Smart Contract
Security Audit Report

Table Of Contents

1 Executive Summary

On 2022.05.16, the SlowMist security team received the team's security audit application for AstridDao, developed the audit plan according to the agreement of both parties and the characteristics of the project, and finally issued the security audit report.

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

The test method information:

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

The vulnerability severity level information:

Level	Description
Critical	Critical severity vulnerabilities will have a significant impact on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities.
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project team should evaluate and consider whether these vulnerabilities need to be fixed.
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.

Level	Description
Suggestion	There are better practices for coding or architecture.

2 Audit Methodology

The security audit process of SlowMist security team for smart contract includes two steps:

Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using automated analysis tools.

Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

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

Serial Number	Audit Class	Audit Subclass
1	Overflow Audit	-
2	Reentrancy Attack Audit	-
3	Replay Attack Audit	-
4	Flashloan Attack Audit	-
5	Race Conditions Audit	Reordering Attack Audit
6	Permission Vulnerability Audit	Access Control Audit
		Excessive Authority Audit

Serial Number	Audit Class	Audit Subclass
7	Security Design Audit	External Module Safe Use Audit
		Compiler Version Security Audit
		Hard-coded Address Security Audit
		Fallback Function Safe Use Audit
		Show Coding Security Audit
		Function Return Value Security Audit
		External Call Function Security Audit
		Block data Dependence Security Audit
		tx.origin Authentication Security Audit
8	Denial of Service Audit	-
9	Gas Optimization Audit	-
10	Design Logic Audit	-
11	Variable Coverage Vulnerability Audit	-
12	"False Top-up" Vulnerability Audit	-
13	Scoping and Declarations Audit	-
14	Malicious Event Log Audit	-
15	Arithmetic Accuracy Deviation Audit	-
16	Uninitialized Storage Pointer Audit	-

3 Project Overview

3.1 Project Introduction

Audit Version

https://github.com/AstridDao/contracts

commit: c0a5bc38c93828cbf7c9ad00cc67759acc908d8b

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Gas Optimization	Others	Suggestion	Confirming
N2	Token compatibility reminder	Others	Suggestion	Confirming
N3	Low-level call reminder	Others	Suggestion	Confirming
N4	Risk of excessive authority	Authority Control Vulnerability	Medium	Confirming
N5	Redundant comment code	Others	Suggestion	Confirming
N6	Excessive authority issues of owner	Authority Control Vulnerability	High	Confirming
N7	Redundant code	Others	Suggestion	Confirming
N8	Coding Standards Issues	Others	Low	Confirming

4 Code Overview

4.1 Contracts Description

The main network address of the contract is as follows:

The code was not deployed to the mainnet.

4.2 Visibility Description

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

ATIDStaking			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	Ownable
setAddresses	External	Can Modify State	onlyOwner
_insertLockedStake	Internal	Can Modify State	-
_removeLockedStake	Internal	Can Modify State	-
stakeLocked	External	Can Modify State	-
unstakeLocked	External	Can Modify State	-
increaseF_COL	External	Can Modify State	-
increaseF_BAI	External	Can Modify State	-
getPendingCOLGain	External	-	-
_getPendingCOLGain	Internal	-	-
getPendingBAIGain	External	-	-
_getPendingBAIGain	Internal	-	-
_getStakeWeight	Internal	-	-
_updateUserSnapshots	Internal	Can Modify State	-

ATIDStaking			
_sendCOLGainToUser	Internal	Can Modify State	-
_requireCallerIsVaultManager	Internal	-	-
_requireCallerIsBorrowerOperations	Internal	-	-
_requireCallerIsActivePool	Internal	-	-
_requireUserHasStake	Internal	-	-
_requireNonZeroAmount	Internal	-	-

ActivePool			
Function Name	Visibility	Mutability	Modifiers
setAddresses	External	Can Modify State	onlyOwner
getCOL	External	-	-
getBAIDebt	External	-	-
sendCOL	External	Can Modify State	-
sendCOLToCollSurplusPool	External	Can Modify State	-
sendCOLToDefaultPool	External	Can Modify State	-
sendCOLToStabilityPool	External	Can Modify State	-
increaseBAIDebt	External	Can Modify State	-
decreaseBAIDebt	External	Can Modify State	-

ActivePool			
_requireCallerIsBorrowerOperationsOrDefaultPool	Internal	-	-
_requireCallerIsBOorVaultMorSP	Internal	-	-
_requireCallerIsBOorVaultM	Internal	-	-
receiveCOL	External	Can Modify State	-
<Receive Ether>	External	Payable	-

Ownable			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
isOwner	Public	-	-
setOwnership	Public	Can Modify State	onlyOwner

CheckContract			
Function Name	Visibility	Mutability	Modifiers
checkContract	Internal	-	-

BaseMath			
Function Name	Visibility	Mutability	Modifiers

ATIDToken			
Function Name	Visibility	Mutability	Modifiers

ATIDToken			
<Constructor>	Public	Can Modify State	Ownable
addCommunityIssuance	External	Can Modify State	onlyOwner
setATIDStaking	External	Can Modify State	onlyOwner
deployLockupContract	External	Can Modify State	onlyOwner
setTransferUnlockTime	External	Can Modify State	onlyOwner
setSenderWhitelistAddress	External	Can Modify State	onlyOwner
setRecipientWhitelistAddress	External	Can Modify State	onlyOwner
totalSupply	External	-	-
balanceOf	External	-	-
getDeploymentStartTime	External	-	-
transfer	External	Can Modify State	-
allowance	External	-	-
approve	External	Can Modify State	-
transferFrom	External	Can Modify State	-
increaseAllowance	External	Can Modify State	-
decreaseAllowance	External	Can Modify State	-
sendToATIDStaking	External	Can Modify State	-
domainSeparator	Public	-	-
permit	External	Can Modify State	-
nonces	External	-	-

ATIDToken			
_chainID	Private	-	-
_buildDomainSeparator	Private	-	-
_transfer	Internal	Can Modify State	-
_mint	Internal	Can Modify State	-
_approve	Internal	Can Modify State	-
_requireValidRecipient	Internal	-	-
_requireCallerIsATIDStaking	Internal	-	-
_isTransferAllowed	Internal	-	-
name	External	-	-
symbol	External	-	-
decimals	External	-	-
version	External	-	-
permitTypeHash	External	-	-

LockupContractFactory			
Function Name	Visibility	Mutability	Modifiers
setATIDTokenAddress	External	Can Modify State	onlyOwner
deployLockupContract	External	Can Modify State	onlyOwner
isRegisteredLockup	Public	-	-
_requireATIDAddressIsSet	Internal	-	-

