



SOLIDProof

Bring trust into your projects

**Blockchain Security | Smart Contract Audits | KYC
Development | Marketing**

MADE IN GERMANY

MUX World

AUDIT

SECURITY ASSESSMENT

29. July, 2023

FOR



MUX



SolidProof_io



@solidproof_io

Introduction	3
Disclaimer	3
Project Overview	4
Summary	4
Social Medias	4
Audit Summary	5
File Overview	6
Imported packages	9
Components	10
Exposed Functions	10
Capabilities	11
Inheritance Graph	12
Audit Information	13
Vulnerability & Risk Level	13
Auditing Strategy and Techniques Applied	14
Methodology	14
Overall Security	15
Upgradeability	15
Ownership	16
Ownership Privileges	17
Minting tokens	17
Burning tokens	18
Blacklist addresses	19
Fees and Tax	20
Lock User Funds	21
Centralization Privileges	22
Audit Results	24

Introduction

[SolidProof.io](#) is a brand of the officially registered company MAKE Network GmbH, based in Germany. We're mainly focused on Blockchain Security such as Smart Contract Audits and KYC verification for project teams.

Solidproof.io assess potential security issues in the smart contracts implementations, review for potential inconsistencies between the code base and the whitepaper/documentation, and provide suggestions for improvement.

Disclaimer

[SolidProof.io](#) reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team. SolidProof.io do not cover testing or auditing the integration with external contract or services (such as Unicrypt, Uniswap, PancakeSwap etc'...)

SolidProof.io Audits do not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technology proprietors. SolidProof Audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

SolidProof.io Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology. Blockchain technology and cryptographic assets present a high level of ongoing risk. SolidProof's position is that each company and individual are responsible for their own due diligence and continuous security. SolidProof in no way claims any guarantee of the security or functionality of the technology we agree to analyze.

Project Overview

Summary

Project Name	MUX World
Website	https://mux.network/
About the project	The MUX Protocol Suite is a complex of protocols with features that will offer optimized trading cost, deep liquidity, a wide range of leverage options and diverse market options for traders.
Chain	Arbitrum
Language	Solidity
Codebase Link	https://github.com/mux-world/muxlp-tranches-protocol/tree/main/contracts
Commit	1e02893
Unit Tests	Provided

Social Medias

Telegram	N/A
Twitter	https://twitter.com/muxprotocol
Facebook	N/A
Instagram	N/A
Github	N/A
Reddit	N/A
Medium	N/A
Discord	N/A
Youtube	N/A
TikTok	N/A
LinkedIn	N/A



Audit Summary

Version	Delivery Date	Changelog
v1.0	29. July 2023	<ul style="list-style-type: none"> • Layout Project • Automated- /Manual-Security Testing • Summary

Note - The following audit report presents a comprehensive security analysis of the smart contract utilized in the project. This analysis did not include functional testing (or unit testing) of the contract/s logic. We cannot guarantee 100% logical correctness of the contract as it was not functionally tested by us.





File Overview

The project provided us with the files that should be tested in the security assessment. This audit covered the following files listed below with an SHA-1 Hash.

File Name	SHA-1 Hash
contracts/router/RouterRewardImp.sol	cc9c552cc5497db96e52d47e7b0148fc12d4c401
contracts/router/RouterSeniorImp.sol	91aa74e6d0fd839d369031d0d84c9a21b12f2dc2
contracts/router/TicketImp.sol	bebe4b6be1a88aad868bed72b3767b7e314b84c8
contracts/router/RouterImp.sol	8751eb6c52e1d621dba1a97ea46c0a850802b8a7
contracts/router/AdapterImp.sol	5d82e8c6ff91ed613fd531c510a01988709f1915
contracts/router/RouterConfig.sol	f8760168901698da3f89851d90e9b3f50f74f490
contracts/router/RouterStore.sol	00a77c2209e31d6fcfd03f3b18645ec2db25117c
contracts/router/UtilsImp.sol	165a2097e12728e6b8dc35cd7c87dc260f6e2945
contracts/router/RouterV1.sol	a63ac7d8a3f85694be83b1e4216d70746e562c71
contracts/router/RouterJuniorImp.sol	d3210e73eb8b471617caa7833468b3a402fd7689
contracts/router/Type.sol	3de308dc2610a55e61509e1fbc9f43ac2dc2076e
contracts/interfaces/IJuniorVault.sol	d26c3b039f020a94917bf54ba95a9f21ddfc9019
contracts/interfaces/ISeniorVault.sol	a68cdd5fdd11f58475495c667397aadedbc4b135
contracts/interfaces/IRewardDistributor.sol	aec5afaf8f5eaf18319458cd814da8969c1dd419
contracts/interfaces/IRewardController.sol	53f97926f9749a3c3929bd1b157c18945ea61cf0



