FIRST LINE OF DEFENSE

javsphere

Audit

Security Assessment

Date: **01. Oktober 2024**

Client: **Javsphere**

Index

FIRST LINE OF DEFENSE

Introduction

line1.io is a distinguished brand under the officially registered entity, FutureVisions Deutschland, based in Germany. Our expertise lies in Blockchain Security, providing comprehensive services including Smart Contract Audits and KYC verification for project teams. At line1.io, we rigorously evaluate smart contracts for security vulnerabilities, verify the alignment between the codebase and related whitepapers/documentation, and deliver detailed recommendations for improvement.

Disclaimer

line1.io reports are neither an endorsement nor a disapproval of any specific project or team. These reports do not reflect the economic viability or value of any product or asset created by any team. line1.io does not test or audit the integration with external contracts or services.

line1.io audits do not offer any warranty or guarantee regarding the absolute absence of bugs in the analyzed technology, nor do they provide any information about the proprietors of the technology. These audits should not be used as a basis for making investment decisions or for involvement in any particular project. The reports do not constitute investment advice and should not be relied upon as such.

line1.io reports represent a comprehensive auditing process designed to assist our clients in enhancing the quality of their code while mitigating the significant risks associated with cryptographic tokens and blockchain technology. Given the high level of inherent risk in blockchain technology and cryptographic assets, line1.io emphasizes that each company and individual must conduct their own due diligence and maintain continuous security. line1.io does not claim any guarantee of the security or functionality of the technologies we analyze.

Project Overview

Overview

Project Name	**Javsphere**
Website	https://javsphere.com/
About the project	Javsphere offers the next generation of Decentralized Finance (DeFi) products. Our ecosystem is being built upon the foundation of DeFiChain, a blockchain-based on Bitcoin code that brings native DeFi capabilities to Bitcoin. On top of native DeFi, DeFi Meta Chain (DMC), was added as a functionality upgrade. DMC enables projects to build on the EVM-compatible layer quickly and deploy dApps and Smart Contracts (SC). Through the DMC, DeFiChain connects Bitcoin with Ethereum.
Chain	TBA
Language	SOLIDITY

Social Media Information

Telegram	https://t.me/javsphere_start
X / Twitter	https://x.com/javsphere
Discord	https://discord.gg/ssfmAn6D5h
YouTube	https://youtube.com/@javsphere

Audit Summary

Version	Date	Changelog
V1.0	14th August 2024	Initial Report
V1.1	25th September 2024	Reaudit
V1.2	1st October 2024	Small adjustments

Note: This audit report provides a detailed security analysis of the solidity codebase used in the project, particularly focusing on potential vulnerabilities to external malicious interference with the program's functions. This analysis did not cover functional testing (or unit testing) of the program's logic. Therefore, we cannot assure complete logical correctness of the code, as we did not perform functional tests on it. This includes the internal calculations in the algorithms implemented in the codebase.

Scope of Work

Line1 will conduct a thorough and comprehensive audit of the provided Solidity Smart Contracts, focusing on identifying vulnerabilities, ensuring code integrity, and verifying adherence to security best practices. This audit is designed to provide assurance that the smart contract is secure, efficient, and performs as intended.

The auditing process follows a structured and methodical approach:

- Pre-Audit ReviewWe begin by reviewing the specifications, source code, and any supplementary documentation provided to Line1. This ensures a clear understanding of the smart contract's scope, functionality, and underlying design, setting the foundation for an in-depth analysis.
- Manual Code ReviewOur team manually inspects the source code, examining each line in detail to identify potential vulnerabilities, such as logic flaws, access control issues, and susceptibility to known exploits (e.g., reentrancy, overflows). The goal is to uncover weaknesses that could be exploited in real-world scenarios.
- Specification ComparisonWe verify that the code aligns with the provided specifications and functional requirements. This comparison ensures that the smart contract behaves as expected and that the implementation matches the intended design.
- Test Coverage AnalysisWe evaluate the test coverage to ensure the testing framework thoroughly exercises the contract. This involves analyzing how much of the code is covered by existing tests and identifying gaps where additional test cases may be necessary to ensure robust functionality.
- Symbolic Execution and Automated ToolsWe use advanced automated tools, including symbolic execution, to simulate a wide range of inputs and execution paths, ensuring that every part of the contract behaves securely under different conditions. This helps identify edge cases and potential failure points that might not be evident through manual inspection alone.
- Best Practices ReviewLeveraging industry-standard guidelines, academic research, and established best practices, we evaluate the smart contract's architecture to ensure it is efficient, maintainable, and secure. Our review focuses on improving clarity, maintainability, and resilience against future threats.
- Actionable RecommendationsUpon completion, Line1 will provide a detailed audit report, including specific, itemized recommendations. These actionable insights will help secure the smart contract and guide any necessary improvements or optimizations.