LockupContract			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
_getReleasedAmount	Internal	-	-
canWithdraw	Public	-	-
withdrawATID	External	Can Modify State	-

CommunityIssuance			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
setATIDTokenAddress	External	Can Modify State	onlyOwner
setStabilityPoolAddress	External	Can Modify State	onlyOwner
setRewardDistributionFractions	External	Can Modify State	onlyOwner
issueATID	External	Can Modify State	-
getAndClearAccumulatedATID	External	Can Modify State	-
_getCumulativeIssuanceFraction	Internal	-	-
sendATID	External	Can Modify State	-
_requireCallerIsStabilityPool	Internal	-	-
_requireCallerIsStabilityPoolForCollateral	Internal	-	-
_isStringEqual	Internal	-	-

GovToken

GovToken			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	Ownable
name	Public	-	-
symbol	Public	-	-
decimals	Public	-	-
totalSupply	Public	-	-
balanceOf	Public	-	-
transfer	Public	Can Modify State	onlyOwner
allowance	Public	-	-
approve	Public	Can Modify State	onlyOwner
mint	Public	Can Modify State	onlyOwner
burn	Public	Can Modify State	onlyOwner
transferFrom	Public	Can Modify State	onlyOwner
increaseAllowance	Public	Can Modify State	onlyOwner
decreaseAllowance	Public	Can Modify State	onlyOwner
_transfer	Internal	Can Modify State	-
_mint	Internal	Can Modify State	-
_burn	Internal	Can Modify State	-
_approve	Internal	Can Modify State	-
_spendAllowance	Internal	Can Modify State	-

GovToken			
_beforeTokenTransfer	Internal	Can Modify State	-
_afterTokenTransfer	Internal	Can Modify State	-

MultiStakeGetter			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
getLockedStakesIDsFromHead	Public	-	-
getLockedStakesIDsFromTail	Public	-	-

AstridBase			
Function Name	Visibility	Mutability	Modifiers
_getCompositeDebt	Internal	-	-
_getNetDebt	Internal	-	-
_getCollGasCompensation	Internal	-	-
_getEntireSystemColl	Internal	-	-
_getEntireSystemDebt	Internal	-	-
_getTCR	Internal	-	-
_checkRecoveryMode	Internal	-	-
_requireUserAcceptsFee	Internal	-	-
setMCR	Public	Can Modify State	onlyOwner
setCCR	Public	Can Modify State	onlyOwner

AstridBase			
setBAIGasCompensation	Public	Can Modify State	onlyOwner
setMinNetDebt	Public	Can Modify State	onlyOwner
setPercentageDivisor	Public	Can Modify State	onlyOwner
setBorrowingFeeFloor	Public	Can Modify State	onlyOwner
setAddresses	Public	Can Modify State	onlyOwner
setParams	Public	Can Modify State	onlyOwner

TellorCaller			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
getTellorCurrentValue	External	-	-

AstridFixedBase			
Function Name	Visibility	Mutability	Modifiers
_getCompositeDebt	Internal	-	-
_getNetDebt	Internal	-	-
_getCollGasCompensation	Internal	-	-
getEntireSystemColl	Public	-	-
getEntireSystemDebt	Public	-	-
_getTCR	Internal	-	-
_checkRecoveryMode	Internal	-	-

AstridFixedBase			
_requireUserAcceptsFee	Internal	-	-

BorrowerOperations			
Function Name	Visibility	Mutability	Modifiers
setAddresses	External	Can Modify State	onlyOwner
openVault	External	Can Modify State	-
addColl	External	Can Modify State	-
moveCOLGainToVault	External	Can Modify State	-
withdrawColl	External	Can Modify State	-
withdrawBAI	External	Can Modify State	-
repayBAI	External	Can Modify State	-
adjustVault	External	Can Modify State	-
_adjustVault	Internal	Can Modify State	-
closeVault	External	Can Modify State	-
claimCollateral	External	Can Modify State	-
_triggerBorrowingFee	Internal	Can Modify State	-
_getUSDValue	Internal	-	-
_updateVaultFromAdjustment	Internal	Can Modify State	-
_moveTokensAndCOLfromAdjustment	Internal	Can Modify State	-
_activePoolAddColl	Internal	Can Modify State	-

BorrowerOperations			
_withdrawBAI	Internal	Can Modify State	-
_repayBAI	Internal	Can Modify State	-
_requireCallerIsBorrower	Internal	-	-
_requireNonZeroAdjustment	Internal	-	-
_requireVaultIsActive	Internal	-	-
_requireVaultIsNotActive	Internal	-	-
_requireNonZeroDebtChange	Internal	-	-
_requireNotInRecoveryMode	Internal	-	-
_requireNoCollWithdrawal	Internal	-	-
_requireValidAdjustmentInCurrentMode	Internal	-	-
_requireICRisAboveMCR	Internal	-	-
_requireICRisAboveCCR	Internal	-	-
_requireNewICRisAboveOldICR	Internal	-	-
_requireNewTCRisAboveCCR	Internal	-	-
_requireAtLeastMinNetDebt	Internal	-	-
_requireValidBAIRepayment	Internal	-	-
_requireCallerIsStabilityPool	Internal	-	-
_requireSufficientBAIBalance	Internal	-	-
_requireValidMaxFeePercentage	Internal	-	-
_getNewNominalICRFromVaultChange	Internal	-	-

BorrowerOperations			
_getNewICRFromVaultChange	Internal	-	-
_getNewVaultAmounts	Internal	-	-
_getNewTCRFromVaultChange	Internal	-	-
getCompositeDebt	External	-	-

CollSurplusPool			
Function Name	Visibility	Mutability	Modifiers
setAddresses	External	Can Modify State	onlyOwner
getCOL	External	-	-
getCollateral	External	-	-
accountSurplus	External	Can Modify State	-
claimColl	External	Can Modify State	-
_requireCallerIsBorrowerOperations	Internal	-	-
_requireCallerIsVaultManager	Internal	-	-
_requireCallerIsActivePool	Internal	-	-
receiveCOL	External	Can Modify State	-
<Receive Ether>	External	Payable	-

DefaultPool			
Function Name	Visibility	Mutability	Modifiers
setAddresses	External	Can Modify State	onlyOwner

DefaultPool			
getCOL	External	-	-
getBAIDebt	External	-	-
sendCOLToActivePool	External	Can Modify State	-
increaseBAIDebt	External	Can Modify State	-
decreaseBAIDebt	External	Can Modify State	-
_requireCallerIsActivePool	Internal	-	-
_requireCallerIsVaultManager	Internal	-	-
receiveCOL	External	Can Modify State	-
<Receive Ether>	External	Payable	-

