

Smart Contract
Security Audit Report

Table Of Contents

1 Executive Summary

2 Audit Methodology

3 Project Overview

3.1 Project Introduction

3.2 Vulnerability Information

4 Code Overview

4.1 Contracts Description

4.2 Visibility Description

4.3 Vulnerability Summary

5 Audit Result

6 Statement

1 Executive Summary

On 2023.04.26, the SlowMist security team received the Jasper Vault team's security audit application for Jasper Vault, 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.
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
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

Serial Number	Audit Class	Audit Subclass
7	Security Design 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

Jasper Vault (jVault) is a Vault-to-Vault (V2V) DeFi protocol incorporating Account Abstraction (AA). jVault features both a permissionless V2V marketplace, and a permissioned V2V asset management platform.

The permissionless V2V marketplace allows DeFi traders to monetise their DeFi recipes, crypto indices, and smart money strategies. DeFi traders can also leverage the Smart Contract Accounts feature of ERC-4337 in jVault to automate their DeFi recipes and trading procedures through bundled transactions, aggregate signatures required for pre-approved transactions, and sponsor gas fees through paymasters.

Through jVault's smart contracts, DeFi traders can charge royalty fees for users to mirror their portfolio or set up subscription-based memberships to gain access to their strategies. DeFi investors can easily mirror these

trading recipes and holdings of the DeFi traders through their AA smart contract mirror vaults, without having to transfer their crypto assets to these traders.

For sophisticated institutional investors which has stricter regulatory requirements, jVault offers a permissioned V2V asset management platform which includes all existing jVault features while prioritising security, compliance, and regulations. Asset managers can seamlessly incorporate jVault into their current institutional MPC custody solution, allowing them to reap the advantages of AA without sacrificing control over their assets.

Rather than moving assets to other funds or traders, asset managers now move the DeFi strategy signals into their jVault, by simply following the DeFi strategist through the Master Vault. Likewise, instead of the conventional redemption of funds, asset managers can now easily unfollow the DeFi strategy by clicking the "unfollow" button.

Through jVault (Permissioned), DeFi strategists now have the ability to simultaneously share their trading signals on any EVM chains with asset managers, and charge pre-defined performance fees through smart contracts at regular intervals.

3.2 Vulnerability Information

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

The permissionless part of the Jasper Vault protocol:

NO	Title	Category	Level	Status
N1	Missing check for path	Design Logic Audit	Low	Fixed
N2	The balance check is incorrect	Design Logic Audit	Suggestion	Fixed
N3	Missing check for ETH in the exchange path	Design Logic Audit	High	Fixed
N4	Error of the return value	Design Logic Audit	Critical	Fixed
N5	Sandwich Attack Risk	Reordering Vulnerability	Medium	Fixed
N6	Missing zero address	Others	Suggestion	Fixed

NO	Title	Category	Level	Status
	check			
N7	call() should be used instead of transfer()	Gas Optimization Audit	Suggestion	Fixed
N8	Redundant code	Others	Low	Fixed
N9	Unclear function logic	Others	Low	Fixed
N10	Missing return value check	Others	Low	Fixed
N11	Lack of event records	Others	Suggestion	Fixed
N12	ERC20 return values not checked	Others	Suggestion	Fixed

The permissioned part of the Jasper Vault protocol:

NO	Title	Category	Level	Status
N13	Missing validation of follow trading	Authority Control Vulnerability Audit	High	Acknowledged
N14	Missing validation of follow the leverage	Authority Control Vulnerability Audit	High	Acknowledged
N15	The DoS issue	Denial of Service Vulnerability	Medium	Fixed
N16	Possible overflow risk	Integer Overflow and Underflow Vulnerability	Suggestion	Fixed
N17	Risk of excessive authority	Authority Control Vulnerability Audit	Medium	Fixed
N18	Lack of scope restrictions	Others	Medium	Fixed
N19	Wrong variable obtained	Scoping and Declarations Audit	Low	Fixed
N20	Lack of authority control	Authority Control Vulnerability Audit	Low	Fixed
N21	Missing variable updates when unsubscribing	Design Logic Audit	High	Fixed

NO	Title	Category	Level	Status
N22	Lack of judgement on EOA address	Design Logic Audit	Medium	Fixed
N23	Lack of judgment on the upper limit of costs	Design Logic Audit	High	Fixed
N24	Lack of Token Whitelist Check and zero value check	Design Logic Audit	Medium	Fixed
N25	The incorrect variable is taken in the judgment	Design Logic Audit	Medium	Fixed
N26	Clear followers list element issue	Gas Optimization Audit	Low	Fixed
N27	Incorrect calculation of profit	Design Logic Audit	Medium	Acknowledged

4 Code Overview

4.1 Contracts Description

Audit scope

- contract/ ProtocolViewer
- contract/ UniswapV2ExchangeAdapter
- contract/ Controller
- contract/ SetTokenCreator
- contract/ IntegrationRegistry
- contract/ SetValuer
- contract/ UniswapPairPriceAdapter
- contract/ PriceOracleProxy
- contract/ IdentityService
- contract/ NavIssuanceModule

- contract/ AaveLeverageModule
- contract/ DebtIssuanceModule
- contract/ DebtIssuanceModuleV2
- contract/ TradeModule
- contract/ StreamingFeeModule
- contract/ SignalSuscriptionModule
- contract/ WrapModuleV2
- contract/ AaveV2WrapV2Adapter
- contract/ ManagerCore
- contract/ DelegatedManagerFactory
- contract/ IssuanceExtension
- contract/ StreamingFeeSplitExtension
- contract/ SignalSuscriptionExtension
- contract/ WrapExtension
- contract/ LeverageExtension
- contract/ CopyTradingExtension
- contract/ NAVIssuanceExtension
- contract/ AaveV2WrapV2Adapter
- contract/ VaultFactory
- contract/ VaultPaymaster
- contract/ CompoundWrapV2Adapter
- contract/ StakingModule
- contract/ StakingExtension

Audit version:

https://github.com/jaspervault/jasper-vaults-contracts

commit: 03ebf79017f76fba3651d6eef975a440e7a8dc5b

Fixed version:

https://github.com/jaspervault/jasper-vaults-contracts

commit: 7ed6bb1d7ebfd27b979e9829a73c7d1cf32b32d4

Only the fix code for this submission will be audited, other new features are not included in this audit.

The main network address of the contract is as follows:

Contract Name	Contract Address	Chain	Open Source Or Not
AaveV2WrapV2Adapter	0xCea2532dd086a9B0b1d18C6293754A7A494A767c	Polygon	☑
UniswapV2ExchangeAdapter	0x908a391E2c3200f564f3a5b1dDF42DFA95754B37	Polygon	☑
Controller	0x6FA215F8347B409fa7537590a49b1065f0ED6b42	Polygon	☑
UniswapPairPriceAdapter	0xb482fa55f9Ec2F056fC0535244F5627C26D16b5C	Polygon	☑
SetTokenCreator	0x9fbdE027a17D83Fbd9AE81693f50b1Cc8A3064B5	Polygon	☑
IntegrationRegistry	0xF48bba9d9D86Fd338F98D96D3E5AF53f89ffDCcC	Polygon	☑
SetValuer	0x508fCbE3c18234D4Af2F81f1bC9C65e96FDbFa79	Polygon	☑
PriceOracle	0xf049290BbEb454F50466C948287c49f18A619781	Polygon	☑
IdentityService	0xDA5c408D50C0Ad06e50067Fea42383CCc3B66F26	Polygon	☑
ManagerCore	0x1e7a9035f01E21B5a51e6f021D61e2A87eD4E0a4	Polygon	☑
DelegatedManagerFactory	0x44FBcD2CD2A603BACDAeD27220d7Ea5299E502F6	Polygon	☑
SubscribeFeePool	0xC1415a77eBad59b6910c637BaCEAcD8f088d4030	Polygon	☑
SignalSuscriptionModule	0xe46037e8366584d10FB8c64dda19ca9d8Ca13F56	Polygon	☑

Contract Name	Contract Address	Chain	Open Source Or Not
NavIssuanceModule	0x95d192270b78689Fd33E0f7d9c939EE27d06CF9D	Polygon	☑
StakingModule	0x08e1488864D52844912cdb8C07659f407311c89c	Polygon	☑
DebtIssuanceModule	0x2ea590ff1f237E5450409C39A476e11779CAE571	Polygon	☑
DebtIssuanceModuleV2	0x43F6BFbE9DAd44cF0A60570C30c307D949bE4Cd4	Polygon	☑
TradeModule	0xD524bA785FA50C1c2998bf45DbcCCAa801304608	Polygon	☑
StreamingFeeModule	0x78D7EEFE2787b7a8FfFA10Dd930617720FD01006	Polygon	☑
WrapModuleV2	0x7e9feAcE89C0a7C3cacF44594297Bd97D73D3583	Polygon	☑
AaveLeverageModule	0x6Ee4c5E36e3F2D452F4D5e606371D84E3fC0d684	Polygon	☑
IssuanceExtension	0x16b76917d27BC7e88d557B72E19Cc4Dc2B3c9A4a	Polygon	☑
StreamingFeeSplitExtension	0x70Ff719DFe1c9188468290e6c3D563A03df31300	Polygon	☑
SignalSuscriptionExtension	0x3C23216A0cCC11236D4BF0944FbcfA7150Bd370E	Polygon	☑
LeverageExtension	0x331CB625441ebBeC5fd756C7CB41d4641332Fe16	Polygon	☑
WrapExtension	0x872eCa74014632B9e0108Eff52F19ca5AE4AFBA7	Polygon	☑
CopyTradingExtension	0x647280fA6FA73159acdE7cF1494c9c013d50CA35	Polygon	☑
NAVIssuanceExtension	0xf925f6eF8420FA34f25142eA483a65806463304d	Polygon	☑
VaultPaymaster	0xA9946af68E66a3A3ecF2263B0087FC060323d2E4	Polygon	☑
VaultFactory	0xFe0c23802C7929059040e1D91E74206da73f4CB7	Polygon	☑