Repository	Commit
https://github.com/Javsphere/contracts/tree/main/contracts	7f5717e
https://github.com/Javsphere/contracts	a40647a

Imported Packages

Used code from other Frameworks.

Dependency / Import Path	Count
@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol	1
@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC20/extensions/ERC20BurnableUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC20/extensions/ERC20CappedUpgradeable.sol	2
@openzeppelin/contracts-upgradeable/token/ERC721/extensions/ERC721URIStorageUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/utils/PausableUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/utils/ReentrancyGuardUpgradeable.sol	10
@openzeppelin/contracts/token/ERC20/IERC20.sol	11
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	19
@openzeppelin/contracts/token/ERC721/IERC721.sol	2
@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol	1
@openzeppelin/contracts/token/ERC721/extensions/IERC721Enumerable.sol	1
@openzeppelin/contracts/token/ERC721/extensions/IERC721Metadata.sol	1
@openzeppelin/contracts/utils/cryptography/ECDSA.sol	1
@openzeppelin/contracts/utils/cryptography/MessageHashUtils.sol	1
@openzeppelin/contracts/utils/math/Math.sol	1

Dependency / Import Path	Count
@openzeppelin/contracts/utils/structs/EnumerableSet.sol	7
@uniswap/v3-core/contracts/interfaces/callback/IUniswapV3SwapCallback.sol	1
@uniswap/v3-periphery/contracts/interfaces/IERC721Permit.sol	1
@uniswap/v3-periphery/contracts/interfaces/IPeripheryImmutableState.sol	1
@uniswap/v3-periphery/contracts/interfaces/IPeripheryPayments.sol	1
@uniswap/v3-periphery/contracts/interfaces/IPoolInitializer.sol	1
@uniswap/v3-periphery/contracts/interfaces/ISwapRouter.sol	2

Note for Investors: We have only audited the dependencies listed in the above scope. We have not reviewed any additional dependencies related to the project that are not included in our audit scope, and we cannot comment on their security. We are not responsible for any security issues arising from these unreviewed dependencies.

Audit Information

Vulnerability & Risk Management Level

Risk represents the probability that a certain source threat will exploit vulnerability and the impact of that event on the organization or system. The risk Level is computed based on CVSS version 3.0.

Level	Value	Description	Required Action
CRITICAL	9-10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
HIGH	7-8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon aspossible.
MEDIUM	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
LOW	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
INFORMATIONAL	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

Auditing Strategy and Techniques Applied

Throughout the audit of the Cosmos SDK Layer 1 blockchain repository, meticulous attention was dedicated to identifying security vulnerabilities, ensuring code quality, and verifying adherence to both specifications and best practices. Our audit was conducted by a seasoned team of penetration testers and blockchain developers with extensive experience in the Cosmos ecosystem and smart contract security.

Our team conducted a thorough line-by-line review of the code, ensuring that no detail was overlooked. Every identified issue was meticulously documented to provide a comprehensive overview of potential vulnerabilities and areas for improvement. Each file within the repository was subjected to a thorough manual examination, maintaining a high level of scrutiny and precision.

While automated tools were employed, their usage was strategically limited to augment the efficiency and effectiveness of the manual review process rather than replace it. This combination of rigorous manual inspection and strategic use of automated tools allows us to deliver an in-depth and reliable assessment, ensuring the highest standards of security and code quality for our clients.

Methodology

The auditing process follows a structured series of steps to ensure thorough examination and assessment:

- Specification Review: Our team meticulously reviewed all relevant documentation, specifications, and guidelines provided initially. This step ensures a deep understanding of the blockchain's architecture, scope, and intended functionalities.

- Manual Code Inspection: The source code was examined line by line in an intensive review aimed at uncovering potential security vulnerabilities that could be exploited maliciously. This careful inspection ensures no detail is overlooked.