contracts/interfaces/IConfigurable.sol	aaf1ac659b200e3b878f9f9bdae559d0d6592af9
contracts/interfaces/mux/IMuxVester.sol	b78633e6b61c110e1563e742390e4110f3958323
contracts/interfaces/mux/IMuxLiquidityPool.sol	ebf072221f1d457fad15cb4858ae9f909cced20b
contracts/interfaces/mux/IMuxOrderBook.sol	4a12c9c53d44fd8760b9c155a4432c242fb3aaca
contracts/interfaces/mux/IMuxLiquidityCallback.sol	92e70736310a53e784d0e91555bc1835081ca690
contracts/interfaces/mux/IMuxRewardRouter.sol	30c2495eb0dda655d9ac93eb92686503e882dcdb
contracts/reward/RewardDistributor.sol	7c40c4d6c23b00050e20231cef4e07c3e98abf5f
contracts/reward/RewardController.sol	b9ab1e808af5f4053074b18cc8aa7e29d1126f0a
contracts/common/Keys.sol	34dfaa499aadea9630bba32df59ee61c2ae578e9
contracts/libraries/LibAsset.sol	9129f1a0da9edd63c40e32579a36b3ed6c50ab5c
contracts/libraries/LibConfigSet.sol	dba97a666509d4cd0f7a8aee4c76151529056d03
contracts/libraries/LibUniswap.sol	b3043091a86f6d417adbd1ccd91232deb5e69abc
contracts/libraries/LibTypeCast.sol	a437252aa6d3c2d0f00e3a40865297d15d659ea1
contracts/libraries/LibReferenceOracle.sol	e08af7938e70488b5c5a21a2c702fc9f26812ee6
contracts/libraries/LibERC4626.sol	af738224cc39298491ec0649b655908eea89d055
contracts/junior/JuniorVaultImp.sol	551e54ebefd044c26c769db3f3292f399864135b
contracts/junior/JuniorVault.sol	1772ed6aff2c2e23926531db18377859ef443725
contracts/junior/JuniorVaultStore.sol	132ddc89fc28323383f6c089d5873f96429c5161



contracts/junior/JuniorConfig.sol	47efd51f101fe054e8894b0e1c4503070d8f2115
contracts/junior/StakeHelperImp.sol	74cb528936362b1e20af16230fd457d8542563c1
contracts/junior/Type.sol	7eb42fd247682092315cdf06f3851f8b8bc42a8e
contracts/senior/SeniorVaultStore.sol	58347eaf0485575d1d8d83d15b731b65075398b1
contracts/senior/SeniorVault.sol	3b7cb9d834808b58bac0045b464f5312cbcaa0c8
contracts/senior/SeniorVaultImp.sol	24b937c1274845815ef3d8f0ecf44263e124ba46
contracts/senior/SeniorConfig.sol	8cd1c5031ebd2cab893b3f6f7a8b43e713aed30c
contracts/senior/Type.sol	4a8744597acd0c5b68f70402422c3a6f140bcf5c

Please note: Files with a different hash value than in this table have been modified after the security check, either intentionally or unintentionally. A different hash value may (but need not) be an indication of a changed state or potential vulnerability that was not the subject of this scan.

Note for Investors: We only audited contracts mentioned in the scope above in this audit report. All contracts related to the project apart from that are not a part of the audit, and we cannot comment on its security and are not responsible for it in any way.

Imported packages

Used code from other Frameworks/Smart Contracts (direct imports).