4.2 Visibility Description

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

VaultPaymaster			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
_authorizeUpgrade	Internal	Can Modify State	onlyOwner
setSetting	External	Can Modify State	onlyOwner
setToekenToPath	External	Can Modify State	onlyOwner
deposit	Internal	Can Modify State	-
depositBalance	External	Payable	-
withdrawTo	Internal	Can Modify State	-
depositEth	Internal	Can Modify State	-
depositToken	Internal	Can Modify State	-
withdrawEth	External	Can Modify State	-
withdrawToken	External	Can Modify State	-
get_path	External	-	-
_swapExactTokensForETH	Internal	Can Modify State	-
_swapExactETHForTokens	Internal	Can Modify State	-
unlockStake	External	Can Modify State	onlyOwner
getDeposit	Public	-	-
withdrawStake	External	Can Modify State	onlyOwner
addStake	External	Payable	onlyOwner

VaultPaymaster			
validatePaymasterUserOp	External	Can Modify State	-
_requireFromEntryPoint	Internal	Can Modify State	-
_validatePaymasterUserOp	Internal	Can Modify State	-
postOp	External	Can Modify State	-
_postOp	Internal	Can Modify State	-
settleFollowers	Internal	Can Modify State	-
<Fallback>	External	Payable	-
<Receive Ether>	External	Payable	-

VaultFactory			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
setSetting	External	Can Modify State	onlyOwner
_authorizeUpgrade	Internal	Can Modify State	onlyOwner
addStake	External	Payable	onlyOwner
getAccountList	External	-	-
createAccount	Public	Can Modify State	-
getAddress	Public	-	-

SignalSuscriptionExtension			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	BaseGlobalExtension

SignalSuscriptionExtension			
setWhiteList	External	Can Modify State	onlyOperator
initializeModule	External	Can Modify State	onlyOwnerAndValidManager
initializeExtension	External	Can Modify State	onlyOwnerAndValidManager
initializeModuleAndExtension	External	Can Modify State	onlyOwnerAndValidManager
removeExtension	External	Can Modify State	-
editSubscribeFee	External	Can Modify State	onlySettle onlyOperator
subscribe	External	Can Modify State	onlySettle ValidWhitelist onlyOperator
update_allowedCopytrading	External	Can Modify State	onlyOperator
unsubscribe	External	Can Modify State	onlySubscribed ValidWhitelist onlyOperator
exectueFollowEnd	External	Can Modify State	-
getFollowers	External	-	-
getExectueFollow	External	-	-
warnLine	External	-	-
unsubscribeLine	External	-	-
_initializeModule	Internal	Can Modify State	-

WrapExtension			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	BaseGlobalExtension
initializeModule	External	Can Modify State	onlyOwnerAndValidManager

WrapExtension			
initializeExtension	External	Can Modify State	onlyOwnerAndValidManager
initializeModuleAndExtension	External	Can Modify State	onlyOwnerAndValidManager
removeExtension	External	Can Modify State	-
wrap	External	Can Modify State	onlySettle onlyOperator onlyAllowedAsset ValidAdapter
wrapWithEther	External	Can Modify State	onlySettle onlyOperator ValidAdapter
unwrap	External	Can Modify State	onlySettle onlyOperator onlyAllowedAsset ValidAdapter
unwrapWithEther	External	Can Modify State	onlySettle onlyOperator onlyAllowedAsset ValidAdapter
wrapWithFollowers	External	Can Modify State	onlySettle onlyOperator onlyAllowedAsset ValidAdapter
_executeWrapWithFollowers	Internal	Can Modify State	-
wrapEtherWithFollowers	External	Can Modify State	onlySettle onlyOperator ValidAdapter
_executeWrapEtherWithFollowers	Internal	Can Modify State	-
unwrapWithFollowers	External	Can Modify State	onlySettle onlyOperator onlyAllowedAsset ValidAdapter
_executeUnwrapWithFollowers	Internal	Can Modify State	-
unwrapWithEtherWithFollowers	External	Can Modify State	onlySettle onlyOperator onlyAllowedAsset ValidAdapter
_executeUnwrapEtherWithFollowers	Internal	Can Modify State	-
_execute	Internal	Can Modify State	-
_initializeModule	Internal	Can Modify State	-

LeverageExtension			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	BaseGlobalExtension
initializeModule	External	Can Modify State	onlyOwnerAndValidManager
initializeExtension	External	Can Modify State	onlyOwnerAndValidManager
initializeModuleAndExtension	External	Can Modify State	onlyOwnerAndValidManager
removeExtension	External	Can Modify State	-
lever	External	Can Modify State	onlySettle onlyOperator ValidAdapter
delever	External	Can Modify State	onlySettle onlyOperator ValidAdapter
deleverToZeroBorrowBalance	External	Can Modify State	onlySettle onlyOperator ValidAdapter
leverFollowers	External	Can Modify State	onlySettle onlyOperator ValidAdapter
deleverFollowers	External	Can Modify State	onlySettle onlyOperator ValidAdapter
_executeFollower	Internal	Can Modify State	-
deleverToZeroBorrowBalanceFollowers	External	Can Modify State	onlySettle onlyOperator ValidAdapter
_executeDeleverToZeroFollower	Internal	Can Modify State	-
addCollateralAssets	External	Can Modify State	onlyOperator
removeCollateralAssets	External	Can Modify State	onlyOperator
addBorrowAssets	External	Can Modify State	onlyOperator
removeBorrowAssets	External	Can Modify State	onlyOperator

LeverageExtension			
_execute	Internal	Can Modify State	-
_initializeModule	Internal	Can Modify State	-

CopyTradingExtension			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	BaseGlobalExtension
addIntegrations	External	Can Modify State	onlyManagerCoreOwner
removeIntegrations	External	Can Modify State	onlyManagerCoreOwner
initializeModule	External	Can Modify State	onlyOwnerAndValidManager
initializeExtension	External	Can Modify State	onlyOwnerAndValidManager
initializeModuleAndExtension	External	Can Modify State	onlyOwnerAndValidManager
removeExtension	External	Can Modify State	-
batchTrade	External	Can Modify State	onlySettle onlyOperator
batchTradeWithFollowers	External	Can Modify State	onlySettle onlyOperator
getIntegrations	External	-	-
_addIntegration	Internal	Can Modify State	-
_executeTrade	Internal	Can Modify State	-
_initializeModule	Internal	Can Modify State	-

NAVIssuanceExtension			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	BaseGlobalExtension
distributeFees	Public	Can Modify State	-

NAVIssuanceExtension			
initializeModule	External	Can Modify State	onlyOwnerAndValidManager
initializeExtension	External	Can Modify State	onlyOwnerAndValidManager
initializeModuleAndExtension	External	Can Modify State	onlyOwnerAndValidManager
removeExtension	External	Can Modify State	-
addReserveAsset	External	Can Modify State	onlyOwner
removeReserveAsset	External	Can Modify State	onlyOwner
editPremium	External	Can Modify State	onlyOwner
editManagerFee	External	Can Modify State	onlyOwner
editFeeRecipient	External	Can Modify State	onlyOwner
_initializeModule	Internal	Can Modify State	-

StakingExtension			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	BaseGlobalExtension
stake	External	Can Modify State	onlySettle onlyOperator ValidAdapter
unstake	External	Can Modify State	onlySettle onlyOperator ValidAdapter
stakeWithFollowers	External	Can Modify State	onlySettle onlyOperator ValidAdapter
unstakeWithFollowers	External	Can Modify State	onlySettle onlyOperator ValidAdapter
_execute	Internal	Can Modify State	-
initializeModule	External	Can Modify State	onlyOwnerAndValidManager
initializeExtension	External	Can Modify State	onlyOwnerAndValidManager

StakingExtension			
initializeModuleAndExtension	External	Can Modify State	onlyOwnerAndValidManager
removeExtension	External	Can Modify State	-
_initializeModule	Internal	Can Modify State	-

SetTokenCreator			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
create	External	Can Modify State	-

IntegrationRegistry			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
addIntegration	Public	Can Modify State	onlyOwner
batchAddIntegration	External	Can Modify State	onlyOwner
editIntegration	Public	Can Modify State	onlyOwner
batchEditIntegration	External	Can Modify State	onlyOwner
removeIntegration	External	Can Modify State	onlyOwner
getIntegrationAdapter	External	-	-
getIntegrationAdapterWithHash	External	-	-
isValidIntegration	External	-	-
_nameHash	Internal	-	-

NavIssuanceModule			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ModuleBase
issue	External	Can Modify State	nonReentrant onlyValidAndInitializedSet
issueWithEther	External	Payable	nonReentrant onlyValidAndInitializedSet
redeem	External	Can Modify State	nonReentrant onlyValidAndInitializedSet
redeemIntoEther	External	Can Modify State	nonReentrant onlyValidAndInitializedSet
addReserveAsset	External	Can Modify State	onlyManagerAndValidSet
removeReserveAsset	External	Can Modify State	onlyManagerAndValidSet
editPremium	External	Can Modify State	onlyManagerAndValidSet
editManagerFee	External	Can Modify State	onlyManagerAndValidSet
editFeeRecipient	External	Can Modify State	onlyManagerAndValidSet
initialize	External	Can Modify State	onlySetManager onlyValidAndPendingSet
removeModule	External	Can Modify State	-
<Receive Ether>	External	Payable	-
getReserveAssets	External	-	-
getIssuePremium	External	-	-
getRedeemPremium	External	-	-
getManagerFee	External	-	-
getExpectedSetTokenIssueQuantity	External	-	-