- Specification Conformance: The code was rigorously compared against the provided specifications to confirm its fidelity in performing as described. This ensures that the implementation accurately reflects the intended design and requirements.

- Test Coverage Analysis: We analyzed the extent of test cases' coverage over the codebase. This involved determining how much code was executed during these tests to identify untested paths and ensure comprehensive test coverage.

- Symbolic Execution: This technique was applied to analyze how different inputs affect the code execution paths. It helped understand the conditions under which various parts of the program would execute, revealing potential vulnerabilities and ensuring robust security.

During this audit, certain aspects, such as unused constants, functions, exported functions, global variables, and parameters, were excluded from the scope of our review. While these elements were not directly evaluated, we suggest revisiting these areas in future audits to ensure that they do not introduce security concerns or affect the overall quality of the codebase.

Metrics

External/Public functions

External/public functions are functions that can be called from outside of a contract, i.e., they can be accessed by other contracts or external accounts on the blockchain. These functions are specified using the function declaration's external or public visibility modifier.

State variables

State variables are variables that are stored on the blockchain as part of the contract's state. They are declared at the contract level and can be accessed and modified by any function within the contract. State variables can be needed within visibility modifier, such as public, private or internal, which determines the access level of the variable.

Components

Contracts	Libraries	Interfaces	Abstract
32	23	48	4

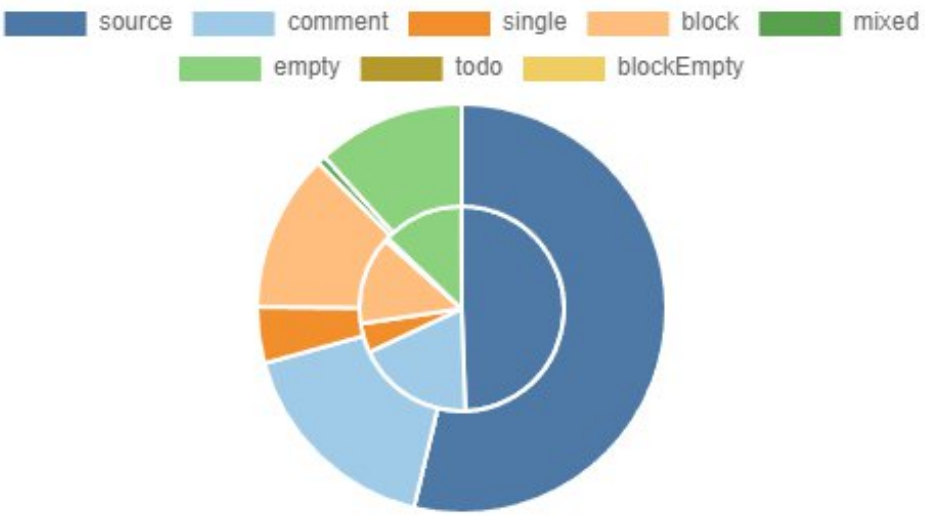
Exposed Functions

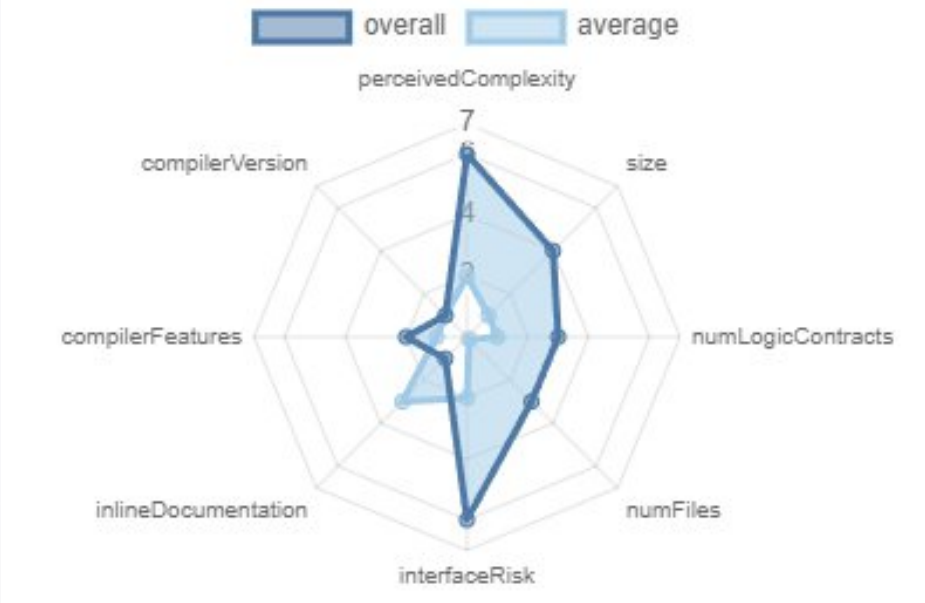
This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

