup	External	Can Modify State	onlyOwner
getClaimFeeRateX10000	External	-	-
setClaimFeeRateX10000	External	Can Modify State	allowedCaller validPercent
getComplainFeeRateX10000	External	-	-
setComplainFeeRateX10000	External	Can Modify State	allowedCaller validPercent

getVotingTimeDefault	External	-	-
setVotingTimeDefault	External	Can Modify State	allowedCaller validTimePeriod
getVotingTimeExtended	External	-	-
setVotingTimeExtended	External	Can Modify State	allowedCaller validTimePeriod
getComplainTime	External	-	-
setComplainTime	External	Can Modify State	allowedCaller validTimePeriod
getVotingMaxWeightRateX10000	External	-	-
setVotingMaxWeightRateX10000	External	Can Modify State	allowedCaller validPercent
getVotingQuorumRateX10000	External	-	-
setVotingQuorumRateX10000	External	Can Modify State	allowedCaller validPercent
getVotingMajorityRateX10000	External	-	-
setVotingMajorityRateX10000	External	Can Modify State	allowedCaller validPercent
getClaimAssessorMinUnstakeTime	External	-	-
setClaimAssessorMinUnstakeTime	External	Can Modify State	allowedCaller validTimePeriod

ClaimVoting			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
setup	External	Can Modify State	onlyOwner
pauseAll	External	Can Modify State	onlyOwner whenNotPaused
unPauseAll	External	Can Modify State	onlyOwner whenPaused
startVoting	External	Can Modify State	allowedCaller whenNotPaused nonReentrant
hasVoted	Public	-	-
isQuorumAchieved	Public	-	-
getOutcomeStatus	Public	-	-

isVotingCompleted	Public	-	-
isVotingSuccessful	Public	-	-
getVotingEndTimstamp	Public	-	-
vote	Public	Can Modify State	allowedCaller whenNotPaused nonReentrant
startComplaining	Public	Can Modify State	allowedCaller whenNotPaused nonReentrant
hasAnyComplain	Public	-	-
isComplainingCompleted	Public	-	-
canComplain	Public	-	-
complain	External	Can Modify State	allowedCaller whenNotPaused nonReentrant

ClaimAssessor			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
setup	External	Can Modify State	onlyOwner
pauseAll	External	Can Modify State	onlyOwner whenNotPaused
unPauseAll	External	Can Modify State	onlyOwner whenPaused
getTotalNumOfVotes	External	-	-
getTotalNumOfAssessors	External	-	-
getNumOfVotes	External	-	-
getLatestVoteTimestamp	External	-	-
setLatestVoteTimestamp	External	Can Modify State	allowedCaller
getVoteStakePeriodEndTime	Public	-	-
getAssessorPriorNumOfVotes	External	-	-
getOverviewPriorNumOfAssessorAndVotes	External	-	-
increaseVotes	External	Can Modify State	allowedCaller whenNotPaused
decreaseVotes	External	Can Modify State	allowedCaller

			whenNotPaused
moveDelegate	Internal	Can Modify State	-
writeCheckpoint	Internal	Can Modify State	-
updateOvvwCheckPoint	Internal	Can Modify State	-

GovernorAlpha			
Function Name	Visibility	Mutability	Modifiers
quorumVotes	Public	-	-
proposalThreshold	Public	-	-
proposalMaxOperations	Public	-	-
votingDelay	Public	-	-
votingPeriod	Public	-	-
initializeStakersPool	Public	Can Modify State	initializer
propose	Public	Can Modify State	-
queue	Public	Can Modify State	-
_queueOrRevert	Internal	Can Modify State	-
execute	Public	Payable	-
cancel	Public	Can Modify State	-
getActions	Public	-	-
getReceipt	Public	-	-
state	Public	-	-
castVote	Public	Can Modify State	-
_castVote	Internal	Can Modify State	-
__acceptAdmin	Public	Can Modify State	-
__abdicate	Public	Can Modify State	-
__queueSetTimelockPendingAdmin	Public	Can Modify State	-
__executeSetTimelockPendingAdmin	Public	Can Modify State	-