NavIssuanceModule			
getExpectedReserveRedeemQuantity	External	-	-
isIssueValid	External	-	-
isRedeemValid	External	-	-
_validateCommon	Internal	-	-
_validateIssuanceInfo	Internal	-	-
_validateRedemptionInfo	Internal	-	-
_createIssuanceInfo	Internal	-	-
_createRedemptionInfo	Internal	-	-
_transferCollateralAndHandleFees	Internal	Can Modify State	-
_transferWETHAndHandleFees	Internal	Can Modify State	-
_handleIssueStateUpdates	Internal	Can Modify State	-
_handleRedeemStateUpdates	Internal	Can Modify State	-
_handleRedemptionFees	Internal	Can Modify State	-
_getIssuePremium	Internal	-	-
_getRedeemPremium	Internal	-	-
_getFees	Internal	-	-
_getProtocolAndManagerFeePercentages	Internal	-	-
_getSetTokenMintQuantity	Internal	-	-
_getRedeemReserveQuantity	Internal	-	-
_getIssuePositionMultiplier	Internal	-	-
_getRedeemPositionMultiplier	Internal	-	-

NavIssuanceModule			
_getIssuePositionUnit	Internal	-	-
_getRedeemPositionUnit	Internal	-	-
_callPreIssueHooks	Internal	Can Modify State	-
_callPreRedeemHooks	Internal	Can Modify State	-

AaveLeverageModule			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ModuleBase
lever	External	Can Modify State	nonReentrant onlyManagerAndValidSet
delever	External	Can Modify State	nonReentrant onlyManagerAndValidSet
deleverToZeroBorrowBalance	External	Can Modify State	nonReentrant onlyManagerAndValidSet
sync	Public	Can Modify State	nonReentrant onlyValidAndInitializedSet
initialize	External	Can Modify State	onlySetManager onlyValidAndPendingSet
removeModule	External	Can Modify State	onlyValidAndInitializedSet
registerToModule	External	Can Modify State	onlyManagerAndValidSet
addUnderlyingToReserveTokensMapping	External	Can Modify State	-
addCollateralAssets	External	Can Modify State	onlyManagerAndValidSet
removeCollateralAssets	External	Can Modify State	onlyManagerAndValidSet
addBorrowAssets	External	Can Modify State	onlyManagerAndValidSet

AaveLeverageModule			
removeBorrowAssets	External	Can Modify State	onlyManagerAndValidSet
updateAllowedSetToken	External	Can Modify State	onlyOwner
updateAnySetAllowed	External	Can Modify State	onlyOwner
moduleIssueHook	External	Can Modify State	onlyModule
moduleRedeemHook	External	Can Modify State	onlyModule
componentIssueHook	External	Can Modify State	onlyModule
componentRedeemHook	External	Can Modify State	onlyModule
getEnabledAssets	External	-	-
_deposit	Internal	Can Modify State	-
_withdraw	Internal	Can Modify State	-
_repayBorrow	Internal	Can Modify State	-
_repayBorrowForHook	Internal	Can Modify State	-
_borrow	Internal	Can Modify State	-
_borrowForHook	Internal	Can Modify State	-
_executeTrade	Internal	Can Modify State	-
_accrueProtocolFee	Internal	Can Modify State	-
_updateLeverPositions	Internal	Can Modify State	-
_updateDeleverPositions	Internal	Can Modify State	-

AaveLeverageModule			
_updateCollateralPosition	Internal	Can Modify State	-
_updateBorrowPosition	Internal	Can Modify State	-
_createAndValidateActionInfo	Internal	-	-
_createAndValidateActionInfoNotional	Internal	-	-
_addUnderlyingToReserveTokensMapping	Internal	Can Modify State	-
_addCollateralAssets	Internal	Can Modify State	-
_addBorrowAssets	Internal	Can Modify State	-
_updateUseReserveAsCollateral	Internal	Can Modify State	-
_validateCommon	Internal	-	-
_validateNewCollateralAsset	Internal	-	-
_validateNewBorrowAsset	Internal	-	-
_getCollateralPosition	Internal	-	-
_getBorrowPosition	Internal	-	-

DebtIssuanceModule			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ModuleBase
issue	External	Can Modify State	nonReentrant onlyValidAndInitializedSet
redeem	External	Can Modify State	nonReentrant onlyValidAndInitializedSet
updateFeeRecipient	External	Can Modify State	onlyManagerAndValidSet

DebtIssuanceModule			
updateIssueFee	External	Can Modify State	onlyManagerAndValidSet
updateRedeemFee	External	Can Modify State	onlyManagerAndValidSet
registerToIssuanceModule	External	Can Modify State	onlyModule onlyValidAndInitializedSet
unregisterFromIssuanceModule	External	Can Modify State	onlyModule onlyValidAndInitializedSet
initialize	External	Can Modify State	onlySetManager onlyValidAndPendingSet
removeModule	External	Can Modify State	-
calculateTotalFees	Public	-	-
getRequiredComponentIssuanceUnits	External	-	-
getRequiredComponentRedemptionUnits	External	-	-
getModuleIssuanceHooks	External	-	-
isModuleIssuanceHook	External	-	-
_calculateRequiredComponentIssuanceUnits	Internal	-	-
_getTotalIssuanceUnits	Internal	-	-
_resolveEquityPositions	Internal	Can Modify State	-
_resolveDebtPositions	Internal	Can Modify State	-
_resolveFees	Internal	Can Modify State	-
_callManagerPreIssueHooks	Internal	Can Modify State	-
_callModulePreIssueHooks	Internal	Can Modify State	-
_callModulePreRedeemHooks	Internal	Can Modify State	-

DebtIssuanceModule			
_executeExternalPositionHooks	Internal	Can Modify State	-

DebtIssuanceModuleV2			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	DebtIssuanceModule
issue	External	Can Modify State	nonReentrant onlyValidAndInitializedSet
redeem	External	Can Modify State	nonReentrant onlyValidAndInitializedSet
getRequiredComponentIssuanceUnits	External	-	-
_resolveEquityPositions	Internal	Can Modify State	-
_resolveDebtPositions	Internal	Can Modify State	-
_getTotalIssuanceUnitsFromBalances	Internal	-	-

TradeModule			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ModuleBase
initialize	External	Can Modify State	onlyValidAndPendingSet onlySetManager
estimateTrade	Public	-	-
trade	External	Can Modify State	nonReentrant onlyManagerAndValidSet
removeModule	External	Can Modify State	-
_createTradeInfo	Internal	-	-
_validatePreTradeData	Internal	-	-

TradeModule			
_executeTrade	Internal	Can Modify State	-
_validatePostTrade	Internal	-	-
_accrueProtocolFee	Internal	Can Modify State	-
_updateSetTokenPositions	Internal	Can Modify State	-

StreamingFeeModule			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ModuleBase
accrueFee	Public	Can Modify State	nonReentrant onlyValidAndInitializedSet
initialize	External	Can Modify State	onlySetManager onlyValidAndPendingSet
removeModule	External	Can Modify State	-
updateStreamingFee	External	Can Modify State	onlySetManager onlyValidAndInitializedSet
updateFeeRecipient	External	Can Modify State	onlySetManager onlyValidAndInitializedSet
getFee	External	-	-
get_profitSharingPercentage	External	-	-
_calculateStreamingFee	Internal	-	-
_calculateStreamingFeeInflation	Internal	-	-
_mintManagerAndProtocolFee	Internal	Can Modify State	-
_editPositionMultiplier	Internal	Can Modify State	-
_feeRecipient	Internal	-	-

StreamingFeeModule			
_lastStreamingFeeTimestamp	Internal	-	-
_maxStreamingFeePercentage	Internal	-	-
_streamingFeePercentage	Internal	-	-
_profitSharingPercentage	Internal	-	-

SignalSuscriptionModule			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ModuleBase
exectueFollowStart	External	Can Modify State	nonReentrant onlyManagerAndValidSet
exectueFollowEnd	External	Can Modify State	nonReentrant onlyManagerAndValidSet
setPlatformAndPlatformFee	External	Can Modify State	nonReentrant onlyOwner
initialize	External	Can Modify State	onlyValidAndPendingSet onlySetManager
removeModule	External	Can Modify State	-
subscribe	External	Can Modify State	nonReentrant onlyManagerAndValidSet
udpate_allowedCopytrading	External	Can Modify State	onlyManagerAndValidSet
unsubscribe	External	Can Modify State	nonReentrant onlyManagerAndValidSet
removeFollower	External	Can Modify State	nonReentrant onlyOwner
get_followers	External	-	-
get_signal_provider	External	-	-
handleFee	External	Can Modify State	nonReentrant onlyManagerAndValidSet

SignalSuscriptionModule			
desposit	Internal	Can Modify State	-
_updatePosition	Internal	Can Modify State	-

WrapModuleV2			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ModuleBase
wrap	External	Can Modify State	nonReentrant onlyManagerAndValidSet
wrapWithEther	External	Can Modify State	nonReentrant onlyManagerAndValidSet
unwrap	External	Can Modify State	nonReentrant onlyManagerAndValidSet
unwrapWithEther	External	Can Modify State	nonReentrant onlyManagerAndValidSet
initialize	External	Can Modify State	onlySetManager
removeModule	External	Can Modify State	-
_validateInputs	Internal	-	-
_validateWrapAndUpdate	Internal	Can Modify State	-
_validateUnwrapAndUpdate	Internal	Can Modify State	-
_createWrapDataAndInvoke	Internal	Can Modify State	-
_createUnwrapDataAndInvoke	Internal	Can Modify State	-
_updatePosition	Internal	Can Modify State	-
_updatePositionCoinType	Internal	Can Modify State	-

WrapModuleV2			
_snapshotTargetAssetsBalance	Internal	-	-

ProtocolViewer			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-

UniswapV2ExchangeAdapter			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
getTradeCalldata	External	-	-
getSpender	External	-	-

Controller			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	External	Can Modify State	onlyOwner
addSet	External	Can Modify State	onlyInitialized onlyFactory
removeSet	External	Can Modify State	onlyInitialized onlyOwner
addFactory	External	Can Modify State	onlyInitialized onlyOwner
removeFactory	External	Can Modify State	onlyInitialized onlyOwner
addModule	External	Can Modify State	onlyInitialized onlyOwner
removeModule	External	Can Modify State	onlyInitialized onlyOwner
addResource	External	Can Modify State	onlyInitialized onlyOwner
removeResource	External	Can Modify State	onlyInitialized onlyOwner