GasPool			
Function Name	Visibility	Mutability	Modifiers

HintHelpers			
Function Name	Visibility	Mutability	Modifiers
setAddresses	External	Can Modify State	onlyOwner
getRedemptionHints	External	-	-
getApproxHint	External	-	-
computeNominalCR	External	-	-
computeCR	External	-	-

MultiVaultGetter			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
getMultipleSortedVaults	External	-	-
_getMultipleSortedVaultsFromHead	Internal	-	-
_getMultipleSortedVaultsFromTail	Internal	-	-

VaultManager			
Function Name	Visibility	Mutability	Modifiers
setAddresses	External	Can Modify State	onlyOwner
setBootstrapPeriod	External	Can Modify State	onlyOwner
getVaultOwnersCount	External	-	-
getVaultFromVaultOwnersArray	External	-	-
liquidate	External	Can Modify State	-
_liquidateNormalMode	Internal	Can Modify State	-
_liquidateRecoveryMode	Internal	Can Modify State	-
_getOffsetAndRedistributionVals	Internal	-	-
_getCappedOffsetVals	Internal	-	-
liquidateVaults	External	Can Modify State	-
_getTotalsFromLiquidateVaultsSequence_RecoveryMode	Internal	Can Modify State	-

VaultManager			
_getTotalsFromLiquidateVaultsSequence_NormalMode	Internal	Can Modify State	-
batchLiquidateVaults	Public	Can Modify State	-
_getTotalFromBatchLiquidate_RecoveryMode	Internal	Can Modify State	-
_getTotalsFromBatchLiquidate_NormalMode	Internal	Can Modify State	-
_addLiquidationValuesToTotals	Internal	-	-
_sendGasCompensation	Internal	Can Modify State	-
_movePendingVaultRewardsToActivePool	Internal	Can Modify State	-
_redeemCollateralFromVault	Internal	Can Modify State	-
_redeemCloseVault	Internal	Can Modify State	-
_isValidFirstRedemptionHint	Internal	-	-
redeemCollateral	External	Can Modify State	-
getNominalICR	Public	-	-
getCurrentICR	Public	-	-
_getCurrentVaultAmounts	Internal	-	-
applyPendingRewards	External	Can Modify State	-
_applyPendingRewards	Internal	Can Modify State	-
updateVaultRewardSnapshots	External	Can Modify State	-

VaultManager			
_updateVaultRewardSnapshots	Internal	Can Modify State	-
getPendingCOLReward	Public	-	-
getPendingBAIDebtReward	Public	-	-
hasPendingRewards	Public	-	-
getEntireDebtAndColl	Public	-	-
removeStake	External	Can Modify State	-
_removeStake	Internal	Can Modify State	-
updateStakeAndTotalStakes	External	Can Modify State	-
_updateStakeAndTotalStakes	Internal	Can Modify State	-
_computeNewStake	Internal	-	-
_redistributeDebtAndColl	Internal	Can Modify State	-
closeVault	External	Can Modify State	-
_closeVault	Internal	Can Modify State	-
_updateSystemSnapshots_excludeCollRemainder	Internal	Can Modify State	-
addVaultOwnerToArray	External	Can Modify State	-
_addVaultOwnerToArray	Internal	Can Modify State	-
_removeVaultOwner	Internal	Can Modify State	-

VaultManager			
getTCR	External	-	-
checkRecoveryMode	External	-	-
_checkPotentialRecoveryMode	Internal	-	-
_updateBaseRateFromRedemption	Internal	Can Modify State	-
getRedemptionRate	Public	-	-
getRedemptionRateWithDecay	Public	-	-
_calcRedemptionRate	Internal	-	-
_getRedemptionFee	Internal	-	-
getRedemptionFeeWithDecay	External	-	-
_calcRedemptionFee	Internal	-	-
getBorrowingRate	Public	-	-
getBorrowingRateWithDecay	Public	-	-
_calcBorrowingRate	Internal	-	-
getBorrowingFee	External	-	-
getBorrowingFeeWithDecay	External	-	-
_calcBorrowingFee	Internal	-	-
decayBaseRateFromBorrowing	External	Can Modify State	-
_updateLastFeeOpTime	Internal	Can Modify State	-
_calcDecayedBaseRate	Internal	-	-

VaultManager			
_minutesPassedSinceLastFeeOp	Internal	-	-
_requireCallerIsBorrowerOperations	Internal	-	-
_requireVaultIsActive	Internal	-	-
_requireBAIBalanceCoversRedemption	Internal	-	-
_requireMoreThanOneVaultInSystem	Internal	-	-
_requireAmountGreaterThanZero	Internal	-	-
_requireTCRoverMCR	Internal	-	-
_requireAfterBootstrapPeriod	Internal	-	-
_requireValidMaxFeePercentage	Internal	-	-
getVaultStatus	External	-	-
getVaultStake	External	-	-
getVaultDebt	External	-	-
getVaultColl	External	-	-
setVaultStatus	External	Can Modify State	-
increaseVaultColl	External	Can Modify State	-
decreaseVaultColl	External	Can Modify State	-
increaseVaultDebt	External	Can Modify State	-
decreaseVaultDebt	External	Can Modify State	-

SortedVaults			
Function Name	Visibility	Mutability	Modifiers
setParams	External	Can Modify State	onlyOwner
insert	External	Can Modify State	-
_insert	Internal	Can Modify State	-
remove	External	Can Modify State	-
_remove	Internal	Can Modify State	-
reInsert	External	Can Modify State	-
contains	Public	-	-
isFull	Public	-	-
isEmpty	Public	-	-
getSize	External	-	-
getMaxSize	External	-	-
getFirst	External	-	-
getLast	External	-	-
getNext	External	-	-
getPrev	External	-	-
validInsertPosition	External	-	-
_validInsertPosition	Internal	-	-
_descendList	Internal	-	-
_ascendList	Internal	-	-

SortedVaults			
findInsertPosition	External	-	-
_findInsertPosition	Internal	-	-
_requireCallerIsVaultManager	Internal	-	-
_requireCallerIsBOorVaultM	Internal	-	-

PriceFeed			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	Ownable
setOracleAddress	Public	Can Modify State	onlyOwner
fetchCachedPrice	External	-	-
fetchPrice	Public	Can Modify State	-

StabilityPool			
Function Name	Visibility	Mutability	Modifiers
setCollateralName	External	Can Modify State	onlyOwner
setAddresses	External	Can Modify State	onlyOwner
getCOL	External	-	-
getTotalBAIDeposits	External	-	-
provideToSP	External	Can Modify State	-
withdrawFromSP	External	Can Modify State	-
withdrawCOLGainToVault	External	Can Modify State	-