LPToken			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
initializeLPToken	Public	Can Modify State	initializer

setup	External	Can Modify State	onlyOwner
rewardDebtOf	External	-	-
burnableAmtOf	External	-	-
pauseAll	External	Can Modify State	onlyOwner whenNotPaused
unPauseAll	External	Can Modify State	onlyOwner whenPaused
mint	External	Can Modify State	onlyMinter whenNotPaused
burn	External	Can Modify State	onlyBurner
proposeToBurn	External	Can Modify State	whenNotPaused onlyBurner
_beforeTokenTransfer	Internal	Can Modify State	-

StakingV2Controller			
Function Name	Visibility	Mutability	Modifiers
initializeStakingV2Controller	Public	Can Modify State	initializer
setTokenToLPTokenMap	External	Can Modify State	onlyOwner
setup	External	Can Modify State	onlyOwner
setStakeInfo	External	Can Modify State	onlyOwner onlyAllowedToken
pauseAll	External	Can Modify State	onlyOwner whenNotPaused
unPauseAll	External	Can Modify State	onlyOwner whenPaused
stakeTokens	External	Payable	whenNotPaused nonReentrant onlyAllowedToken
proposeUnstake	External	Can Modify State	nonReentrant whenNotPaused onlyAllowedToken
withdrawTokens	External	Can Modify State	nonReentrant whenNotPaused onlyAllowedToken
claimRewardsFromPools	External	Can Modify State	whenNotPaused nonReentrant
showRewardsFromPools	External	-	-

4.2 Contract Information

The contract has not been deployed to the mainnet.

4.3 Code Audit

4.3.1 High-risk vulnerabilities

4.3.1.1 Reordering attack risk

When the owner calls preparePaymentForPayout, it will go to uniswap to calculate the required amountIn, and then perform the swap operation according to the amountIn. There is a risk of rearrangement attacks that may cause losses in the InsurAce pool. It is recommended to check the slippage of swap.

Reference：

https://www.odaily.com/post/5162888

https://medium.com/coinmonks/demystify-the-dark-forest-on-ethereum-sandwich-attacks-5a3ae c9fa33e

- contracts/pool/StakersPool.sol

```
function claimPayout(
        address _fromToken,
        address _paymentToken,
        uint256 _settleAmtPT,
        address _claimTo,
        uint256 _claimId
    ) external override allowedCaller {
        require(_fromToken == poolToken, "CP:1");
        if (_settleAmtPT == 0) {
            return;
        }
        uint256 temp = _getTokenforExactPaymentToken(_fromToken, _paymentToken, _settleAmtPT);
        uint256 amountInMax = Math.min(stakedAmount, temp.mul(11).div(10));
        uint256 convertOut = _convertTokenforExactPaymentToken(_fromToken, _paymentToken, _settleAmtPT, amountInMax);
        stakedAmount = stakedAmount.sub(convertOut);
        claimPayouts.push(convertOut);
        claimPayoutsClaimId.push(_claimId);
        _transferTokenTo(_paymentToken, _settleAmtPT, _claimTo, _claimId);

    }

function _convertTokenforExactPaymentToken(
        address _tokenFrom,
        address _tokenTo,
        uint256 _amountOut,
        uint256 _amountInMax
    ) private returns (uint256) {
        require(_tokenFrom != _tokenTo, "CT2EPT:1");
        address[] memory path = new address[](2);
        uint256[] memory ret;

        if (_tokenFrom == Constant.ETHTOKENADDRESS) {
            path[0] = uniswapRouter.WETH();
            path[1] = _tokenTo;
            ret = uniswapRouter.swapETHForExactTokens{value: _amountInMax}(
                _amountOut,
                path,
                address(this),
                block.timestamp + 120 // solhint-disable-line not-rely-on-time
            );
            return ret[0];
        }

        if (_tokenTo == Constant.ETHTOKENADDRESS) {
            path[0] = _tokenFrom;
            path[1] = uniswapRouter.WETH();
            IERC20Upgradeable(path[0]).approve(Constant.UNISWAPV2_ROUTER_ADDRESS, _amountInMax);
            ret = uniswapRouter.swapTokensForExactETH(
                _amountOut,
                _amountInMax,
                path,
                address(this),
                block.timestamp + 120 // solhint-disable-line not-rely-on-time
            );
            return ret[0];
        }

        path[0] = _tokenFrom;
        path[1] = _tokenTo;

        IERC20Upgradeable(path[0]).approve(Constant.UNISWAPV2_ROUTER_ADDRESS, _amountInMax);
        ret = uniswapRouter.swapTokensForExactTokens(
            _amountOut,
            _amountInMax,
            path,
            address(this),
            block.timestamp + 120 // solhint-disable-line not-rely-on-time
        );
        return ret[0];
    }
```