Controller			
addFee	External	Can Modify State	onlyInitialized onlyOwner
editFee	External	Can Modify State	onlyInitialized onlyOwner
editFeeRecipient	External	Can Modify State	onlyInitialized onlyOwner
getModuleFee	External	-	-
getFactories	External	-	-
getModules	External	-	-
getResources	External	-	-
getSets	External	-	-
isSystemContract	External	-	-

IntegrationRegistry			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
addIntegration	Public	Can Modify State	onlyOwner
batchAddIntegration	External	Can Modify State	onlyOwner
editIntegration	Public	Can Modify State	onlyOwner
batchEditIntegration	External	Can Modify State	onlyOwner
removeIntegration	External	Can Modify State	onlyOwner
getIntegrationAdapter	External	-	-
getIntegrationAdapterWithHash	External	-	-
isValidIntegration	External	-	-
_nameHash	Internal	-	-

SetValuer			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
calculateSetTokenValuation	External	-	-

UniswapPairPriceAdapter			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
getPrice	External	-	-
addPool	External	Can Modify State	onlyOwner
removePool	External	Can Modify State	onlyOwner
getAllowedUniswapPools	External	-	-
_getUniswapPrice	Internal	-	-

PriceOracleProxy			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
getUnderlyingPrice	Public	-	-
setSaiPrice	Public	Can Modify State	-

IdentityService			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
set_account_type	Public	Can Modify State	onlyOwner
newFund	Public	Can Modify State	onlyOwner onlyValidSet

IdentityService			
removeFund	Public	Can Modify State	onlyOwner onlyValidSet
removeAccount	Public	Can Modify State	onlyOwner
batchSet_account_type	External	Can Modify State	onlyOwner
getFundsByAccount	Public	-	-
getAccounts	External	-	-
_isSetValid	Internal	-	-

AaveV2WrapV2Adapter			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
getWrapCallData	External	-	_onlyValidTokenPair
getUnwrapCallData	External	-	_onlyValidTokenPair
getSpenderAddress	External	-	-
validTokenPair	Internal	-	-

IssuanceExtension			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	BaseGlobalExtension
distributeFees	Public	Can Modify State	-
initializeModule	External	Can Modify State	onlyOwnerAndValidManager
initializeExtension	External	Can Modify State	onlyOwnerAndValidManager
initializeModuleAndExtension	External	Can Modify State	onlyOwnerAndValidManager
removeExtension	External	Can Modify State	-

IssuanceExtension			
updateIssueFee	External	Can Modify State	onlyOwner
updateRedeemFee	External	Can Modify State	onlyOwner
updateFeeRecipient	External	Can Modify State	onlyOwner
_initializeModule	Internal	Can Modify State	-

StakingModule			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ModuleBase
stake	External	Can Modify State	onlyManagerAndValidSet
unstake	External	Can Modify State	onlyManagerAndValidSet
componentIssueHook	External	Can Modify State	onlyModule
componentRedeemHook	External	Can Modify State	onlyModule
moduleIssueHook	External	Can Modify State	-
moduleRedeemHook	External	Can Modify State	-
initialize	External	Can Modify State	onlySetManager onlyValidAndPendingSet
removeModule	External	Can Modify State	-
hasStakingPosition	Public	-	-
getStakingContracts	Public	-	-
getStakingPosition	Public	-	-
getStakingPositionUnit	Public	-	-
_stake	Internal	Can Modify	-

StakingModule			
		State	
_unstake	Internal	Can Modify State	-
_updateStakeState	Internal	Can Modify State	-
_updateUnstakeState	Internal	Can Modify State	-

SubscribeFeePool			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
setSetting	External	Can Modify State	onlyOwner
setTokenWhiteList	External	Can Modify State	onlyOwner
deposit	External	Can Modify State	onlyTokenWhiteList nonReentrant
witdrawAndJasperVault	External	Can Modify State	nonReentrant
witdrawAllAndJasperVault	External	Can Modify State	nonReentrant
witdraw	External	Can Modify State	nonReentrant
witdrawAll	External	Can Modify State	nonReentrant
getDepositInfo	External	-	-

ManagerCore			
Function Name	Visibility	Mutability	Modifiers
initialize	External	Can Modify State	onlyOwner
addExtension	External	Can Modify State	onlyInitialized onlyOwner
removeExtension	External	Can Modify State	onlyInitialized onlyOwner
addFactory	External	Can Modify State	onlyInitialized onlyOwner

ManagerCore			
removeFactory	External	Can Modify State	onlyInitialized onlyOwner
addManager	External	Can Modify State	onlyInitialized onlyFactory
removeManager	External	Can Modify State	onlyInitialized onlyOwner
getExtensions	External	-	-
getFactories	External	-	-
getManagers	External	-	-
_addExtension	Internal	Can Modify State	-
_addFactory	Internal	Can Modify State	-

StreamingFeeSplitExtension			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	BaseGlobalExtension
accrueFeesAndDistribute	Public	Can Modify State	-
initializeModule	External	Can Modify State	onlyOwnerAndValidManager
initializeExtension	External	Can Modify State	onlyOwnerAndValidManager
initializeModuleAndExtension	External	Can Modify State	onlyOwnerAndValidManager
removeExtension	External	Can Modify State	-
updateStreamingFee	External	Can Modify State	onlyOwner
updateFeeRecipient	External	Can Modify State	onlyOwner
_initializeModule	Internal	Can Modify State	-

DelegatedManagerFactory			
Function Name	Visibility	Mutability	Modifiers

DelegatedManagerFactory			
<Constructor>	Public	Can Modify State	-
createSetAndManager	External	Can Modify State	-
createManager	External	Can Modify State	-
initialize	External	Can Modify State	-
_deploySet	Internal	Can Modify State	-
_deployManager	Internal	Can Modify State	-
_initializeExtensions	Internal	Can Modify State	-
_setInitializationState	Internal	Can Modify State	-
_setManagerState	Internal	Can Modify State	-
_validateManagerParameters	Internal	-	-
_validateComponentsIncludedInAssetsList	Internal	-	-

CompoundWrapV2Adapter			
Function Name	Visibility	Mutability	Modifiers
getWrapCallData	External	-	-
getUnwrapCallData	External	-	-
getSpenderAddress	External	-	-

4.3 Vulnerability Summary

[N1] [Low] Missing check for path

Category: Design Logic Audit

Content

In the VaultPaymaster contract, the owner role can set the token exchange path, but there is no check on

_path[0]. If _path[0] is not equal to the _token parameter due to an administrator error when calling this function, it will cause an error in the token exchange operation.

Code location:

jasper-vaults-contracts/contracts/protocol/4337/VaultPaymaster.sol#L110-117

```
    function setToekenToPath(
        address _token,
        address[] calldata _path
    ) external onlyOwner {
        require(_path.length > 0, "path length greater than zero");
        token2path[_token] = _path;
        emit SetTokenPath(_token, _path);
    }
```

Solution

It is recommended to add a judgment: _path[0] == _token.

Status

Fixed

[N2] [Suggestion] The balance check is incorrect

Category: Design Logic Audit

Content

In the VaultPaymaster contract, The _postOp function resolves the sender by the incoming context, and then determines the balance of the sender's owner. However, the correct balance check should be `balance >= (charge + opFee)` instead of `balance >= charge`, because user2balance[owner] is equal to balance - charge - opFee.

Code location:

jasper-vaults-contracts/contracts/protocol/4337/VaultPaymaster.sol#L338

```
    function _postOp(
        PostOpMode mode,
        bytes calldata context,
        uint256 actualGasCost
    ) internal {
```

```
        ...
        require(balance >= charge, "Paymaster Post:not sufficient funds");
        ...
    }
```

Solution

It is recommended to change `balance >= charge` to `balance >= (charge + opFee)`.

Status

Fixed

[N3] [High] Missing check for ETH in the exchange path

Category: Design Logic Audit

Content

In the VaultPaymaster contract, the exchange of tokens is mainly done through the _swapExactTokensForETH and _swapExactETHForTokens functions.

1.In the _swapExactTokensForETH function, eth is added to the end of the exchange path, where it is missing to determine if the last token of _path is eth (i.e. _path[len - 1] == eth). This may cause an error when calling router's swapExactTokensForETH function.

2.There is a similar problem in the _swapExactETHForTokens function, except that eth is added at the beginning of the exchange path without checking that the first token of the incoming _path is eth (i.e. _path[0] == eth).

Code location:

jasper-vaults-contracts/contracts/protocol/4337/VaultPaymaster.sol#L217

```
    function _swapExactTokensForETH(
        address[] memory _path,
        uint256 _amountIn
    ) internal returns (uint256) {
        uint256 len = _path.length;
        address[] memory path = new address[](len + 1);
        for (uint256 i = 0; i < len; i++) {
            path[i] = _path[i];
        }
        path[len] = eth;
```

```
        ...
    }
```

jasper-vaults-contracts/contracts/protocol/4337/VaultPaymaster.sol#L237

```
    function _swapExactETHForTokens(
        address _to,
        address[] memory _path,
        uint256 _amountIn
    ) internal returns (uint256) {
        uint256 len = _path.length;
        address[] memory path = new address[](len + 1);
        path[0] = eth;
        uint256 index = 1;
        for (uint256 i = 0; i < len; i++) {
            path[index] = _path[len - 1 - i];
            index += 1;
        }
        ...
    }
```

Solution

It is recommended to add a judgment to the _swapExactTokensForETH function: `path[len - 1] ! = eth` and add a judgement to the _swapExactETHForTokens function: `path[0] ! = eth`.

Status

Fixed

[N4] [Critical] Error of the return value

Category: Design Logic Audit

Content

In the VaultPaymaster contract, the _swapExactTokensForETH and _swapExactETHForTokens functions called when swapping tokens use the value of amounts[1] as the amount of tokens returned after the swap. However, when there are three or more tokens in the path, it will result in the number returned not being the number of the last token in the exchange path. This can cause the balance of the user's deposit to be increased incorrectly.