StabilityPool			
_triggerATIDIssuance	Internal	Can Modify State	-
_updateG	Internal	Can Modify State	-
_computeATIDPerUnitStaked	Internal	Can Modify State	-
offset	External	Can Modify State	-
_computeRewardsPerUnitStaked	Internal	Can Modify State	-
_updateRewardSumAndProduct	Internal	Can Modify State	-
_moveOffsetCollAndDebt	Internal	Can Modify State	-
_decreaseBAI	Internal	Can Modify State	-
getDepositorCOLGain	Public	-	-
_getCOLGainFromSnapshots	Internal	-	-
getDepositorATIDGain	Public	-	-
_getATIDGainFromSnapshots	Internal	-	-
getCompoundedBAIDeposit	Public	-	-
_getCompoundedStakeFromSnapshots	Internal	-	-
_sendBAItoStabilityPool	Internal	Can Modify State	-
_sendCOLGainToDepositor	Internal	Can Modify State	-
_sendBAIToDepositor	Internal	Can Modify State	-
_updateDepositAndSnapshots	Internal	Can Modify State	-
_payOutATIDGains	Internal	Can Modify State	-
_requireCallerIsActivePool	Internal	-	-

StabilityPool			
_requireCallerIsVaultManager	Internal	-	-
_requireNoUnderCollateralizedVaults	Internal	Can Modify State	-
_requireUserHasDeposit	Internal	-	-
_requireUserHasNoDeposit	Internal	-	-
_requireNonZeroAmount	Internal	-	-
_requireUserHasVault	Internal	-	-
_requireUserHasCOLGain	Internal	-	-
_requireValidKickbackRate	Internal	-	-
receiveCOL	External	Can Modify State	-
<Receive Ether>	External	Payable	-

VaultManagerV1			
Function Name	Visibility	Mutability	Modifiers
setAddresses	External	Can Modify State	onlyOwner
getVaultOwnersCount	External	-	-
getVaultFromVaultOwnersArray	External	-	-
liquidate	External	Can Modify State	-
_liquidateNormalMode	Internal	Can Modify State	-
_liquidateRecoveryMode	Internal	Can Modify State	-

VaultManagerV1			
_getOffsetAndRedistributionVals	Internal	-	-
_getCappedOffsetVals	Internal	-	-
liquidateVaults	External	Can Modify State	-
_getTotalsFromLiquidateVaultsSequence_RecoveryMode	Internal	Can Modify State	-
_getTotalsFromLiquidateVaultsSequence_NormalMode	Internal	Can Modify State	-
batchLiquidateVaults	Public	Can Modify State	-
_getTotalFromBatchLiquidate_RecoveryMode	Internal	Can Modify State	-
_getTotalsFromBatchLiquidate_NormalMode	Internal	Can Modify State	-
_addLiquidationValuesToTotals	Internal	-	-
_sendGasCompensation	Internal	Can Modify State	-
_movePendingVaultRewardsToActivePool	Internal	Can Modify State	-
_redeemCollateralFromVault	Internal	Can Modify State	-
_redeemCloseVault	Internal	Can Modify State	-
_isValidFirstRedemptionHint	Internal	-	-
redeemCollateral	External	Can Modify State	-
getNominalICR	Public	-	-
getCurrentICR	Public	-	-

VaultManagerV1			
_getCurrentVaultAmounts	Internal	-	-
applyPendingRewards	External	Can Modify State	-
_applyPendingRewards	Internal	Can Modify State	-
updateVaultRewardSnapshots	External	Can Modify State	-
_updateVaultRewardSnapshots	Internal	Can Modify State	-
getPendingCOLReward	Public	-	-
getPendingBAIDebtReward	Public	-	-
hasPendingRewards	Public	-	-
getEntireDebtAndColl	Public	-	-
removeStake	External	Can Modify State	-
_removeStake	Internal	Can Modify State	-
updateStakeAndTotalStakes	External	Can Modify State	-
_updateStakeAndTotalStakes	Internal	Can Modify State	-
_computeNewStake	Internal	-	-
_redistributeDebtAndColl	Internal	Can Modify State	-
closeVault	External	Can Modify State	-
_closeVault	Internal	Can Modify State	-

VaultManagerV1			
_updateSystemSnapshots_excludeCollRemainder	Internal	Can Modify State	-
addVaultOwnerToArray	External	Can Modify State	-
_addVaultOwnerToArray	Internal	Can Modify State	-
_removeVaultOwner	Internal	Can Modify State	-
getTCR	External	-	-
checkRecoveryMode	External	-	-
_checkPotentialRecoveryMode	Internal	-	-
_updateBaseRateFromRedemption	Internal	Can Modify State	-
getRedemptionRate	Public	-	-
getRedemptionRateWithDecay	Public	-	-
_calcRedemptionRate	Internal	-	-
_getRedemptionFee	Internal	-	-
getRedemptionFeeWithDecay	External	-	-
_calcRedemptionFee	Internal	-	-
getBorrowingRate	Public	-	-
getBorrowingRateWithDecay	Public	-	-
_calcBorrowingRate	Internal	-	-
getBorrowingFee	External	-	-
getBorrowingFeeWithDecay	External	-	-

VaultManagerV1			
_calcBorrowingFee	Internal	-	-
decayBaseRateFromBorrowing	External	Can Modify State	-
_updateLastFeeOpTime	Internal	Can Modify State	-
_calcDecayedBaseRate	Internal	-	-
_minutesPassedSinceLastFeeOp	Internal	-	-
_requireCallerIsBorrowerOperations	Internal	-	-
_requireVaultIsActive	Internal	-	-
_requireBAIBalanceCoversRedemption	Internal	-	-
_requireMoreThanOneVaultInSystem	Internal	-	-
_requireAmountGreaterThanZero	Internal	-	-
_requireTCRoverMCR	Internal	-	-
_requireAfterBootstrapPeriod	Internal	-	-
_requireValidMaxFeePercentage	Internal	-	-
getVaultStatus	External	-	-
getVaultStake	External	-	-
getVaultDebt	External	-	-
getVaultColl	External	-	-
setVaultStatus	External	Can Modify State	-
increaseVaultColl	External	Can Modify State	-

VaultManagerV1			
decreaseVaultColl	External	Can Modify State	-
increaseVaultDebt	External	Can Modify State	-
decreaseVaultDebt	External	Can Modify State	-

WASTR			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	Ownable
_calculateDomainSeparator	Private	-	-
DOMAIN_SEPARATOR	External	-	-
totalSupply	External	-	-
<Receive Ether>	External	Payable	-
deposit	External	Payable	-
depositTo	External	Payable	-
depositToAndCall	External	Payable	-
maxFlashLoan	External	-	-
flashFee	External	-	-
flashLoan	External	Can Modify State	-
withdraw	External	Can Modify State	-
withdrawTo	External	Can Modify State	-
withdrawFrom	External	Can Modify State	-