Dependency / Import Path	Count
@openzeppelin/contracts-upgradeable/access/AccessControlEnumerableUpgradeable.sol	4
@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol	2
@openzeppelin/contracts-upgradeable/interfaces/IERC4626Upgradeable.sol	1
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol	7
@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol	6
@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol	14
@openzeppelin/contracts-upgradeable/token/ERC20/extensions/IERC20MetadataUpgradeable.sol	2
@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol	14
@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol	2
@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol	4
@uniswap/v3-periphery/contracts/interfaces/IQuoter.sol	1
@uniswap/v3-periphery/contracts/interfaces/ISwapRouter.sol	3

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 defined with a visibility modifier, such as public, private, or internal, which determines the access level of the variable.

Components

 Contracts	 Libraries	 Interfaces	 Abstract
11	16	13	0


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
240	1





External	Internal	Private	Pure	View
175	282	5	28	155

StateVariables

Total	 Public
47	18



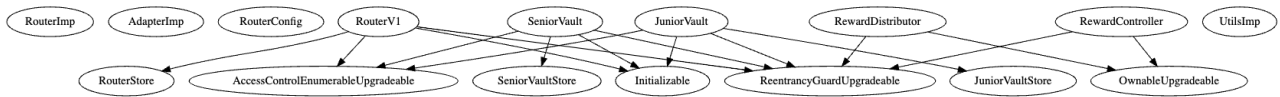
Capabilities

Solidity Versions observed	 Experimenta l Features	 Can Receive Funds	 Uses Assembl y	 Has Destroyable Contracts
0.8.17	-----	----	----	-----



Inheritance Graph

An inheritance graph is a graphical representation of the inheritance hierarchy among contracts. In object-oriented programming, inheritance is a mechanism that allows one class (or contract, in the case of Solidity) to inherit properties and methods from another class. It shows the relationships between different contracts and how they are related to each other through inheritance.



Audit Information

Vulnerability & Risk 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	Vulnerability	Risk (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 as possible.
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 review process, care was taken to check the repository for security-related issues, code quality, and compliance with specifications and best practices. To this end, our team of experienced pen-testers and smart contract developers reviewed the code line by line and documented any issues discovered.

We check every file manually. We use automated tools only so that they help us achieve faster and better results.

Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
 - a. Reviewing the specifications, sources, and instructions provided to SolidProof to ensure we understand the size, scope, and functionality of the smart contract.
 - b. Manual review of the code, i.e., reading the source code line by line to identify potential vulnerabilities.
 - c. Comparison to the specification, i.e., verifying that the code does what is described in the specifications, sources, and instructions provided to SolidProof.
2. Testing and automated analysis that includes the following:
 - a. Test coverage analysis determines whether test cases cover code and how much code is executed when those test cases are executed.
 - b. Symbolic execution, which is analysing a program to determine what inputs cause each part of a program to execute.
3. Review best practices, i.e., review smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on best practices, recommendations, and research from industry and academia.
4. Concrete, itemized and actionable recommendations to help you secure your smart contracts.