Code location:

jasper-vaults-contracts/contracts/protocol/4337/VaultPaymaster.sol#L227

```
function _swapExactTokensForETH(
    address[] memory _path,
    uint256 _amountIn
) internal returns (uint256) {
    ...
    uint[] memory amounts = router.swapExactTokensForETH(
        _amountIn,
        0,
        path,
        address(this),
        deadline
    );
    return amounts[1];
}
```

jasper-vaults-contracts/contracts/protocol/4337/VaultPaymaster.sol#L250

```
function _swapExactETHForTokens(
    address _to,
    address[] memory _path,
    uint256 _amountIn
) internal returns (uint256) {
    ...
    uint[] memory amounts = router.swapExactETHForTokens{value: _amountIn}(
        0,
        path,
        _to,
        deadline
    );
    return amounts[1];
}
```

Solution

It is recommended to return `amounts[amouts.length-1]` instead of `amounts[1]`.

Status

Fixed

[N5] [Medium] Sandwich Attack Risk

Category: Reordering Vulnerability

Content

In the VaultPaymaster contract, the exchange of eth and other tokens is done through the _swapExactTokensForETH and _swapExactETHForTokens functions. But the passed in `amountOutMin` parameter is 0 during the swap, which will cause the exchange process to be subject to a sandwich attack.

Code location:

jasper-vaults-contracts/contracts/protocol/4337/VaultPaymaster.sol#L222

```
function _swapExactTokensForETH(
    address[] memory _path,
    uint256 _amountIn
) internal returns (uint256) {
    ...
    uint[] memory amounts = router.swapExactTokensForETH(
        _amountIn,
        0,
        path,
        address(this),
        deadline
    );
    ...
}
```

jasper-vaults-contracts/contracts/protocol/4337/VaultPaymaster.sol#L245

```
function _swapExactETHForTokens(
    address _to,
    address[] memory _path,
    uint256 _amountIn
) internal returns (uint256) {
    ...
    uint[] memory amounts = router.swapExactETHForTokens{value: _amountIn}(
        0,
        path,
        _to,
        deadline
    );
    ...
}
```

Solution

It is recommended to add a check for slippage to ensure that amountOutMin meets user expectations.

Status

Fixed

[N6] [Suggestion] Missing zero address check

Category: Others

Content

1.In the VaultPaymaster contract, here the zero address check is missing in the initialize function. If the entryPoint and router parameters are passed to the zero address due to a mistake during initialization, and there is no function in the contract that can modify these values again, then the contract will have problems running.

Code location:

jasper-vaults-contracts/contracts/protocol/4337/VaultPaymaster.sol#L72-90

```
    function initialize(
        IEntryPoint _entryPoint,
        IUniswapV2Router _router,
        address _weth,
        uint256 _fee,
        ISignalSuscriptionExtension _signalSuscriptionExtension,
        IDelegatedManagerFactory _delegatedManagerFactory
    ) public initializer {
        entryPoint = _entryPoint;
        router = _router;
        eth = _weth;
        fee = _fee;
        signalSuscriptionExtension = _signalSuscriptionExtension;
        delegatedManagerFactory = _delegatedManagerFactory;
        __Ownable_init();
        __UUPSUpgradeable_init();
        DIVISOR = 10 ** 18;
        ETH_TOKEN_ADDRESS = 0x0000000000000000000000000000000000000000;
    }
```

2.In the DelegatedManagerFactory contract, here the zero address check is missing in the createSetAndManager and createManager function. If the _owner parameter is passed to the zero address due to a mistake, and there is no function in the contract that can modify the value again. This will result in the loss of administrative privileges for the manager contract.

Code location:

jasper-vaults-contracts/contracts/manager/factories/DelegatedManagerFactory.sol#L136-234

```
function createSetAndManager(
    uint256 _vaultType,
    address[] memory _components,
    int256[] memory _units,
    string memory _name,
    string memory _symbol,
    address _owner,
    address _methodologist,
    address[] memory _modules,
    address[] memory _adapters,
    address[] memory _operators,
    address[] memory _assets,
    address[] memory _extensions
) external returns (IJasperVault, address) {
    ...
    _setInitializationState(
        jasperVault,
        address(manager),
        _owner,
        _methodologist
    );
    ...
}
```

```
function createManager(
    IJasperVault _jasperVault,
    address _owner,
    address _methodologist,
    address[] memory _adapters,
    address[] memory _operators,
    address[] memory _assets,
    address[] memory _extensions
) external returns (address) {
    ...
    _setInitializationState(
        _jasperVault,
        address(manager),
        _owner,
        _methodologist
    );
   ...
}
```

Solution

It is recommended to add the check for zero address.

Status

Fixed

[N7] [Suggestion] call() should be used instead of transfer()

Category: Gas Optimization Audit

Content

The transfer() and send() functions forward a fixed amount of 2300 gas. Historically, it has often been recommended to use these functions for value transfers to guard against reentrancy attacks. However, the gas cost of EVM instructions may change significantly during hard forks which may break already deployed contract systems that make fixed assumptions about gas costs. For example. EIP 1884 broke several existing smart contracts due to a cost increase of the SLOAD instruction.

Code Location:

jasper-vaults-contracts/contracts/protocol/module/v1/NavIssuanceModule.sol#L416

```
    function redeemIntoEther(
        IJasperVault _jasperVault,
        uint256 _setTokenQuantity,
        uint256 _minReserveReceiveQuantity,
        address payable _to
    ) external nonReentrant onlyValidAndInitializedSet(_jasperVault) {
        ...

        weth.withdraw(redeemInfo.netFlowQuantity);

        _to.transfer(redeemInfo.netFlowQuantity);

        ...
    }
```

Solution

It is recommended to use call() instead of transfer(), but be sure to respect the CEI pattern and/or add re-entrancy guards.

Status

Fixed

[N8] [Low] Redundant code

Category: Others

Content

In the NavIssuanceModule contract, the code logic of the _getIssuePremium and _getRedeemPremium functions is the same: both get the `navIssuanceSettings[_jasperVault].premiumPercentage`.

Code Location:

jasper-vaults-contracts/contracts/protocol/module/v1/NavIssuanceModule.sol#L1031-1049

```
    function _getIssuePremium(
        IJasperVault _jasperVault,
        address, /* _reserveAsset */
        uint256 /* _reserveAssetQuantity */
    ) internal view virtual returns (uint256) {
        return navIssuanceSettings[_jasperVault].premiumPercentage;
    }

    function _getRedeemPremium(
        IJasperVault _jasperVault,
        address, /* _reserveAsset */
        uint256 /* _setTokenQuantity */
    ) internal view virtual returns (uint256) {
        return navIssuanceSettings[_jasperVault].premiumPercentage;
    }
```

Solution

It is recommended to merge these two functions into the same function.

Status

Fixed

[N9] [Low] Unclear function logic

Category: Others

Content

In the StakingModule contract, the user can remove this module from the JasperVault through the

removeModule function. But there is only one judgment in the function without any action to clear the user state.

Code Location:

jasper-vaults-contracts/contracts/protocol/module/v1/StakingModule.sol

```
    function removeModule() external override {
        IJasperVault jasperVault = IJasperVault(msg.sender);
        address[] memory components = jasperVault.getComponents();
        for (uint256 i = 0; i < components.length; i++) {
            require(
                stakingPositions[jasperVault][IERC20(components[i])]
                    .stakingContracts
                    .length == 0,
                "Open positions must be closed"
            );
        }
    }
```

Solution

It is recommended to add code to delete user status according to contract logic.

Status

Fixed

[N10] [Low] Missing return value check

Category: Others

Content

In the AaveLeverageModule contract, the _withdraw and _repayBorrow function will will call _jasperVault's invokeWithdraw and invokeRepay functions to make withdrawals and repayments. But the check of return values is missing here.

Code Location:

jasper-vaults-contracts/contracts/protocol/module/v1/AaveLeverageModule.sol#L945-978

```
    function _withdraw(
        IJasperVault _jasperVault,
        ILendingPool _lendingPool,
```

```
        IERC20 _asset,
        uint256 _notionalQuantity
    ) internal {
        _jasperVault.invokeWithdraw(
            _lendingPool,
            address(_asset),
            _notionalQuantity
        );
    }

    function _repayBorrow(
        IJasperVault _jasperVault,
        ILendingPool _lendingPool,
        IERC20 _asset,
        uint256 _notionalQuantity
    ) internal {
        _jasperVault.invokeApprove(
            address(_asset),
            address(_lendingPool),
            _notionalQuantity
        );
        _jasperVault.invokeRepay(
            _lendingPool,
            address(_asset),
            _notionalQuantity,
            BORROW_RATE_MODE
        );
    }
```

Solution

It is recommended to add a check for return values.

Status

Fixed

[N11] [Suggestion] Lack of event records

Category: Others

Content

There is no corresponding event logged when a sensitive parameter in the contract is modified.

Code Location:

jasper-vaults-contracts/contracts/protocol/4337/VaultFactory.sol

```
    function setSetting(IEntryPoint _entryPoint,IDelegatedManagerFactory
_delegatedManagerFactory) external onlyOwner{
        ...
    }
```

Code Location:

jasper-vaults-contracts/contracts/protocol/4337/VaultPaymaster.sol

```
    function setSetting(
        uint256 _fee,
        address _weth,
        address _eth,
        ISignalSuscriptionExtension _signalSuscriptionExtension,
        IDelegatedManagerFactory _delegatedManagerFactory
    ) external onlyOwner {
        ...
    }
```

Code Location:

jasper-vaults-contracts/contracts/protocol/module/v1/SignalSuscriptionModule.sol

```
    function setPlatformAndPlatformFee(
        address _platform_vault,
        uint256 _fee
    ) external nonReentrant onlyOwner {
        ...
    }

    function removeFollower(
        address target,
        address follower
    ) external nonReentrant onlyOwner {
        ...
    }
```

Code Location:

jasper-vaults-contracts/contracts/protocol/integration/oracles/UniswapPairPriceAdapter.sol

```
    function addPool(address _poolAddress) external onlyOwner {
        ...
```

```
    }

    function removePool(address _poolAddress) external onlyOwner {
        ...
    }
```

Code Location:

jasper-vaults-contracts/contracts/protocol/IdentityService.sol

```
    function set_account_type(address account, uint8 value) public onlyOwner {
        ...
    }

    function newFund(address account, address fund)
        public
        onlyOwner
        onlyValidSet(fund)
    {
        ...
    }

    function removeFund(address account, address fund)
        public
        onlyOwner
        onlyValidSet(fund)
    {
        ...
    }

    function removeAccount(address account) public onlyOwner {
        ...
    }
```

Solution

It is recommended to record events and 0 address checks when sensitive parameters are modified for self-inspection or community review.