- contracts/pool/StakersPool.sol

```
function _getTokenforExactPaymentToken(
        address _tokenFrom,
        address _tokenTo,
        uint256 _amount
    ) private view returns (uint256) {
        if (_tokenFrom == _tokenTo) {
            return _amount;
        }
        address[] memory path = new address[](2);
        if (_tokenFrom == Constant.ETHTOKENADDRESS) {
            path[0] = uniswapRouter.WETH();
        } else {
            path[0] = _tokenFrom;
        }

        if (_tokenTo == Constant.ETHTOKENADDRESS) {
            path[1] = uniswapRouter.WETH();
        } else {
            path[1] = _tokenTo;
        }
        uint256[] memory ret = uniswapRouter.getAmountsIn(_amount, path);
        return ret[0];
    }
```

Fix Status: The issues has been fixed.

4.3.1.2 Missing permission check

The addCoverOwner function does not perform permission checking, any user can call this function to add owner. It is recommended to add permission check code.

- contracts/cover/CoverData.sol

```
function addCoverOwner(address owner) public {
    require(owner != address(0), "ACO: 1");
    require(!allCoverOwnerFlagMap[owner], "ACO: 2");
    allCoverOwnerList.push(owner);
    allCoverOwnerFlagMap[owner] = true;
}
```

Fix Status: The issues has been fixed.

4.3.2 Medium-risk vulnerabilities

4.3.2.1 DoS issue

_getDelAccuRwAmtPS has 3 while loop nestings, which will be affected by the parameters of lastScheduleCounter, gRewardTokenRatePerStakedTokenArray, _unstakeLockArrayBlockPerStaker, and dos due to more users or more mining cycles added.