Public	Payable	External	Internal	Private	Pure	View
726	38	681	829	58	69	440

Capabilities

Overall, the structure of the codebase, with its emphasis on thorough documentation and clear separation of code, comments, and blank lines, suggests a high level of professionalism and foresight in its development. This meticulous approach not only facilitates easier maintenance and updates but also ensures that the project can be effectively scaled and adapted as needed in the future.





Source Units in Scope

File	Lines	nLines	nSLOC	Comment Lines	Complex. Score
contracts\WJavToken.sol	58	54	24	23	25
contracts\trade\LLPToken.sol	43	43	25	8	21
contracts\trade\JavTradingStorage.sol	248	210	124	44	122
contracts\trade\JavTradingProcessing.sol	80	72	37	19	40
contracts\trade\JavTradingInteractions.sol	117	90	55	19	57
contracts\trade\JavReferrals.sol	176	157	92	34	89
contracts\trade\JavPriceImpact.sol	167	123	72	27	66
contracts\trade\JavPriceAggregator.sol	104	86	49	20	42
contracts\trade\JavPairsStorage.sol	200	180	102	40	108
contracts\trade\JavMultiCollatDiamond.sol	26	26	17	11	12
contracts\trade\JavFeeTiers.sol	111	86	49	20	41
contracts\trade\JavBorrowingProvider.sol	469	424	326	32	255
contracts\trade\JavBorrowingFees.sol	268	177	111	34	84
contracts\trade\abstract\JavDiamondStorage.sol	26	26	15	6	15
contracts\trade\abstract\JavDiamondLoupe.sol	58	56	30	20	31
contracts\trade\abstract\JavDiamondCut.sol	311	278	160	78	151
contracts\trade\abstract\JavAddressStore.sol	96	92	49	27	39
contracts\libraries\trade\updatePositionSize\UpdatePositionSizeUtils.sol	189	182	114	41	45

File	Lines	nLines	nSLOC	Comment	Complex.
contracts\libraries\trade\updatePositionSize\IncreasePositionSizeUtils.sol	326	309	215	66	91
contracts\libraries\trade\updatePositionSize\DecreasePositionSizeUtils.sol	197	186	118	49	54
contracts\libraries\trade\updateLeverage\UpdateLeverageUtils.sol	233	213	140	52	58
contracts-main\contracts\libraries\trade\TradingStorageUtils.sol	620	570	335	121	216
contracts\libraries\trade\TradingProcessingUtils.sol	751	679	479	122	184
contracts\libraries\trade\TradingInteractionsUtils.sol	444	388	246	83	152
contracts\libraries\trade\TradingCommonUtils.sol	932	796	490	236	283
contracts\libraries\trade\StorageUtils.sol	25	25	14	9	12
contracts\libraries\trade\ReferralsUtils.sol	427	407	223	102	123
contracts\libraries\trade\PriceImpactUtils.sol	617	520	308	143	189
contracts\libraries\trade\PriceAggregatorUtils.sol	204	185	110	52	62
contracts\libraries\trade\PairsStorageUtils.sol	494	459	237	147	177
contracts\libraries\trade\PackingUtils.sol	92	82	40	33	42
contracts\libraries\trade\FeeTiersUtils.sol	322	299	177	74	90
contracts\libraries\trade\DiamondUtils.sol	30	30	14	12	6
contracts\libraries\trade\ConstantsUtils.sol	49	43	29	17	16
contracts\libraries\trade\CollateralUtils.sol	33	31	13	13	6
contracts\libraries\trade\ChainUtils.sol	22	22	8	10	5
contracts\libraries\trade\BorrowingFeesUtils.sol	1054	892	569	219	273