Status

Fixed

[N12] [Suggestion] ERC20 return values not checked

Category: Others

Content

Some tokens (like USDT) don't correctly implement the EIP20 standard and their transfer/transferFrom function return void instead of a successful boolean. Calling these functions with the correct EIP20 function signatures will always revert.

Code location:

jasper-vaults-contracts/contracts/protocol/SubscribeFeePool.sol

```
function deposit(
    address _token,
    address _to,
    uint256 _amount
) external onlyTokenWhiteList(_token) nonReentrant {
    if (_amount > 0) {
        IERC20(_token).transferFrom(msg.sender, address(this), _amount);

        ...

    }
}

//withdraw   JasperVault
function witdrawAndJasperVault(
    address _token,
    address _jasperVault,
    address _to,
    uint256 _amount
) external nonReentrant {
    ...
    IERC20(_token).transfer(_to, _amount);
    ...
}

function witdrawAllAndJasperVault(
    address _jasperVault,
    address _to
) external nonReentrant {
    ...

    for (uint256 i = 0; i < compounts.length; i++) {
        if (amounts[i] > 0) {
            DepositInfos[_jasperVault].amounts[i] = 0;
            IERC20(compounts[i]).transfer(_to, amounts[i]);
```

```
            ...

        }
    }
}

// //withdraw
function witdraw(
    address _token,
    address _to,
    uint256 _amount
) external nonReentrant {

    ...

    IERC20(_token).transfer(_to, _amount);
    emit Witdraw(msg.sender, _to, _amount);
}

function witdrawAll(address _to) external nonReentrant {
    ...

    for (uint256 i = 0; i < compounts.length; i++) {
        if (amounts[i] > 0) {
            DepositInfos[msg.sender].amounts[i] = 0;
            IERC20(compounts[i]).transfer(_to, amounts[i]);
            emit Witdraw(msg.sender, _to, amounts[i]);
        }
    }
}
```

Solution

It is recommended to use OpenZeppelin's SafeERC20 versions with the safeTransfer and safeTransferFrom functions that handle the return value check as well as non-standard-compliant tokens.

Status

Fixed

[N13] [High] Missing validation of follow trading

Category: Authority Control Vulnerability Audit

Content

In the CopyTradingExtension contract, users can call the batchTradeWithFollowers function to perform trade

operations for their followers. Here the followers are trading information that is passed in externally by the user to whom they are subscribed. But the slippage and the _data(tokens exchange path) are manipulated by him, and the followers are not asked to verify the transaction information.

This will result in the user being able to pass in malicious tokens or tokens with low liquidity in the exchange path and extremely high slippage to drain the followers' funds and arbitrage himself.

Code location: jasper-vaults-contracts/contracts/manager/extensions/CopyTradingExtension.sol#L261-296

```
    function batchTradeWithFollowers(
        IJasperVault _jasperVault,
        TradeInfo[] memory _trades
    ) external
       onlySettle(_jasperVault)
       onlyOperator(_jasperVault) {
        ...
        address[] memory followers = signalSuscriptionModule.get_followers(
            address(_jasperVault)
        );
        uint256 tradesLength = _trades.length;
        for (uint256 i = 0; i < tradesLength; i++) {

if(!ValidAdapterByModule(_jasperVault,address(tradeModule),_trades[i].exchangeName)){
               continue;
            }
            _executeTrade(_jasperVault, _trades[i]);
            for (uint256 m = 0; m < followers.length; m++) {
                TradeInfo memory newTrade = TradeInfo({
                    exchangeName: _trades[i].exchangeName,
                    sendToken: _trades[i].sendToken,
                    sendQuantity: _trades[i].sendQuantity,
                    receiveToken: _trades[i].receiveToken,
                    receiveQuantity: _trades[i].receiveQuantity,
                    data: _trades[i].data,
                    isFollower: true
                });
                _executeTrade(IJasperVault(followers[m]), newTrade);

            }
        }
    }
```

Solution

Add a whitelist of the tokens exchange path and slippage to jasperVault or its manager, and take the whitelist

values set by each follower from their manager contracts for verification during follow trading.

Status

Acknowledged; The project team response: Follow-the-trade is an advanced feature only available to institutional users. An institutional user is responsible to appoint own dedicated trader by whitelisting feature and only be able to follow the trade made by the appointed trader. The institution holds the right to take proper actions against the trader in the situation of suspicion of receiving malicious tokens or tokens with low liquidity in the exchange path and extremely high slippage to drain the followers' funds arbitrage himself.

[N14] [High] Missing validation of follow the leverage

Category: Authority Control Vulnerability Audit

Content

In the LeverageExtension contract, users can call the leverFollowers function to overlay leverage for their followers. But the number of tokens borrowed when followers leveraged up and the slippage of subsequent transactions lack whitelist checks(these parameters are manipulated by the users to whom they are subscribed). This could lead to a scenario where there is an impact on the subscriber's assets: if the follower is close to the liquidation line at this point in their liability profile, then if the subscriber they are subscribed to passes in a larger amount of borrowing, then they could be in a liquidated state in the next block. At this point they will be liquidated and lose assets.

Refer to issue N7 for the case of missing sliding point check.

Code location:

jasper-vaults-contracts/contracts/manager/extensions/LeverageExtension.sol

```
    function leverFollowers(
        IJasperVault _jasperVault,
        IERC20 _borrowAsset,
        IERC20 _collateralAsset,
        uint256 _borrowQuantityUnits,
        uint256 _minReceiveQuantityUnits,
        string memory _tradeAdapterName,
        bytes memory _tradeData
    )
        external
        onlySettle(_jasperVault)
```

```
        onlyOperator(_jasperVault)
        ValidAdapter(_jasperVault, address(leverageModule), _tradeAdapterName)
    {
        ...
_executeFollower(ILeverageModule.lever.selector,_jasperVault,_borrowAsset,_collateral
Asset,_borrowQuantityUnits,_minReceiveQuantityUnits,_tradeAdapterName,_tradeData);
    }

    ...

    function _executeFollower(
        bytes4  selector,
        IJasperVault _jasperVault,
        IERC20 _assetsOne,
        IERC20 _assetsTwo,
        uint256 _quantityUnits,
        uint256 _minQuantityUnits,
        string memory _tradeAdapterName,
        bytes memory _tradeData
    ) internal {
           address[] memory followers =
signalSuscriptionModule.get_followers(address(_jasperVault));
            for (uint256 i = 0; i < followers.length; i++) {
                bytes memory callData = abi.encodeWithSelector(
                    selector,
                    IJasperVault(followers[i]),
                    _assetsOne,
                    _assetsTwo,
                    _quantityUnits,
                    _minQuantityUnits,
                    _tradeAdapterName,
                    _tradeData
                );
                _execute(
                    _manager(IJasperVault(followers[i])),
                    address(leverageModule),
                    callData
                );
            }
     }
```

Solution

It is recommended to add a determination of borrowing volume and own health factor and a whitelist check for slippage to the manager contracts of the followers.

Status

Acknowledged; The project team response: Follow-the-trade is an advanced feature only available to institutional users. An institutional user is responsible to appoint own dedicated trader by whitelisting feature and only be able to follow the trade made by the appointed trader. The institution holds the right to take proper actions against the trader in the situation of suspicion of receiving malicious tokens or tokens with low liquidity in the exchange path and extremely high slippage to drain the followers' funds arbitrage himself.

[N15] [Medium] The DoS issue

Category: Denial of Service Vulnerability

Content

There is no maximum limit on the number of followers when subscribing, which causes the DoS issue when following operations (such as trade or lever) because of the excessive number of loops.

Code location:

jasper-vaults-contracts/contracts/manager/extensions/CopyTradingExtension.sol

```
    function batchTradeWithFollowers(
        IJasperVault _jasperVault,
        TradeInfo[] memory _trades
    ) external
       onlySettle(_jasperVault)
       onlyOperator(_jasperVault) {
        ...
        address[] memory followers = signalSuscriptionModule.get_followers(
            address(_jasperVault)
        );
            ...
            for (uint256 m = 0; m < followers.length; m++) {
                ...
            }
        }
    }
```

Code location:

jasper-vaults-contracts/contracts/manager/extensions/LeverageExtension.sol

```
    function _executeFollower(
            ...
    ) internal {
```

```
            address[] memory followers =
signalSuscriptionModule.get_followers(address(_jasperVault));
            for (uint256 i = 0; i < followers.length; i++) {
                ...
            }
     }

    function _executeDeleverToZeroFollower(
            ...
    ) internal {
           address[] memory followers =
signalSuscriptionModule.get_followers(address(_jasperVault));
            for (uint256 i = 0; i < followers.length; i++) {
                ...
            }
    }
```

Code location:

jasper-vaults-contracts/contracts/manager/extensions/StakingExtension.sol

```
    function stakeWithFollowers(
        ...
    )
        external
        onlySettle(_jasperVault)
        onlyOperator(_jasperVault)
        ValidAdapter(_jasperVault, address(stakingModule), _adapterName)
    {
        ...
        address[] memory followers = signalSuscriptionModule.get_followers(
            address(_jasperVault)
        );
        for (uint256 i = 0; i < followers.length; i++) {
            ...
        }
    }

    function unstakeWithFollowers(
        ...
    )
        external
        onlySettle(_jasperVault)
        onlyOperator(_jasperVault)
        ValidAdapter(_jasperVault, address(stakingModule), _adapterName)
    {
        ...
        address[] memory followers = signalSuscriptionModule.get_followers(
```

```
            address(_jasperVault)
        );
        for (uint256 i = 0; i < followers.length; i++) {
            ...
        }
    }
```