Overall Security

Upgradeability

Contract is an upgradeable

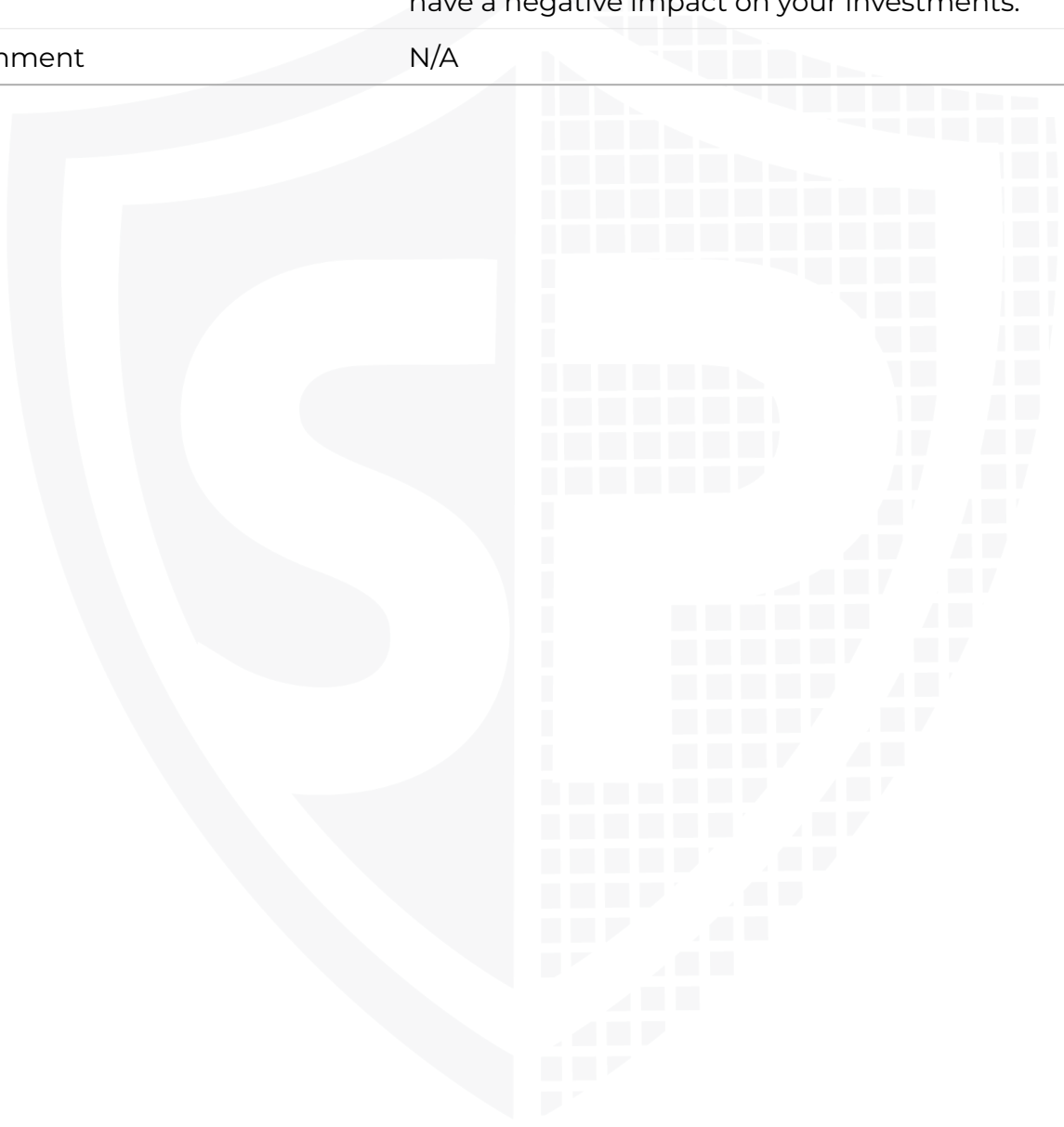
✗ Deployer can update the contract with new functionalities

Description

The deployer can replace the old contract with a new one with new features. Be aware of this, because the owner can add new features that may have a negative impact on your investments.

Comment

N/A



Ownership

The ownership is not renounced

✗ The owner is not renounce

Description

The owner has not renounced the ownership that means that the owner retains control over the contract's operations, including the ability to execute functions that may impact the contract's users or stakeholders. This can lead to several potential issues, including:

- Centralizations
- The owner has significant control over contract's operations

Comment

N/A

Note - If the contract is not deployed then we would consider the ownership to be not renounced. Moreover, if there are no ownership functionalities, ownership is automatically considered renounced.



Ownership Privileges

These functions can be dangerous. Please note that abuse can lead to financial loss. We have a guide where you can learn more about these Functions.

Minting tokens

Minting tokens refer to the process of creating new tokens in a cryptocurrency or blockchain network. This process is typically performed by the project's owner or designated authority, who has the ability to add new tokens to the network's total supply.

Contract owner cannot mint new tokens

 **The owner cannot mint new tokens**


Description	The owner is not able to mint new tokens once the contract is deployed.
-------------	---

Comment	N/A
---------	-----



Burning tokens

Burning tokens is the process of permanently destroying a certain number of tokens, reducing the total supply of a cryptocurrency or token. This is usually done to increase the value of the remaining tokens, as the reduced supply can create scarcity and potentially drive up demand.