File	Lines	nLines	nSLOC	Comment	Complex.
contracts\libraries\trade\ArrayGetters.sol	202	189	120	40	173
contracts-main\contracts\libraries\trade\AddressStoreUtils.sol	31	31	14	12	6
contracts\libraries\PriceUtils.sol	31	27	14	9	12
contracts\launch\TokenVestingFreezer.sol	388	347	235	72	108
contracts\launch\TokenVesting.sol	378	338	218	79	110
contracts\launch\CommunityLaunchETH.sol	112	108	72	12	55
contracts\launch\CommunityLaunch.sol	609	573	428	45	228
contracts\launch\Airdrop.sol	92	82	56	13	41
contracts\JavToken.sol	54	54	24	23	25
contracts\JavStakeX.sol	411	391	249	75	145
contracts\JavNetwork.sol	49	49	32	3	25
contracts\JavMarket.sol	255	239	159	41	104
contracts\JavFreezer.sol	697	647	452	107	234
contracts\JavFarming.sol	825	760	466	160	322
contracts\interfaces\trade\types\IUpdatePositionSize.sol	58	58	44	9	1
contracts\interfaces\trade\types\IUpdateLeverage.sol	24	24	14	7	1
contracts\interfaces\trade\types\ITypes.sol	32	32	24	4	21
contracts\interfaces\trade\types\ITradingStorage.sol	121	121	93	46	1
contracts\interfaces\trade\types\ITradingProcessing.sol	49	49	37	8	1
contracts-	18	18	8	5	1

File	Lines	nLines	nSLOC	Comment	Complex.
main\contracts\interfaces\trade\types\ITradingInteractions.sol					
contracts\interfaces\trade\types\IReferrals.sol	39	39	30	17	1
contracts\interfaces\trade\types\IPriceImpact.sol	53	53	39	14	1
contracts\interfaces\trade\types\IPriceAggregator.sol	29	29	15	7	1
contracts\interfaces\trade\types\IPairsStorage.sol	54	54	43	18	1
contracts\interfaces\trade\types\IFeeTiers.sol	32	32	22	12	1
contracts\interfaces\trade\types\IDiamondStorage.sol	46	46	28	14	1
contracts\interfaces\trade\types\IBorrowingFees.sol	104	104	87	36	1
contracts\interfaces\trade\types\IAddressStore.sol	29	29	19	10	1
contracts\interfaces\trade\libraries\IUpdatePositionSizeUtils.sol	62	62	31	26	3
contracts\interfaces\trade\libraries\IUpdateLeverageUtils.sol	35	35	17	15	3
contracts\interfaces\trade\libraries\ITradingStorageUtils.sol	394	17	4	230	71
contracts\interfaces\trade\libraries\ITradingProcessingUtils.sol	192	21	13	100	27
contracts\interfaces\trade\libraries\ITradingInteractionsUtils.sol	202	20	5	105	43
contracts\interfaces\trade\libraries\ITradingCommonUtils.sol	77	77	35	34	1
contracts\interfaces\trade\libraries\IReferralsUtils.sol	277	19	4	169	51
contracts-main\contracts\interfaces\trade\libraries\IPriceImpactUtils.sol	259	18	5	142	37
contracts\interfaces\trade\libraries\IPriceAggregatorUtils.sol	97	19	5	54	26
contracts\interfaces\trade\libraries\IPairsStorageUtils.sol	279	16	4	167	65

File	Lines	nLines	nSLOC	Comment	Complex.
contracts\interfaces\trade\libraries\IFeeTiersUtils.sol	164	19	4	92	23
contracts\interfaces\trade\libraries\IBorrowingFeesUtils.sol	447	18	4	220	51
contracts\interfaces\trade\IJavMultiCollatDiamond.sol	37	37	26	7	23
contracts\interfaces\trade\IJavDiamondLoupe.sol	42	24	7	22	9
contracts\interfaces\trade\IJavDiamondCut.sol	40	22	4	21	5
contracts\interfaces\trade\IJavDiamond.sol	17	17	7	5	9
contracts\interfaces\trade\IJavBorrowingProvider.sol	28	6	3	1	9
contracts\interfaces\trade\IJavAddressStore.sol	59	17	5	33	13
contracts\interfaces\trade\IGeneralErrors.sol	34	34	27	5	1
contracts\interfaces\libraries\IPriceUtils.sol	19	14	3	9	3
contracts\interfaces\IWDFI.sol	10	7	4	1	10
contracts\interfaces\IVanillaRouter02.sol	191	6	4	1	64
contracts\interfaces\IVanillaPair.sol	82	8	5	1	53
contracts\interfaces\ITokenVesting.sol	39	26	14	10	5
contracts-main\contracts\interfaces\ISwapRouter.sol	74	59	37	15	23
contracts\interfaces\IStateRelayer.sol	82	59	26	44	9
contracts\interfaces\IRewardsDistributor.sol	7	6	3	1	3
contracts\interfaces\INonfungiblePositionManager.sol	180	137	63	69	40
contracts\interfaces\IJavStakeX.sol	7	6	3	1	3
contracts\interfaces\IJavPriceAggregator.sol	34	17	9	15	12