Code location:

jasper-vaults-contracts/contracts/manager/extensions/WrapExtension.sol

```
    function _executeWrapWithFollowers(
        IJasperVault _jasperVault,
        WrapInfo memory _wrapInfo
    ) internal {
        address[] memory followers = signalSuscriptionModule.get_followers(
            address(_jasperVault)
        );
        for (uint256 i = 0; i < followers.length; i++) {
            ...
        }
    }


    ...

    function _executeWrapEtherWithFollowers(
        IJasperVault _jasperVault,
        WrapInfo memory _wrapInfo
    ) internal {
        address[] memory followers = signalSuscriptionModule.get_followers(
            address(_jasperVault)
        );
        for (uint256 i = 0; i < followers.length; i++) {
            ...
        }
    }

    function _executeUnwrapWithFollowers(
        IJasperVault _jasperVault,
        UnwrapInfo memory _unwrapInfo
    ) internal {
        address[] memory followers = signalSuscriptionModule.get_followers(
            address(_jasperVault)
        );
        for (uint256 i = 0; i < followers.length; i++) {
            ...
        }
```

```
    }

    function _executeUnwrapEtherWithFollowers(
        IJasperVault _jasperVault,
        UnwrapInfo memory _unwrapInfo
    ) internal {
        address[] memory followers = signalSuscriptionModule.get_followers(
            address(_jasperVault)
        );
        for (uint256 i = 0; i < followers.length; i++) {
            ...
        }
    }
```

Solution

It is recommended to add a maximum limit on the number of followers when subscribing or limit the number of for loops to avoid DoS.

Status

Fixed

[N16] [Suggestion] Possible overflow risk

Category: Integer Overflow and Underflow Vulnerability

Content

In the TradeModule contract, The user can predict the result of a trade by calling the estimateTrade function. But in the calculation of the result will deduct the fee: result[1]-=protocolFeeTotal, because the contract version is 0.6.10, if the fee is not capped will lead to overflow problems.

Code location:

jasper-vaults-contracts/contracts/protocol/module/v1/TradeModule.sol#L135

```
    function estimateTrade(
        IJasperVault _jasperVault,
        string memory _exchangeName,
        address _sendToken,
        uint256 _sendQuantity,
        address _receiveToken,
        uint256 _minReceiveQuantity)
        public view
         returns(TradeInfo memory,uint[] memory){
```

```
        ...
         uint256 protocolFeeTotal = getModuleFee(TRADE_MODULE_PROTOCOL_FEE_INDEX,
result[1]);
         result[1]-=protocolFeeTotal;
         return (tradeInfo,result);
    }
```

Solution

It is recommended to use the sub function of the safeMath library for subtraction operations.

Status

Fixed

[N17] [Medium] Risk of excessive authority

Category: Authority Control Vulnerability Audit

Content

1.In the SignalSuscriptionModule contract, the owner role can set the platformFee and remove any user's followers through the setPlatformAndPlatformFee and removeFollower function. If permissions are lost, it may pose a threat to the contract.

Code Location:

jasper-vaults-contracts/contracts/protocol/module/v1/SignalSuscriptionModule.sol

```
    function setPlatformAndPlatformFee(
        address _platform_vault,
        uint256 _fee
    ) external nonReentrant onlyOwner {
        platformFee = _fee;
        platform_vault = _platform_vault;
    }

    function removeFollower(
        address target,
        address follower
    ) external nonReentrant onlyOwner {
        followers[target].removeStorage(follower);
        delete Signal_provider[follower];
    }
```

2.In the Controller contract, the owner can add and modify fees. If permissions are lost, it may pose a threat to the contract.

Code Location:

jasper-vaults-contracts/contracts/protocol/Controller.sol#291-316

```
    function addFee(address _module, uint256 _feeType, uint256 _newFeePercentage)
external onlyInitialized onlyOwner {
        require(isModule[_module], "Module does not exist");

        require(fees[_module][_feeType] == 0, "Fee type already exists on module");

        fees[_module][_feeType] = _newFeePercentage;

        emit FeeEdited(_module, _feeType, _newFeePercentage);
    }

    function editFee(address _module, uint256 _feeType, uint256 _newFeePercentage)
external onlyInitialized onlyOwner {
        require(isModule[_module], "Module does not exist");

        require(fees[_module][_feeType] != 0, "Fee type does not exist on module");

        fees[_module][_feeType] = _newFeePercentage;

        emit FeeEdited(_module, _feeType, _newFeePercentage);
    }
```

3.In the PriceOracleProxy contract, the guardian role can set the SAI price. If permissions are lost, it may pose a threat to the contract.

Code Location:

jasper-vaults-contracts/external/contracts/compund/PriceOracleProxy.sol#L106-111

```
    function setSaiPrice(uint price) public {
        require(msg.sender == guardian, "only guardian may set the SAI price");
        require(saiPrice == 0, "SAI price may only be set once");
        require(price < 0.1e18, "SAI price must be < 0.1 ETH");
        saiPrice = price;
    }
```

Solution

It is recommended to transfer the permissions of the owner and guardian role to the multi-signature wallet management or the community governance.

Status

Fixed; The transactions relating to the transfer of permissions to time-locked contracts are as follows:

https://polygonscan.com/tx/0xdf200d4c7c385de9070bc820fd30301cb697adfeab663d7d33814eb53e411e49

https://polygonscan.com/tx/0xc78abc0dc94ca2bdd94714dbeba82ca208082dc729515f6fd3450aa1ff0a8bbb

https://polygonscan.com/tx/0x4f17b146725d20effbe6a210303aed6ff7adae1cac73d9c548f114617d5a2402

https://polygonscan.com/tx/0x8c600508b1c832f80861cb168687cf5f3686759ed1eb7fea03c79045f855160b

https://polygonscan.com/tx/0x0dcefcc204004827d2daa5bae9a24a8ccbd45e5651ae28b7981acfdb13a87092

https://polygonscan.com/tx/0x9831e92af351d320a357ef9dbff7773592ebb1dcfd17f89d8539780e4436bcef

https://polygonscan.com/tx/0x685f23599266792425b4c1f1cf31ae13b59579ee851d771ebc6507c732c0abd4

https://polygonscan.com/tx/0xa88a6759fa4ff4a4dbc14395566be1905bcda277fd25f60246519abe77843c85

https://polygonscan.com/tx/0xc7193970f96ba1b01516a443c4b5250a61507f0136be4a7e7eb7aa9d5facbe49

https://polygonscan.com/tx/0x313d30bee74799cfb84a2571171cf88221f85201302983373e1a1ab363ca8ff3

https://polygonscan.com/tx/0x906a1037c47e38ca5761e7cfef381fd8831720fe9b66ede47dacdd4cd6e3eebc

https://polygonscan.com/tx/0xd2b2194a126386450635bafc83647a12a634443203093487e7271f9b67e9d891

[N18] [Medium] Lack of scope restrictions

Category: Others

Content

1.In the SignalSuscriptionModule contract, the owner role can set the platformFee through the setPlatformAndPlatformFee function. However, the lack of a range limit when setting fees may cause unintended errors if set too large.

Code Location:

jasper-vaults-contracts/contracts/protocol/module/v1/SignalSuscriptionModule.sol#L108-114

```
    function setPlatformAndPlatformFee(
        address _platform_vault,
        uint256 _fee
```

```
    ) external nonReentrant onlyOwner {
        platformFee = _fee;
        platform_vault = _platform_vault;
    }
```

2.In the Controller contract, the owner can add and modify fees. However, the lack of a range limit when setting fees may cause unintended errors if set too large.

Code Location:

jasper-vaults-contracts/contracts/protocol/Controller.sol#291-316

```
    function addFee(address _module, uint256 _feeType, uint256 _newFeePercentage)
external onlyInitialized onlyOwner {
        require(isModule[_module], "Module does not exist");

        require(fees[_module][_feeType] == 0, "Fee type already exists on module");

        fees[_module][_feeType] = _newFeePercentage;

        emit FeeEdited(_module, _feeType, _newFeePercentage);
    }

    function editFee(address _module, uint256 _feeType, uint256 _newFeePercentage)
external onlyInitialized onlyOwner {
        require(isModule[_module], "Module does not exist");

        require(fees[_module][_feeType] != 0, "Fee type does not exist on module");

        fees[_module][_feeType] = _newFeePercentage;

        emit FeeEdited(_module, _feeType, _newFeePercentage);
    }
```

Solution

It is recommended to add range limits when setting fees.

Status

Fixed

[N19] [Low] Wrong variable obtained

Category: Scoping and Declarations Audit

Content

In the SignalSuscriptionExtension contract, the users can get the value of the warningLine through the warnLine function. But this function returns the value of warnLine instead of the warningLine.

Code Location:

jasper-vaults-contracts/contracts/manager/extensions/SignalSuscriptionExtension.sol#L254

```
    function warnLine() external  view returns(uint256){
        return signalSuscriptionModule.warnLine();
    }
```

Solution

It is recommended to change `signalSuscriptionModule.warnLine()` to `signalSuscriptionModule.warningLine()`.

Status

Fixed

[N20] [Low] Lack of authority control

Category: Authority Control Vulnerability Audit

Content

In the SignalSuscriptionExtension contract, anyone can call the exectueFollowEnd function to execute the exectueFollowEnd function of the SignalSuscriptionModule contract via any jasperVault passed in. This may affect up-chain event logging and down-chain scripts that fetch data to run.

Code Location: jasper-vaults-contracts/contracts/manager/extensions/SignalSuscriptionExtension.sol#L232-242

```
    function exectueFollowEnd(address _jasperVault) external {
         bytes memory callData = abi.encodeWithSelector(
            ISignalSuscriptionModule.exectueFollowEnd.selector,
            _jasperVault
        );
        _invokeManager(
            _manager(IJasperVault(_jasperVault)),
            address(signalSuscriptionModule),
            callData
```

```
        );
    }
```

Solution

It is recommended to add a permission check: only the owner corresponding to the jaspervault passed in is allowed to call the function.