- contracts/staking/ScheduledMiningProgram.sol

```
function _getDelAccuRwAmtPS(
        uint256 _lastCalculatedBlockPerStaker,
        uint256 _stakedAmtPerStaker,
        uint256[] memory _unstakeLockArrayBlockPerStaker,
        uint256[] memory _unstakeLockArrayAmtPerStaker
```

```
    ) private view returns (uint256) {
        console.log("getDeltaAccumulativeRewardAmtPerStaker++");
        console.log(_lastCalculatedBlockPerStaker);
        console.log(_stakedAmtPerStaker);
        console.log(_unstakeLockArrayBlockPerStaker.length);
        uint256 retV = 0;
        // go thru the list of all schedules
        uint256 scheduleIndex = lastScheduleCounter;
        while (scheduleIndex >= 1) {
            if (_lastCalculatedBlockPerStaker >= endMiningBlockPerSchedule[scheduleIndex]) {
                break;
            }
            // narrow down block delta
            uint256 minWall = Math.max(_lastCalculatedBlockPerStaker, startMiningBlockPerSchedule[scheduleIndex]);
            uint256 maxWall = Math.min(block.number, endMiningBlockPerSchedule[scheduleIndex]);
            console.log("minWall: ", minWall);
            console.log("maxWall: ", maxWall);
            if (minWall >= maxWall) {
                scheduleIndex = scheduleIndex.sub(1);
                continue;
            }
            uint256 rateChangeIndex = gRewardTokenRatePerStakedTokenArray.length;
            if (rateChangeIndex == 0) {
                break;
            }
            uint256 rewardAccumulatedBetweenWalls = 0;
            while (rateChangeIndex > 0) {
                uint256 blockNumber = gRewardTokenRatePerStakedTokenArray[rateChangeIndex - 1];
                console.log("blockNumber: ", blockNumber);
                if (blockNumber >= maxWall) {
                    rateChangeIndex = rateChangeIndex.sub(1);
                    continue;
                }
                if (blockNumber >= minWall) {
                    uint256 delta = _getDeltaAccumulativeRewardsWithFixRatePerStaker(blockNumber, maxWall,
gRewardTokenRatePerStakedTokenMap[blockNumber], _stakedAmtPerStaker, _unstakeLockArrayBlockPerStaker,
_unstakeLockArrayAmtPerStaker);
                    rewardAccumulatedBetweenWalls = delta.add(rewardAccumulatedBetweenWalls);
                    maxWall = blockNumber;
                    rateChangeIndex = rateChangeIndex.sub(1);
                    continue;
                }
                if (blockNumber < minWall) {
                    uint256 delta = _getDeltaAccumulativeRewardsWithFixRatePerStaker(minWall, maxWall,
gRewardTokenRatePerStakedTokenMap[blockNumber], _stakedAmtPerStaker, _unstakeLockArrayBlockPerStaker,
_unstakeLockArrayAmtPerStaker);
                    rewardAccumulatedBetweenWalls = delta.add(rewardAccumulatedBetweenWalls);
                    break;
                }
            }
            retV = rewardAccumulatedBetweenWalls.add(retV);
            scheduleIndex = scheduleIndex.sub(1);
        }
        return retV;
    }
```

Fix Status: This issue has been fixed,

4.3.3 Low-risk vulnerabilities

4.3.3.1 Excessive authority issue

Admin has permission to add sender, There is a issues of excessive authority. It is recommended to set Owner to Timelock contract or governance contract.

- contracts/token/INSURToken.sol

```
function addSender(address _from) external onlyAdmin {
        if (1 == transferFromAllowedList[_from]) {
            return;
        }
        membersFrom.push(_from);
        transferFromAllowedList[_from] = 1;
  }
```

The admin can remove the sender arbitrarily, and there is a risk of denial of service. When the admin adds too many senders, the data in the memberFrom array will be very large, so when the removeSender is removed, the depth of the for loop call will be too large, resulting in The call fails. It is recommended to change memberFrom to storage in the way of mapping, and use address as the key to avoid dos caused by this type of looping to obtain data.

- contracts/token/INSURToken.sol

```
function removeSender(address _from) external onlyAdmin {
        uint256 arrayLength = membersFrom.length;
        uint256 indexToBeDeleted;
        bool toDelete = false;
        for (uint256 i = 0; i < arrayLength; i++) {
            if (membersFrom[i] == _from) {
                indexToBeDeleted = i;
                toDelete = true;
                break;
            }
        }
        if (!toDelete) {
            return;
        }
        // if index to be deleted is not the last index, swap position.
        if (indexToBeDeleted < arrayLength - 1) {
            membersFrom[indexToBeDeleted] = membersFrom[arrayLength - 1];
        }
        // we can now reduce the array length by 1
        membersFrom.pop();
        delete transferFromAllowedList[_from];
    }
```

MINTER can call mint arbitrarily, and there is no upper limit for minting.

- contracts/token/INSURToken.sol

```
 function mint(address to, uint256 amount) public virtual {
        require(hasRole(MINTER_ROLE, _msgSender()), "ERC20PresetMinterPauser: must have minter role to mint");
        _mint(to, amount);
    }
```

Fix Status: This issue has been confirmed, after communication and feedback, the minting and Owner permissions may be transferred to address(0) in the future.

Owner can set lpTokenMinter and lpTokenBurner. The roles of lpTokenMinter and lpTokenBurner can mint and burn the user's LP. There is a issues of excessive authority. It is recommended to set Owner to Timelock contract or governance contract. And make sure the lpTokenMinter and lpTokenBurner cannot be EOA account.