WASTR			
approve	External	Can Modify State	-
approveAndCall	External	Can Modify State	-
permit	External	Can Modify State	-
transfer	External	Can Modify State	-
transferFrom	External	Can Modify State	-
transferAndCall	External	Can Modify State	-

BAIToken			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	Ownable
setVaultManagerAddress	External	Can Modify State	onlyOwner
setStabilityPoolAddress	External	Can Modify State	onlyOwner
setBorrowerOperationsAddress	External	Can Modify State	onlyOwner
mint	External	Can Modify State	-
burn	External	Can Modify State	-
sendToPool	External	Can Modify State	-
returnFromPool	External	Can Modify State	-
totalSupply	External	-	-
balanceOf	External	-	-
transfer	External	Can Modify State	-

BAIToken			
allowance	External	-	-
approve	External	Can Modify State	-
transferFrom	External	Can Modify State	-
increaseAllowance	External	Can Modify State	-
decreaseAllowance	External	Can Modify State	-
domainSeparator	Public	-	-
permit	External	Can Modify State	-
nonces	External	-	-
_chainID	Private	-	-
_buildDomainSeparator	Private	-	-
_transfer	Internal	Can Modify State	-
_mint	Internal	Can Modify State	-
_burn	Internal	Can Modify State	-
_approve	Internal	Can Modify State	-
_requireValidRecipient	Internal	-	-
_requireCallerIsBorrowerOperations	Internal	-	-
_requireCallerIsBOorVaultMorSP	Internal	-	-
_requireCallerIsStabilityPool	Internal	-	-
_requireCallerIsVaultMorSP	Internal	-	-
name	External	-	-

BAIToken			
symbol	External	-	-
decimals	External	-	-
version	External	-	-
permitTypeHash	External	-	-

4.3 Vulnerability Summary

[N1] [Suggestion] Gas Optimization

Category: Others

Content

Using assert will consume the remaining gas when the transaction fails to execute.

- Code location: contracts/BAIToken.sol #L237,L238,L246,L254,L262,L263
- Code location: contracts/ATID/CommunityIssuance.sol #L93,L196
- Code location: contracts/v0/BorrowerOperations.sol #L108,L177,L296
- Code location: contracts/v0/StabilityPool.sol #L498,L523,L563
- Code location: contracts/v0/VaultManager.sol #L392,L942,L1182,L1230,L1293,L1299,L1353,L1428
- Code location: contracts/v1/VaultManagerV1.sol #L388,L940,L1180,L1228,L1291,L1297,L1351,L1426

Solution

It is recommended to use require instead of assert to optimize gas.

Status

Confirming

[N2] [Suggestion] Token compatibility reminder

Category: Others

Content

The contract uses the incoming amount to directly participate in bookkeeping, so the contract is not compatible with deflationary tokens and inflationary tokens.

The transfer of the contract code involving the token uses transfer, and uses require to check the return value, and does not use safeTransfer, so it is not compatible with the transfer token (non-standard ERC20) that has no return value.

Solution

It is recommended to check the compatibility between the token contract and the AstridDao project when docking the token contract, avoid interacting with deflationary or inflationary tokens, and avoid docking tokens with no return value from the transfer function.

Status

Confirming

[N3] [Suggestion] Low-level call reminder

Category: Others

Content

Low-level calls are used in the WASTR contract, and there is no limit to the amount of gas used to transfer tokens to pools or users.

Code location: contracts/WrappedTokens/WASTR.sol #L200,L230,L317,L360,L390

Solution

When using low-level calls, it is recommended to limit the amount of gas used.

Status

Confirming

[N4] [Medium] Risk of excessive authority

Category: Authority Control Vulnerability

Content

Owner can set the following parameters, these addresses will affect the logic of external calls. Contracts that are not security audited may have security issues that affect users' assets.

- Code location: contracts/ATID/ATIDStaking.sol#L86-L125

```
function setAddresses
    (
        address _atidTokenAddress,
        address _baiTokenAddress,
        address _colTokenAddress,
        address _vaultManagerAddress,
        address _borrowerOperationsAddress,
        address _activePoolAddress,
        address _govTokenAddress
    )
        external
        onlyOwner
        override
    {
        checkContract(_atidTokenAddress);
        checkContract(_baiTokenAddress);
        checkContract(_colTokenAddress);
        checkContract(_vaultManagerAddress);
        checkContract(_borrowerOperationsAddress);
        checkContract(_activePoolAddress);
        checkContract(_govTokenAddress);

        atidToken = IATIDToken(_atidTokenAddress);
        baiToken = IBAIToken(_baiTokenAddress);
        colToken = IERC20(_colTokenAddress);
        vaultManagerAddress = _vaultManagerAddress;
        borrowerOperationsAddress = _borrowerOperationsAddress;
        activePoolAddress = _activePoolAddress;
        govToken = IGovToken(_govTokenAddress);

        emit ATIDTokenAddressSet(_atidTokenAddress);
        emit BAITokenAddressSet(_baiTokenAddress);
        emit COLTokenAddressSet(_colTokenAddress);
```

```
        emit VaultManagerAddressSet(_vaultManagerAddress);
        emit BorrowerOperationsAddressSet(_borrowerOperationsAddress);
        emit ActivePoolAddressSet(_activePoolAddress);
        emit GovTokenAddressSet(_govTokenAddress);

        // _renounceOwnership();
    }
```

- Code location: contracts/Dependencies/AstridBase.sol#L101-L121

```
    function setAddresses(
        address _activePool,
        address _defaultPool,
        address _priceFeed
    ) public onlyOwner {
        checkContract(_activePool);
        checkContract(_defaultPool);
        checkContract(_priceFeed);
        activePool = IActivePool(_activePool);
        defaultPool = IDefaultPool(_defaultPool);
        priceFeed = IPriceFeed(_priceFeed);
    }
```

- Code location: contracts/v0/ActivePool.sol#L37-L66

```
    function setAddresses(
        address _borrowerOperationsAddress,
        address _vaultManagerAddress,
        address _stabilityPoolAddress,
        address _defaultPoolAddress,
        address _collateralTokenAddress
    )
        external
        onlyOwner
    {
        checkContract(_borrowerOperationsAddress);
        checkContract(_vaultManagerAddress);
        checkContract(_stabilityPoolAddress);
        checkContract(_defaultPoolAddress);
        checkContract(_collateralTokenAddress);
```

```
        borrowerOperationsAddress = _borrowerOperationsAddress;
        vaultManagerAddress = _vaultManagerAddress;
        stabilityPoolAddress = _stabilityPoolAddress;
        defaultPoolAddress = _defaultPoolAddress;
        COLToken = IERC20(_collateralTokenAddress);

        emit BorrowerOperationsAddressChanged(_borrowerOperationsAddress);
        emit VaultManagerAddressChanged(_vaultManagerAddress);
        emit StabilityPoolAddressChanged(_stabilityPoolAddress);
        emit DefaultPoolAddressChanged(_defaultPoolAddress);
        emit COLTokenAddressChanged(_collateralTokenAddress);

        // _renounceOwnership();
    }
```