File	Lines	nLines	nSLOC	Comment	Complex.
contracts\interfaces\IJavFreezer.sol	31	6	3	1	11
contracts\interfaces\IJavBank.sol	7	6	3	1	3
contracts\interfaces\IERC20Extended.sol	15	8	4	1	11
contracts\InfinityPass.sol	37	33	24	2	21
contracts\helpers\RewardsDistributor.sol	164	151	109	9	87
contracts\helpers\RewardsCollerctor.sol	59	59	42	3	31
contracts\helpers\RewardRateConfigurable.sol	75	66	48	1	16
contracts\helpers\LPProvider.sol	281	251	187	9	191
contracts\helpers\JavPriceAggregator.sol	144	135	91	15	86
contracts\DUSDStaking.sol	144	141	92	21	74
contracts\base\BaseUpgradable.sol	53	53	32	7	29
Totals	18918	14773	9274	4646	6159

Overall Security

Checks	Privileges
Upgradeability	➢ The contract contains the functionality in which the contract deployer can update and deploy the new version of the contract.
Ownership	➢ The ownership of the contract is not renounced.
Minting	➢ The owner is able to mint new tokens once the contract is deployed.
Fees	➢ The owner can set fees of more than 25%.
Locking	➢ The owner can lock the functions.

Centralization Privileges

File	Privileges
JavAddressStore	➢ The role manager can set roles in the contract.
JavDiamondCut	➢ The role manager can add, replace, or remove functions from the facets.
JavBorrowingFees	➢ The manager can set the borrowing pair parameters in the contract. ➢ The manager can update the borrowing group parameter in the contract.
JavBorrowingProvider	➢ The admin can add new tokens to the contract. ➢ The admin can initially buy the JLP tokens. ➢ The owner can update the PNL handler in the contract. ➢ The PNL handler in the contract can send assets to the receiver.
JavFeeTiers	➢ The governor can set the group indices and volume multiplier in the contract. ➢ The governor can add fee tiers in the contract.
JavLPToken	➢ The owner can update any arbitrary address as the borrowing provider address in the contract including zero or dead address. ➢ The borrowing provider can mint an unlimited amount of tokens to any arbitrary address.
JavPairsStorage	➢ The governor can add new pairs to the contract. ➢ The governor can update the existing pairs in the contract. ➢ The governor can add or update groups in the contract. ➢ The governor can add or update the pair's fees in the contract. ➢ The manager can update the maximum leverage value in the contract.

JavPriceAggregator	➢ The governor can update the collateral USD Price feed value in the contract. ➢ The admin can add or remove signers from the contract. ➢ The admin can update the price update fees in the contract.
LPProvider	➢ The admin can update any arbitrary address as the bot address in the contract. ➢ The admin can update the rewards distributor address in the contract. ➢ The admin can update any arbitrary address as the WDFI token address. ➢ The admin can swap tokens ETH to WDFI and WDFI to DFI tokens. ➢ The admin can withdraw tokens from the contract. ➢ The admins can withdraw DFI tokens from the contract.
RewardsCollerctor	➢ The admin can add or remove the addresses from the allowed list.
RewardsDistributor	➢ The admin can add or remove the addresses from the allowed list. ➢ The admin can update any arbitrary amount to burn and freezer percent in the contract. ➢ The admin can update any arbitrary value as the pool fees in the contract. ➢ The allowed addresses have the authority to distribute rewards from the contract.
Airdrop	➢ The admin can update the vesting address in the contract. ➢ The admin can airdrop the tokens to the recipient's wallet address.
CommunityLaunch	➢ The admin can update the vesting parameters in the contract. ➢ The admin can update the bot address. ➢ The admin can update the state relayer address. ➢ The admin can update the vesting address. ➢ The admin can update the DUSD, USDT, pair, freezer address in the contract. ➢ The admin can activate or inactivate the sale in the contract. ➢ The admin can set any arbitrary value as the start token and end token price in the contract. ➢ The admin can update the section number and token amount to sale value in the contract. ➢ The admin can update the token amount by type in the contract. ➢ The admin can update the max bag size in the contract including zero. ➢ The admin can update the token factor in the contract. ➢ The admin can withdraw tokens, DFI to the recipient address.
CommunityLaunchETH	➢ The admin can update the USDT address in the contract.