Status

Fixed

[N21] [High] Missing variable updates when unsubscribing

Category: Design Logic Audit

Content

In the SignalSuscriptionExtension contract, when the fund manager calls the editFeeAndInfo function it unsubscribes the current follower, but the `currentSubscribeNumber[IJasperVault(target)]` variable is not updated at this time. This may cause other clients to fail the following check when subscribing again: `currentSubscribeNumber[_jasperVault]<allowMaxSubscribe[_jasperVault]`.

Code Location:

jasper-vaults-contracts/contracts/manager/extensions/SignalSuscriptionExtension.sol#L190-221

```
    function editFeeAndInfo(
        IJasperVault _jasperVault,
        address _masterToken,
        uint256 _followFee,
        uint256 _profitShareFee,
        uint256 _delay
    ) external onlyReset(_jasperVault) onlyOperator(_jasperVault) {
        address[] memory followers = signalSuscriptionModule.get_followers(
            address(_jasperVault)
        );
        for (uint256 i = 0; i < followers.length; i++) {
            bytes memory callData = abi.encodeWithSelector(
                ISignalSuscriptionModule.unsubscribe.selector,
                followers[i],
                address(_jasperVault)
            );
            _invokeManager(
                _manager(IJasperVault(followers[i])),
                address(signalSuscriptionModule),
```

```
                callData
            );
            _manager(IJasperVault(followers[i])).setSubscribeStatus(2);
            emit SetSubscribeStatus(IJasperVault(followers[i]), 2);
        }

       ...

    }
```

Solution

It is recommended to add an update to this variable in the unsubscribe loop:

`currentSubscribeNumber[IJasperVault(target)] -= 1`.

Status

Fixed

[N22] [Medium] Lack of judgement on EOA address

Category: Design Logic Audit

Content

In the SignalSuscriptionExtension contract, the checkWhiteList function will call the owner function of msg.sender to perform the whitelist check, but if the jasperVault operator role is an EOA address, this will cause the revert.

Code Location:

jasper-vaults-contracts/contracts/manager/extensions/SignalSuscriptionExtension.sol#L369

```
    function checkWhiteList(IJasperVault _jasperVault) internal  {
        ...

        address owner=IOwnable(msg.sender).owner();
        address[] memory list=whiteList[_jasperVault];
        bool isExist=list.contains(owner);
        require(isExist,"user is not in the whitelist");
    }
```

Solution

It is recommended to add a judgment on whether msg.sender is a contract.

Status

Fixed

[N23] [High] Lack of judgment on the upper limit of costs

Category: Design Logic Audit

Content

In the SignalSuscriptionModule contract, the user calls the handleFee function to settle the various fees required to follow the order. when there is a profit, a platform fee and a strategy fee are charged based on the portion of the profit. However, there is no limit to the sum of these two fees, and when the sum of these two fees exceeds 10 ** 18 (platformFee + _strategistFee <= 10 ** 18), this will consume the user's principal, which is not expected.

Code Location: jasper-vaults-contracts/contracts/protocol/module/v1/SignalSuscriptionModule.sol#L208-242

```
function handleFee(
    IJasperVault _jasperVault
) external nonReentrant onlyManagerAndValidSet(_jasperVault) {
    ...
    uint256 strategistFeeBalance=preBalance.preciseMul(followFee);
    if (nextBalance > preBalance) {
        //Calculated profit
        uint256   fee=nextBalance-preBalance;
        //calculate platformFee
        uint256 platformFeeBalance = fee.preciseMul(platformFee);
        if(platformFeeBalance>0){
            //approve
            _jasperVault.invokeApprove(
                masterToken,
                address(subscribeFeePool),
                platformFeeBalance
            );
            desposit(
                _jasperVault,
                masterToken,
                platform_vault,
                platformFeeBalance
            );
        }
        //calculate strategistsFee
```

```
            uint256 _strategistFee=profitShareFees[address(_jasperVault)];
            strategistFeeBalance
=strategistFeeBalance.add(fee.preciseMul(_strategistFee));
        }
        ...
    }
```

Solution

It is recommended to add a limit on the sum of these two fees.

Status

Fixed

[N24] [Medium] Lack of Token Whitelist Check and zero value check

Category: Design Logic Audit

Content

In the SubscribeFeePool contract, anyone can call the desposit function to make a deposit for a given account. However the contract lacks a token whitelist restriction, so anyone can increase the length of the compounts and amounts arrays by transferring tokens of their own creation or worthless tokens for the specified account or let the value of the passed in _amount parameter be 0. This can lead to dos problems when withdrawing due to excessive length of for loop traversal, and eventually withdrawals fail due to out of gas.

Code location:

jasper-vaults-contracts/contracts/protocol/SubscribeFeePool.sol#L57-69

```
    function desposit(address _token,address _to,uint256 _amount) external
nonReentrant {
        IERC20(_token).transferFrom(msg.sender, address(this), _amount);
        address[] memory compounts=despositInfos[msg.sender].compounts;
        (uint256 index,bool exist)=compounts.indexOf(_token);
        if(exist){
            uint256 balance=despositInfos[_to].amounts[index];
            despositInfos[_to].amounts[index]=balance.add(_amount);
        }else{
            despositInfos[_to].compounts.push(_token);
            despositInfos[_to].amounts.push(_amount);
        }
        emit Desposit(msg.sender,_to,_amount);
    }
```

Solution

It is recommended to add a whitelist check for incoming tokens and a zero value check for incoming _amount parameter.

Status

Fixed

[N25] [Medium] The incorrect variable is taken in the judgment

Category: Design Logic Audit

Content

In the SubscribeFeePool contract, anyone can call the desposit function to make a deposit for a given account, and the contract will take the value from despositInfos to determine whether the transferred tokens exist or not, and if not, add information about the tokens. However, the value taken here is `despositInfos[msg.sender].compounts` instead of `despositInfos[to].compounts`, this can cause duplicate tokens to be added to the specified address and the dos issue when withdrawing due to the large array length.

Code location:

jasper-vaults-contracts/contracts/protocol/SubscribeFeePool.sol#L59

```
    function desposit(address _token,address _to,uint256 _amount) external
nonReentrant {
        IERC20(_token).transferFrom(msg.sender, address(this), _amount);
        address[] memory compounts=despositInfos[msg.sender].compounts;
        (uint256 index,bool exist)=compounts.indexOf(_token);

        ...
    }
```

Solution

It is recommended to change the `despositInfos[msg.sender].compounts` to `despositInfos[to].compounts`.

Status

Fixed

[N26] [Low] Clear followers list element issue

Category: Gas Optimization Audit

Content

In the SignalSuscriptionModule contract, the manager of the jasperVault can can empty the list of followers directly by calling the unsubscribeByMaster function. But the delete function will only make the list element empty, and will not remove it from the list. So the element that has been cleared will still occupy a position in the list. This will result in additional gas consumption for follow trade operations.

Code location:

jasper-vaults-contracts/contracts/protocol/module/v1/SignalSuscriptionModule.sol#L181

```
    function unsubscribeByMaster(address target) external nonReentrant
onlyManagerAndValidSet(IJasperVault(target)) {
        delete followers[target];
    }
```

Solution

It is recommended to use the pop function to remove elements in the followers.

Status

Fixed

[N27] [Medium] Incorrect calculation of profit

Category: Design Logic Audit

Content

In the SignalSuscriptionModule contract, the current value of the jasperVault is recorded once when the user subscribes, and the difference between the before and after value is used to calculate the profit when the handlefee function is called to settle the fees. However, the calculation of the before and after value is inconsistent. The before value is the sum of all the values of the tokens of the contract positions obtained from the oracle machine (with masterToken as the price underlying), while the current value taken at settlement is the balance of masterToken in jasperVault taken directly. The inconsistency between the two leads to an incorrect way of calculating profits.

Code location: jasper-vaults-contracts/contracts/protocol/module/v1/SignalSuscriptionModule.sol

```
    function subscribe(
        IJasperVault _jasperVault,
        address target
    ) external nonReentrant onlyManagerAndValidSet(_jasperVault) {
        uint256 preBalance = controller
            .getSetValuer()
            .calculateSetTokenValuation(
                _jasperVault,
                _jasperVault.masterToken()
            );

        uint256 decimals=IERC20(_jasperVault.masterToken()).decimals();
        jasperVaultPreBalance[_jasperVault]
=preBalance.mul(10**decimals).preciseMul(_jasperVault.totalSupply()).div(PreciseUnitM
ath.preciseUnit());
        ...
    }

    ...

    function handleFee(
        IJasperVault _jasperVault
    ) external nonReentrant onlyManagerAndValidSet(_jasperVault) {
        address masterToken = _jasperVault.masterToken();
        uint256 preBalance = jasperVaultPreBalance[_jasperVault];

        ...

        uint256 nextBalance=IERC20(masterToken).balanceOf(address(_jasperVault));
        uint256 totalSupply = _jasperVault.totalSupply();
        uint256 followFee=followFees[address(_jasperVault)];
        uint256 strategistFeeBalance=preBalance.preciseMul(followFee);
        if (nextBalance > preBalance) {
            //Calculated profit
            uint256   fee=nextBalance-preBalance;

            ...

        }

        ...
}
```

Solution

It is recommended to unify the way of calculating the before and after value, either by unifying the way the prophecy machine calculates the total value of all components, or by unifying the use of the mastertoken's balance in the jasperVault.

Status

Acknowledged; The project team response: It is also most accurate to take the balance directly at the time of settlement when all positions have been cleared and converted to MasterToken.

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002305220002	SlowMist Security Team	2023.04.26 - 2023.05.22	Low 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 critical risk, 5 high risks, 8 medium risks, 7 low risks, and 6 suggestions. All the findings were fixed or acknowledged. The code has been deployed to the mainnet. The permissions of core roles such as the owner of the core contract have been transferred to the timelock contract.

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