- Code location: contracts/v0/BorrowerOperations.sol#L91-L149

```
function setAddresses(
        address _vaultManagerAddress,
        address _activePoolAddress,
        address _defaultPoolAddress,
        address _stabilityPoolAddress,
        address _gasPoolAddress,
        address _collSurplusPoolAddress,
        address _priceFeedAddress,
        address _sortedVaultsAddress,
        address _colTokenAddress,
        address _baiTokenAddress,
        address _atidStakingAddress
    )
        external
        override
        onlyOwner
    {
        // This makes impossible to open a vault with zero withdrawn BAI
        assert(MIN_NET_DEBT > 0);

        checkContract(_vaultManagerAddress);
        checkContract(_activePoolAddress);
        checkContract(_defaultPoolAddress);
        checkContract(_stabilityPoolAddress);
        checkContract(_gasPoolAddress);
```

```
        checkContract(_collSurplusPoolAddress);
        checkContract(_priceFeedAddress);
        checkContract(_sortedVaultsAddress);
        checkContract(_colTokenAddress);
        checkContract(_baiTokenAddress);
        checkContract(_atidStakingAddress);

        vaultManager = IVaultManager(_vaultManagerAddress);
        activePool = IActivePool(_activePoolAddress);
        defaultPool = IDefaultPool(_defaultPoolAddress);
        stabilityPoolAddress = _stabilityPoolAddress;
        gasPoolAddress = _gasPoolAddress;
        collSurplusPool = ICollSurplusPool(_collSurplusPoolAddress);
        priceFeed = IPriceFeed(_priceFeedAddress);
        sortedVaults = ISortedVaults(_sortedVaultsAddress);
        COLToken = IERC20(_colTokenAddress);
        baiToken = IBAIToken(_baiTokenAddress);
        atidStakingAddress = _atidStakingAddress;
        atidStaking = IATIDStaking(_atidStakingAddress);

        emit VaultManagerAddressChanged(_vaultManagerAddress);
        emit ActivePoolAddressChanged(_activePoolAddress);
        emit DefaultPoolAddressChanged(_defaultPoolAddress);
        emit StabilityPoolAddressChanged(_stabilityPoolAddress);
        emit GasPoolAddressChanged(_gasPoolAddress);
        emit CollSurplusPoolAddressChanged(_collSurplusPoolAddress);
        emit PriceFeedAddressChanged(_priceFeedAddress);
        emit SortedVaultsAddressChanged(_sortedVaultsAddress);
        emit COLTokenAddressChanged(_colTokenAddress);
        emit BAITokenAddressChanged(_baiTokenAddress);
        emit ATIDStakingAddressChanged(_atidStakingAddress);

        // _renounceOwnership();
    }
```

- Code location: contracts/v0/CollSurplusPool.sol#L32-L57

```
function setAddresses(
        address _borrowerOperationsAddress,
        address _vaultManagerAddress,
        address _activePoolAddress,
        address _collateralTokenAddress
```

```
    )
        external
        override
        onlyOwner
    {
        checkContract(_borrowerOperationsAddress);
        checkContract(_vaultManagerAddress);
        checkContract(_activePoolAddress);

        borrowerOperationsAddress = _borrowerOperationsAddress;
        vaultManagerAddress = _vaultManagerAddress;
        activePoolAddress = _activePoolAddress;
        COLToken = IERC20(_collateralTokenAddress);

        emit BorrowerOperationsAddressChanged(_borrowerOperationsAddress);
        emit VaultManagerAddressChanged(_vaultManagerAddress);
        emit ActivePoolAddressChanged(_activePoolAddress);
        emit COLTokenAddressChanged(_collateralTokenAddress);

        // _renounceOwnership();
    }
```

- Code location: contracts/v0/DefaultPool.sol#L36-L57

```
function setAddresses(
        address _vaultManagerAddress,
        address _activePoolAddress,
        address _collateralTokenAddress
    )
        external
        onlyOwner
    {
        checkContract(_vaultManagerAddress);
        checkContract(_activePoolAddress);
        checkContract(_collateralTokenAddress);

        vaultManagerAddress = _vaultManagerAddress;
        activePoolAddress = _activePoolAddress;
        COLToken = IERC20(_collateralTokenAddress);

        emit VaultManagerAddressChanged(_vaultManagerAddress);
        emit ActivePoolAddressChanged(_activePoolAddress);
```

```
        emit COLTokenAddressChanged(_collateralTokenAddress);

        // _renounceOwnership();
    }
```

- Code location: contracts/v0/HintHelpers.sol#L26-L43

```
    function setAddresses(
        address _sortedVaultsAddress,
        address _vaultManagerAddress
    )
        external
        onlyOwner
    {
        checkContract(_sortedVaultsAddress);
        checkContract(_vaultManagerAddress);

        sortedVaults = ISortedVaults(_sortedVaultsAddress);
        vaultManager = IVaultManager(_vaultManagerAddress);

        emit SortedVaultsAddressChanged(_sortedVaultsAddress);
        emit VaultManagerAddressChanged(_vaultManagerAddress);

        // _renounceOwnership();
    }
```

- Code location: contracts/v0/StabilityPool.sol#L239-L281

```
function setAddresses(
        address _borrowerOperationsAddress,
        address _vaultManagerAddress,
        address _activePoolAddress,
        address _colTokenAddress,
        address _baiTokenAddress,
        address _sortedVaultsAddress,
        address _priceFeedAddress,
        address _communityIssuanceAddress
    )
        external
        override
        onlyOwner
```

```
{
    checkContract(_borrowerOperationsAddress);
    checkContract(_vaultManagerAddress);
    checkContract(_activePoolAddress);
    checkContract(_colTokenAddress);
    checkContract(_baiTokenAddress);
    checkContract(_sortedVaultsAddress);
    checkContract(_priceFeedAddress);
    checkContract(_communityIssuanceAddress);

    borrowerOperations = IBorrowerOperations(_borrowerOperationsAddress);
    vaultManager = IVaultManager(_vaultManagerAddress);
    activePool = IActivePool(_activePoolAddress);
    COLToken = IERC20(_colTokenAddress);
    baiToken = IBAIToken(_baiTokenAddress);
    sortedVaults = ISortedVaults(_sortedVaultsAddress);
    priceFeed = IPriceFeed(_priceFeedAddress);
    communityIssuance = ICommunityIssuance(_communityIssuanceAddress);

    emit BorrowerOperationsAddressChanged(_borrowerOperationsAddress);
    emit VaultManagerAddressChanged(_vaultManagerAddress);
    emit ActivePoolAddressChanged(_activePoolAddress);
    emit COLTokenAddressChanged(_colTokenAddress);
    emit BAITokenAddressChanged(_baiTokenAddress);
    emit SortedVaultsAddressChanged(_sortedVaultsAddress);
    emit PriceFeedAddressChanged(_priceFeedAddress);
    emit CommunityIssuanceAddressChanged(_communityIssuanceAddress);

    // _renounceOwnership();
}
```