	➢ The admin can activate or inactivate the sale in the contract. ➢ The admin can update the number of available tokens in the contract. ➢ The admin can update the token per transaction value in the contract including zero. ➢ The admin can withdraw tokens to the recipient wallet in the contract.
TokenVesting	➢ The admin can add or remove the wallets from the allowed list. ➢ The allowed addresses can create a new vesting schedule single or in batches. ➢ The admin can revoke the vesting schedule for the given vesting schedule ID. ➢ The admin can withdraw the amount and send it to the recipient wallet. ➢ The admin can pause the releasing of the vested tokens from the contract for an indefinite period of time.
TokenVestingFreezer	➢ The admin can update the freezer address in the contract. ➢ The admin can add or remove the allowed addresses from the contract. ➢ The allowed addresses can create a new vesting schedule single or in batches. ➢ The admin can revoke the vesting schedule for the given vesting schedule ID. ➢ The admin can pause the releasing of the vested tokens from the contract for an indefinite period of time.
JavPriceImpact	➢ The governor can update the windows count to not more than 5 in the contract. ➢ The governor can update the price impact windows duration in between 10 minutes to 30 days in the contract. ➢ The manager can update the pair depths.
JavReferrals	➢ The governor can update the Ally fee, Start referrer fee, update referral open fees. ➢ The governor can update the referrer target volume in USD value in the contract. ➢ The governor can whitelist/un-whitelist allies in the contract. ➢ The governor can whitelist/un-whitelist referrers in the contract.
JavTradingInteractions	➢ Any arbitrary address can open and close trades for themself. ➢ Any arbitrary address can update the open orders for themself.
JavTradingProcessing	➢ The governor can update the vault closing fees. ➢ The governor can claim pending fees.
JavTradingStorage	➢ The governor can activate trading. ➢ The governor can add collaterals to the contract.

	➢ The governor can active/inactive the state of collaterals.
BaseUpgradable	➢ The admin can pause and un-pause the contract. ➢ The admin can set the admin address.
DUSDStaking	➢ The admin can update the bot address. ➢ The bot contract will be able to update the investment and withdraw the claimable amount for the users. ➢ The admin can pause the deposit and withdraw request functionality for an indefinite period of time.
InfinityPass	➢ The owner can mint the NFT to any wallets in the contract.
JavFarming	➢ The admin can add or remove the pairs from the contract. ➢ The admin can update the reward token address. ➢ The admin can update the WDFI and router address. ➢ The admin can update the reward configurations, which will be rewards per block and rewards update block interval value. ➢ The admin can update the last reward block value in the contract. ➢ The admin can add new LP pools in the contract. ➢ The admin can update the pool's allocation point.
JavFreezer	➢ The admin can update the lock period value in the contract. ➢ The admin can add pools to the contract. ➢ The admin can update the rewards info in the contract. ➢ The admin can update the reward configurations. ➢ The admin can update the vesting contract address. ➢ The admin can update the rewards distributor address. ➢ The admin can update the pool info, pool fees, lock period, lock period multiplier, and infinity pass percentage value in the contract.
JavMarket.sol	➢ The admin can update the treasury address. ➢ The admin can update the fees in the contract. ➢ The admin can add or remove the bot address. ➢ The admin can add or remove the token address from the contract.
JavNetwork	➢ The admin can update the bot address. ➢ The bot can save the consumer ID in the contract.
JavStakeX.sol	➢ The admin can update the reward distributor address and the Infinity pass contract address. ➢ The admin can update the infinity pass percent value in the contract.

	➢ The admin can update the rewards configurations in the contract. ➢ The admin can add pools to the contract. ➢ The admin can update the pool info in the contract. ➢ The admin can update the pool fees in the contract.
JavToken	➢ The owner can mint an unlimited number of tokens after initial deployment.
WJavToken	➢ The owner can mint an unlimited number of tokens after the initial deployment.