- contracts/token/LPToken.sol

```
function setup(address _lpTokenMinter, address _lpTokenBurner) external onlyOwner {
        require(_lpTokenMinter != address(0), "S:1");
        lpTokenMinter = _lpTokenMinter;
        require(_lpTokenBurner != address(0), "S:2");
        lpTokenBurner = _lpTokenBurner;
    }
```

Fix Status: This issue has been communicated back to the project team. The project team is aware of this and will adopt governance mechanism to secure the permission when the governance module goes live.

4.3.3.2 DoS issue

The incoming _callers will add data to allowedCallersArray[_callee]. If too many _callers are added at one time, it will cause Out of Gas. When there are too many data in allowedCallersArray[_callee], the setAllowdCallersPerCallee function will DoS. It is recommended to set the data Use the mapping method to store instead, avoid using the for loop to find the value.

- contracts/secmatrix/SecurityMatrix.sol

```
function addAllowdCallersPerCallee(address _callee, address[] memory _callers) external onlyOwner {
        require(_callers.length != 0, "AACPC:1");
        require(allowedCallersArray[_callee].length != 0, "AACPC:2");

        for (uint256 index = 0; index < _callers.length; index++) {
            console.log("_callers index: ", _callers[index], index);
            allowedCallersArray[_callee].push(_callers[index]);
            allowedCallersMap[_callee][_callers[index]] = 1;
        }
    }
```

- contracts/secmatrix/SecurityMatrix.sol

```
function setAllowdCallersPerCallee(address _callee, address[] memory _callers) external onlyOwner {
        console.log("_callee: ", _callee);
        console.log("_callers.length: ", _callers.length);
        require(_callers.length != 0, "SACPC:1");
        // check if callee exist
        if (allowedCallersArray[_callee].length == 0) {
            // not exist, so add callee
            allowedCallees.push(_callee);
        } else {
            // if callee exist, then purge data
            for (uint256 i = 0; i < allowedCallersArray[_callee].length; i++) {
                delete allowedCallersMap[_callee][allowedCallersArray[_callee][i]];
            }
            delete allowedCallersArray[_callee];
        }
        // and overwrite
        for (uint256 index = 0; index < _callers.length; index++) {
            console.log("_callers index: ", _callers[index], index);
            allowedCallersArray[_callee].push(_callers[index]);
            allowedCallersMap[_callee][_callers[index]] = 1;
        }
    }
```

Fix Status: This issue has been communicated back to project team. The project team is aware of this issue and the method will only be used by admin when setting up security matrix.

The "setAllowdCallersPerCallee" method will be used to create security matrix entries, and the "addAllowdCallersPerCallee" method will be used to add delta matrix if needed.

4.3.3.3 Repeatable call risk

If Owner call setupVestors function multiple times, there will be duplicate vestors in the vestor array.When the setupVestors is called multiple times, if the vestor calls withdrawRewardPV intentionally or unintentionally during the calling process, initRewardPV and insurVestingTotalPV may get unexpected values.

If setupVestors can be called multiple times, then when the owner is called, the vestor also calls withdrawRewardPV. In this case, the gas price of calling withdrawRewardPV is higher than that of calling setupVestors. Will execute withdrawRewardPV first, and then execute setupVestors, the data will appear unexpected. Competitive conditions similar to approve.

- contracts/fixedvesting/FixedVesting.sol

```
function setupVestors(
        address[] memory _vestors,
        uint256[] memory _vestingRewardPV,
        uint256[] memory _initRewardPV
    ) external onlyOwner {
        require(_vestors.length == _vestingRewardPV.length, "AV:1");
        require(_initRewardPV.length == _vestingRewardPV.length, "AV:2");
        for (uint256 i = 0; i < _vestors.length; i++) {
            address vestor = _vestors[i];
            vestors.push(vestor);
            initRewardPV[vestor] = _initRewardPV[i];
            insurVestingTotalPV[vestor] = _vestingRewardPV[i];
        }
    }
```

Fix Status: This issue has been fixed.