- Code location: contracts/v0/VaultManager.sol#L214-L268

```
function setAddresses(
    address _borrowerOperationsAddress,
    address _activePoolAddress,
    address _defaultPoolAddress,
    address _stabilityPoolAddress,
    address _gasPoolAddress,
    address _collSurplusPoolAddress,
    address _priceFeedAddress,
```

```
        address _baiTokenAddress,
        address _sortedVaultsAddress,
        address _atidTokenAddress,
        address _atidStakingAddress
    )
        external
        override
        onlyOwner
    {
        checkContract(_borrowerOperationsAddress);
        checkContract(_activePoolAddress);
        checkContract(_defaultPoolAddress);
        checkContract(_stabilityPoolAddress);
        checkContract(_gasPoolAddress);
        checkContract(_collSurplusPoolAddress);
        checkContract(_priceFeedAddress);
        checkContract(_baiTokenAddress);
        checkContract(_sortedVaultsAddress);
        checkContract(_atidTokenAddress);
        checkContract(_atidStakingAddress);

        borrowerOperationsAddress = _borrowerOperationsAddress;
        activePool = IActivePool(_activePoolAddress);
        defaultPool = IDefaultPool(_defaultPoolAddress);
        stabilityPool = IStabilityPool(_stabilityPoolAddress);
        gasPoolAddress = _gasPoolAddress;
        collSurplusPool = ICollSurplusPool(_collSurplusPoolAddress);
        priceFeed = IPriceFeed(_priceFeedAddress);
        baiToken = IBAIToken(_baiTokenAddress);
        sortedVaults = ISortedVaults(_sortedVaultsAddress);
        atidToken = IATIDToken(_atidTokenAddress);
        atidStaking = IATIDStaking(_atidStakingAddress);

        emit BorrowerOperationsAddressChanged(_borrowerOperationsAddress);
        emit ActivePoolAddressChanged(_activePoolAddress);
        emit DefaultPoolAddressChanged(_defaultPoolAddress);
        emit StabilityPoolAddressChanged(_stabilityPoolAddress);
        emit GasPoolAddressChanged(_gasPoolAddress);
        emit CollSurplusPoolAddressChanged(_collSurplusPoolAddress);
        emit PriceFeedAddressChanged(_priceFeedAddress);
        emit BAITokenAddressChanged(_baiTokenAddress);
        emit SortedVaultsAddressChanged(_sortedVaultsAddress);
        emit ATIDTokenAddressChanged(_atidTokenAddress);
        emit ATIDStakingAddressChanged(_atidStakingAddress);
```

```
        // _renounceOwnership();
    }
```

Owner can set atidStakingAddresses. atidStakingAddresses can transfer users' assets arbitrarily.

- Code location: contracts/ATID/ATIDToken.sol#L151-L159

```
    function setATIDStaking(
        address _atidStakingAddress,
        uint _active
    ) external onlyOwner {
        checkContract(_atidStakingAddress);
        atidStakingAddresses[_atidStakingAddress] = _active;
        emit ATIDStakingSet(_atidStakingAddress, _active);
    }
```

- Code location: contracts/ATID/ATIDToken.sol#L255-L258

```
    function sendToATIDStaking(address _sender, uint256 _amount) external override {
        _requireCallerIsATIDStaking();
        _transfer(_sender, msg.sender, _amount);
    }
```

- Code location: contracts/ATID/CommunityIssuance.sol#L200-L204

```
    function sendATID(address _account, uint _ATIDamount) external override {
        _requireCallerIsStabilityPool();

        require(atidToken.transfer(_account, _ATIDamount), "CommunityIssuance: cannot
receive ATID reward");
    }
```

- Code location: contracts/v0/CollSurplusPool.sol#L32-L57

```
    function setAddresses(
```

```
        address _borrowerOperationsAddress,
        address _vaultManagerAddress,
        address _activePoolAddress,
        address _collateralTokenAddress
    )
        external
        override
        onlyOwner
    {
        checkContract(_borrowerOperationsAddress);
        checkContract(_vaultManagerAddress);
        checkContract(_activePoolAddress);

        borrowerOperationsAddress = _borrowerOperationsAddress;
        vaultManagerAddress = _vaultManagerAddress;
        activePoolAddress = _activePoolAddress;
        COLToken = IERC20(_collateralTokenAddress);

        emit BorrowerOperationsAddressChanged(_borrowerOperationsAddress);
        emit VaultManagerAddressChanged(_vaultManagerAddress);
        emit ActivePoolAddressChanged(_activePoolAddress);
        emit COLTokenAddressChanged(_collateralTokenAddress);

        // _renounceOwnership();
    }
```

- Code location: contracts/Dependencies/AstridBase.sol#L114-L135

```
    function setParams(
        uint _MCR,
        uint _CCR,
        uint _BAIGasCompensation,
        uint _minNetDebt,
        uint _percentageDivisor,
        uint _borrowingFeeFloor
    ) public onlyOwner {
        require(_MCR > _100pct, "MCR cannot < 100%");
        require(_CCR > _100pct, "CCR cannot < 100%");
        require(_BAIGasCompensation > 0, "Gas compensation must > 0");
        require(_minNetDebt > 0, "Min net debt must > 0");
        require(_percentageDivisor > 0, "Percentage divisor must > 0");
        // Borrowing fee floor can be 0 if necessary.
```

```
        MCR = _MCR;
        CCR = _CCR;
        BAI_GAS_COMPENSATION = _BAIGasCompensation;
        MIN_NET_DEBT = _minNetDebt;
        PERCENT_DIVISOR = _percentageDivisor;
        BORROWING_FEE_FLOOR = _borrowingFeeFloor;
    }
```

- Code location: contracts/v0/SortedVaults.sol#L75-L89

```
    function setParams(uint256 _size, address _vaultManagerAddress, address
_borrowerOperationsAddress) external override onlyOwner {
        require(_size > 0, "SortedVaults: Size can't be zero");
        checkContract(_vaultManagerAddress);
        checkContract(_borrowerOperationsAddress);

        data.maxSize = _size;

        vaultManager = IVaultManager(_vaultManagerAddress);
        borrowerOperationsAddress = _borrowerOperationsAddress;

        emit VaultManagerAddressChanged(_vaultManagerAddress);
        emit BorrowerOperationsAddressChanged(_borrowerOperationsAddress);

        // _renounceOwnership();
    }
```

Solution

It is recommended to add an initialized modifier, which can only be set if it is not initialized, And the function of changing contract parameters should be managed by governance contracts.