Audit Results

Critical Issues - [0]

No Critical Issues

High Issues - [0]

No High Issues

Medium Issues - [5]

#M-1 The owner can lock selling		
Severity	**Location / Line**	**Status**
Medium	JavBorrowingProvider.sol/L159-171	Open
Description	The contract contains the functionality in which the manager has the ability to pause or resume the sale of JLP tokens as needed to prevent any disruptions, ensuring that the tokens remain secure in the event of a functionality issue. The contract can have a locking period so that the functionality of the contract should not be locked for an indefinite period.	
Advisory	It is recommended that there must be a locking period so that there will not be any locking for an indefinite period of time.	

#M-2 Missing 'isContract' check		
Severity	**Location / Line**	**Status**
Medium	LPProvider.sol/L78-82	Open
Medium	Airdrop.sol/L34-38	Open
Medium	CommunityLaunch.sol/L156-196	Open
Medium	DUSDStaking.sol/L60-64	Open
Medium	JavFarming.sol/L121-146, L154-167	Open
Medium	JavFreezer.sol/L219-229, L273-277	Open
Medium	JavMarket.sol/L113-117, L125-129, L131-141	Open
Medium	JavNetwork.sol/L32-36	Open
Medium	JavStakeX.sol/L109-113, L121-125	Open
Description	The contract contains the functionality in which the admin of the contract can update any arbitrary address as the contract address, which is not recommended as this can cause functionality failure if the address is set to any arbitrary address. It is recommended that there must be a check so that the address can only be set to the contract address to avoid these circumstances	

Advisory	Add a 'require' check so that the address cannot be set to the contract address to avoid this issue in the contract.

#M-3 Missing Threshold		
Severity	**Location / Line**	**Status**
Medium	RewardsDistributor.sol/L95-100	Open
Description	The contract contains the functionality in which the admin of the contract can update any arbitrary amount as the burn and freezer percent in the contract including 100% which is not recommended as this can cause the lock of rewards for the user if the admin has set the values to 100%, No rewards will be distributed to the wallets from the contract.	
Advisory	It is recommended that there must be a check in which the burn and freezer percentage cannot be set more than 25% in the contract.	

#M-4 Missing zero check		
Severity	**Location / Line**	**Status**
Medium	CommunityLaunch.sol/L204-238	Open
Medium	CommunityLaunchETH.sol/L64-74	Open
Medium	JavFarming.sol/L148-152, L172-188	Open
Medium	JavFreezer.sol/L212-217, L231-243, L255-277	Open
Description	The contract contains the functionalities in which the admin of the contract can update any arbitrary value in the parameter including zero which is not recommended as this can cause the failure of the functionalities if the value is set to any incorrect number There must be check so that he value scan not be zero in the contract.	
Advisory	Add a 'require' check so that the value cannot be set to any arbitrary number, including zero.	

#M-5 Missing Threshold		
Severity	**Location / Line**	**Status**
Medium	JavStakeX.sol/L207-229	Open
Description	The contract contains the functionality in which the admin can pause the claim and un-stake functionality for an indefinite period of time, which can prevent the user from claiming the rewards and tokens from the contract.	

Low Issues - [3]

#L-1 Floating pragma solidity version		
Severity	**Location / Line**	**Status**
Low	All	Open
Description	Adding the constant version of solidity is recommended, as this prevents the unintentional deployment of a contract with an outdated compiler that contains unresolved bugs.	
Advisory	Check and add the constant version of solidity in the contract.	

#L-2 Missing zero or dead address check		
Severity	**Location / Line**	**Status**
Low	All	Open
Description	It is recommended to check that the address cannot be set to zero or dead address.	
Advisory	Add a require check that the address cannot be set to zero or dead address.	

#L-3 Remove math library		
Severity	**Location / Line**	**Status**
Low	All	Open
Description	The compiler version above 0.8.0 has the ability to control arithmetic overflow/underflow. It is recommended that the unwanted code be removed in order to avoid high gas fees.	
Advisory	Check and update the contract.	

Informational - [1]

#I-1 Natspec documentation missing		
Severity	**Location / Line**	**Status**
Informational	All	Open
Description	The documentation generated from the Natspec format typically includes information about the functions and variables in the contract, including descriptions of their functionality and usage, as well as details about their input and output parameters. It may also include other information about the contract, such as its overall purpose and intended use. It is recommended to comment on your code to describe the functionality of the code.	