4.3.3.4 Overflow risk

safemath should be used to calculate the length of the array to avoid overflow issues: if Currency is not added, the removal may cause overflow issues.

- contracts/cover/CoverConfig.sol

```
function removeCurrency(address currency) external allowedCaller {
        require(currency != address(0), "RC: 1");
        uint256 arrayLength = currencyValidAddressArray.length;
        uint256 indexToBeDeleted;
        bool toDelete = false;
        for (uint256 i = 0; i < arrayLength; i++) {
            if (currencyValidAddressArray[i] == currency) {
                indexToBeDeleted = i;
                toDelete = true;
                break;
            }
        }
        if (!toDelete) {
            require(toDelete, "RC: 1");
        }
        // if index to be deleted is not the last index, swap position.
        if (indexToBeDeleted < arrayLength - 1) {
            currencyValidAddressArray[indexToBeDeleted] = currencyValidAddressArray[arrayLength - 1];
        }
        // we can now reduce the array length by 1
        currencyValidAddressArray.pop();
        delete currencyValidAddressMap[currency];
    }
```

Fix Status: This issue has been fixed.

4.3.3.5 FlashLoan attack risk

Unstake is judged by >= when there are already voting tasks. If claimsAssessorMinUnstakeTime is 0, then there will be a issue of using flashloan to vote.

- contracts/claim/Claim.sol

```
function unstake(address insurTokenAddress, uint256 insurAmount) external payable whenNotPaused nonReentrant {
        require(insurTokenAddress != address(0), "USTK: 1");

        address payable assessor = _msgSender();
        ClaimReward(crw).recalculateAssessor(assessor);

        bool canUnstake = false;
        uint256 latestVoteTimestamp = ClaimAssessor(asr).getLatestVoteTimestamp(assessor);
        if (latestVoteTimestamp == 0) {
            canUnstake = true;
        } else {
            if (
                block.timestamp >= ClaimAssessor(asr).getVoteStakePeriodEndTime(assessor) // solhint-disable-line
not-rely-on-time
            ) {
                canUnstake = true;
            }
        }

        require(canUnstake, "USTK: 2");
        require(insurAmount <= ClaimAssessor(asr).getNumOfVotes(assessor), "USTK: 3");
        require(IERC20Upgradeable(insurTokenAddress).balanceOf(address(this)) >= insurAmount, "USTK: 4");

        ClaimAssessor(asr).decreaseVotes(assessor, insurAmount);

        IERC20Upgradeable(insurTokenAddress).safeTransfer(assessor, insurAmount);
    }
```

Fix Status: This issue has been fixed.

4.3.4 Enhancement Suggestions

4.3.4.1 Token compatibility risk

IERC20Upgradeable(stakedToken).safeTransferFrom(_msgSender(), address(this), _amount); The transfer operation of an external token is adopted. It is recommended to pay attention to the compatibility of the project and the token when adding a new token, such as: token return Value issues, fake token recharge issues, compatibility issues with deflationary tokens, etc.

- contracts/staking/StakeOps.sol

```
function stakeTokens(uint256 _amount, address _token) external payable whenNotPaused nonReentrant {
    require(IMiningProgram(iMiningProgram).canStake(_amount), "ST:1");
        address stakedToken = StakersData(stakerDataAddr).stakedToken();
        require(_token == stakedToken, "ST:2");
        if (stakedToken == Constant.ETHTOKENADDRESS) {
            require(_amount <= msg.value, "ST:3");
        } else { require(IERC20Upgradeable(stakedToken).balanceOf(_msgSender()) >= _amount, "ST:4");
            uint256 allowanceAmt = IERC20Upgradeable(stakedToken).allowance(_msgSender(), address(this));
            require(allowanceAmt >= _amount, "ST:5");
        }

        _reCalcPerStaker();

        if (stakedToken != Constant.ETHTOKENADDRESS) {
            IERC20Upgradeable(stakedToken).safeTransferFrom(_msgSender(), address(this), _amount);
        }
        // dispatch token to pool
        if (stakedToken == Constant.ETHTOKENADDRESS) {
            IStakersPool(iStakersPool).addStkAmount{value: _amount}(stakedToken, _amount);
        } else {
     IERC20Upgradeable(stakedToken).safeTransfer(iStakersPool, _amount);
```

Fix Status: This issue has been communicated back to project team. The project team is aware of this and has already performed compatibility checks on the staking tokens, such as ETH, WETH, USDC, USDT, DAI, and INSUR, which are all compatible with the relevant standards.