Status

Confirming

[N5] [Suggestion] Redundant comment code

Category: Others

Content

There is a lot of commented code in the contract code.

- Code location: contracts/ATID/ATIDStaking.sol#L124
- Code location: contracts/ATID/ATIDStaking.sol#L380-L383
- Code location: contracts/ATID/CommunityIssuance.sol#L95
- Code location: contracts/v0/ActivePool.sol#L65
- Code location: contracts/v0/BorrowerOperations.sol#L148
- Code location: contracts/v0/CollSurplusPool.sol#L56
- Code location: contracts/v0/DefaultPool.sol#L56
- Code location: contracts/v0/HintHelpers.sol#L42
- Code location: contracts/v0/SortedVaults.sol#L88
- Code location: contracts/v0/StabilityPool.sol#L280
- Code location: contracts/v0/VaultManager.sol#L267
- Code location: contracts/v0/ActivePool.sol#L176-L178
- Code location: contracts/v0/BorrowerOperations.sol#L394-L408
- Code location: contracts/v0/CollSurplusPool.sol#L92
- Code location: contracts/v0/CollSurplusPool.sol#L127-L128
- Code location: contracts/v0/DefaultPool.sol#L122-L124
- Code location: contracts/v0/StabilityPool.sol#L842-L844
- Code location:

  contracts/WrappedTokens/WASTR.sol#L82,L90,L98,L109,L144,L153,L162,L183,L199,L303,L336,L346
- Code location: contracts/Dependencies/Ownable.sol#L51-L108
- Code location: contracts/BAIToken.sol#L74-L85

Solution

It is recommended to clarify business logic and remove redundant commented code.

Status

Confirming

[N6] [High] Excessive authority issues of owner

Category: Authority Control Vulnerability

Content

Owner can mint and burn any address's token arbitrarily.

- Code location: contracts/ATID/GovToken.sol#L150-L153

```
    function mint(address account, uint256 amount) public virtual override onlyOwner
{
        _mint(account, amount);
    }
    function burn(address account, uint256 amount) public virtual override onlyOwner
{
        _burn(account, amount);
    }
```

Owner can set multiple new Owners.

- Code location: contracts/Dependencies/Ownable.sol#L29-L47

```
    modifier onlyOwner() {
        require(isOwner(), "Ownable: caller is not the owner");
        _;
    }

    /**
     * @dev Returns true if the caller is a current owner.
     */
    function isOwner() public view returns (bool) {
        return owners[msg.sender] != 0;
    }
```

```
/**
 * @dev Sets whether an address is an owner.
 */
function setOwnership(address _address, uint _isOwned) public onlyOwner {
    owners[_address] = _isOwned;
    emit OwnershipChanged(_address, _isOwned);
}
```

Owner can modify the Oracle contract.

- Code location: contracts/v0/PriceFeed.sol#L57-L60

```
function setOracleAddress(address _newDiaOracleV2Address) public onlyOwner {
    checkContract(_newDiaOracleV2Address);
    diaOracle = IDIAOracleV2(_newDiaOracleV2Address);
}
```

Solution

It is recommended to separate permissions, only ATIDStaking and Governance contracts can call mint and burn functions.And the Owner role should be managed using timelock, governance contract or multi-sign contract.

Status

Confirming

[N7] [Suggestion] Redundant code

Category: Others

Content

The contract does not receive native coins by default, so there is no need to declare the receive function.

- Code location: contracts/v0/ActivePool.sol#L174-L179

```
receive() external payable {
    revert("ActivePool: should not pay to this contract.");
    // _requireCallerIsBorrowerOperationsOrDefaultPool();
    // ETH = ETH.add(msg.value);
```

```
        // emit ActivePoolETHBalanceUpdated(ETH);
    }
```

- Code location: contracts/v0/CollSurplusPool.sol#L125-L129

```
    receive() external payable {
        revert("CollSurplusPool: should not pay to this contract.");
        // _requireCallerIsActivePool();
        // ETH = ETH.add(msg.value);
    }
```

- Code location: contracts/v0/StabilityPool.sol#L840

```
    receive() external payable {
        revert("StabilityPool: should not pay to this contract.");
        // _requireCallerIsActivePool();
        // ETH = ETH.add(msg.value);
        // StabilityPoolETHBalanceUpdated(ETH);
    }
```

Solution

It is recommended to delete redundant code to optimize the size of the contract and optimize the Gas.

Status

Confirming

[N8] [Low] Coding Standards Issues

Category: Others

Content

After executing `COLToken.transfer`, execute `IActivePool(activePool).receiveCOL(_amount);` and then execute `require(success, "DefaultPool: sending collateral failed");` Does not conform to coding standards.

- Code location: contracts/v0/DefaultPool.sol#L84

```
function sendCOLToActivePool(uint _amount) external override nonReentrant {
    _requireCallerIsVaultManager();
    address activePool = activePoolAddress; // cache to save an SLOAD
    COL = COL.sub(_amount);
    emit DefaultPoolCOLBalanceUpdated(COL);
    emit COLSent(activePool, _amount);

    bool success = COLToken.transfer(activePool, _amount);
    IActivePool(activePool).receiveCOL(_amount);
    require(success, "DefaultPool: sending collateral failed");
}
```

Solution

It is recommended to execute `COLToken.transfer` and execute `require(success, "DefaultPool: sending collateral failed");` and then execute `IActivePool(activePool).receiveCOL(_amount);` .

Status

Confirming

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002206090002	SlowMist Security Team	2022.05.16 - 2022.06.09	High Risk

Summary conclusion: The SlowMist security team uses a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 1 high risk, 1 medium risk, 1 low risk, 5 suggestion vulnerabilities. All the findings were fixed. The code was not deployed to the mainnet.

6 Statement

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

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

Official Website

www.slowmist.com

E-mail

team@slowmist.com

Twitter

@SlowMist_Team

Github

https://github.com/slowmist