4.3.4.2 Event log is missing

It is recommended to add an event to record securityMatrix changes, applicable to all set functions.

```
function setup(address _securityMatrix) external onlyOwner {
        require(_securityMatrix != address(0), "S:1");
        securityMatrix = _securityMatrix;
  }
```

Fix Status: This issue has been communicated back to project team. The project team will add more event logs in their development, including not limited to "setup".

4.3.4.3 Redundant code

The `if (!toDelete) {require(toDelete, "RC: 1"); }` code can be simplified to `require(toDelete, "RC: 1");`.

- contracts/cover/CoverConfig.sol

```
function removeCurrency(address currency) external allowedCaller {
        require(currency != address(0), "RC: 1");
        uint256 arrayLength = currencyValidAddressArray.length;
        uint256 indexToBeDeleted;
        bool toDelete = false;
        for (uint256 i = 0; i < arrayLength; i++) {
            if (currencyValidAddressArray[i] == currency) {
                indexToBeDeleted = i;
                toDelete = true;
                break;
            }
        }
        if (!toDelete) {
            require(toDelete, "RC: 1");
        }
        // if index to be deleted is not the last index, swap position.
        if (indexToBeDeleted < arrayLength - 1) {
            currencyValidAddressArray[indexToBeDeleted] = currencyValidAddressArray[arrayLength - 1];
        }
        // we can now reduce the array length by 1
        currencyValidAddressArray.pop();
        delete currencyValidAddressMap[currency];
    }
```

Fix Status: This issue has been fixed.

4.3.4.4 Hard coded issue

The external contract address is hard-coded and cannot be modified. It is recommended that the external contract adopts a changeable method to avoid the problem that the project cannot operate normally due to the upgrade of the external contract.

- common/Constant.sol

```
address public constant UNISWAPV2_ROUTER_ADDRESS = address(0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D);
```

Fix Status: This issue has been communicated back to project team. The project team is aware of this issue, and made design changes, such as adding exchange library lately, which will include token to token exchange queries from 1inch and Uniswap. In the case of address change, the ABI of the address may change accordingly, as such the project team will need to double check, and/or extend exchange library in tandem.

5. Audit Result

5.1 Conclusion

Audit Result : Low Risk

Audit Number : 0X002104190001

Audit Date : April. 19, 2021

Audit Team : SlowMist Security Team

Summary conclusion: The SlowMist security team use manual and SlowMist team's analysis tool to audit the project, 2 high-risk, 1 medium-risk, 5 low-risk vulnerabilities， 4 enhancement suggestions were found during the audit, the high-risk, medium-risk and low-risk vulnerabilities identified have been fixed or confirmed except excessive authority issue, as communicated with the project team, the owner authority will be transferred to the timelock contract along with the go-live of the governance module.

6. Statement

SlowMist issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility base on these.

For the facts that occurred or existed after the issuance, SlowMist is not able to judge the security status of this project, and is not responsible for them. The security audit analysis and other contents of this report are based on the documents and materials provided to SlowMist by the information provider till the date of the insurance this report (referred to as "provided information"). SlowMist assumes: The information provided is not missing, tampered with, deleted or concealed. If the information provided is missing, tampered with, deleted, concealed, or inconsistent with the actual situation, the SlowMist shall not be liable for any loss or adverse effect resulting therefrom. SlowMist only conducts the agreed security audit on the security situation of the project and issues this report. SlowMist is not responsible for the background and other conditions of the project.