Contract owner cannot burn tokens		 The owner cannot burn tokens
Description	The owner is not able burn tokens without any allowances.	
Comment	N/A	



Blacklist addresses

Blacklisting addresses in smart contracts is the process of adding a certain address to a blacklist, effectively preventing them from accessing or participating in certain functionalities or transactions within the contract. This can be useful in preventing fraudulent or malicious activities, such as hacking attempts or money laundering.

Contract owner cannot blacklist addresses



The owner cannot blacklist addresses

Description

The owner is not able blacklist addresses to lock funds.

Comment

N/A





Fees and Tax

In some smart contracts, the owner or creator of the contract can set fees for certain actions or operations within the contract. These fees can be used to cover the cost of running the contract, such as paying for gas fees or compensating the contract's owner for their time and effort in developing and maintaining the contract.

Contract owner cannot set fees more than 25%



The owner cannot levy unfair taxes

Description

The owner is not able to set the fees above 25%

Comment

N/A



Lock User Funds

In a smart contract, locking refers to the process of restricting access to certain tokens or assets for a specified period of time. When tokens or assets are locked in a smart contract, they cannot be transferred or used until the lock-up period has expired or certain conditions have been met.

Owner cannot lock the contract



The owner cannot lock the contract

Description

The owner is not able to lock the contract by any functions or updating any variables.

Comment

N/A

Centralization Privileges

Centralization can arise when one or more parties have privileged access or control over the contract's functionality, data, or decision-making. This can occur, for example, if the contract is controlled by a single entity or if certain participants have special permissions or abilities that others do not.

In the project, there are authorities that have access to the following functions:

File	Privileges
1. SeniorConfig.sol	<ul style="list-style-type: none"> ❖ OnlyAdmin <ul style="list-style-type: none"> - Set the Lock Type(soft or hard) - Set the Maximum borrows amount to a very small number like one and in that case the users can only borrow that amount and if they are trying to borrow again then it won't be possible - Set Penalty rate but not more than 1 - Set Lock Period to any arbitrary value - Set penalty recipient address - Set max borrows by vault
2. SeniorVault.sol	<ul style="list-style-type: none"> ❖ HANDLER_ROLE <ul style="list-style-type: none"> - Deposit and Withdraw assets from the vault(The caller may be owner or someone approved by the owner). - Borrow and Repay assets to the vault - Thus, the whole contract is centralized
3. RewardController.sol	<ul style="list-style-type: none"> ❖ onlyOwner <ul style="list-style-type: none"> - Set the Handler Address - Set Minimum and Maximum Stable APY, but it has to be less than or equal to one. Moreover, it could be set to zero also. - Set Initial time as the current timestamp - Set Senior Reward rate to any arbitrary value - Set Uniswap Contracts - Set Swap Paths
4. RewardDistributor.sol	<ul style="list-style-type: none"> ❖ onlyOwner <ul style="list-style-type: none"> - Set the Handler Address - The handler address can claim tokens for other addresses.
5. JuniorConfig.sol	<ul style="list-style-type: none"> ❖ OnlyAdmin <ul style="list-style-type: none"> - Set reward router address and Liquidity pool address

File	Privileges
6. JuniorVault.sol	❖ HANDLER_ROLE <ul style="list-style-type: none"> - Deposit and Withdraw assets from the vault(The caller may be owner or someone approved by the owner). - Transfer Assets In and Out from/to the router manually - Claim Rewards from staking cotnracts for any user - Adjust Vesting
7. RouterConfig.sol	❖ OnlyAdmin <ul style="list-style-type: none"> - Set reward Router and Liquidity pool addresses - Set Order Book Address - Set Target leverage to any arbitrary unsigned value except for 1 - Set rebalance threshold value - Set Liquidation Leverage
8. RouterV1.sol	❖ KEEPER_ROLE <ul style="list-style-type: none"> - Rebalance and Router router.

Recommendations

To avoid potential hacking risks, it is advisable for the client to manage the private key of the privileged account with care. Additionally, we recommend enhancing the security practices of centralized privileges or roles in the protocol through a decentralized mechanism or smart-contract-based accounts, such as multi-signature wallets.

Here are some suggestions of what the client can do:

- Consider using multi-signature wallets: Multi-signature wallets require multiple parties to sign off on a transaction before it can be executed, providing an extra layer of security e.g. Gnosis Safe
- Use of a timelock at least with a latency of e.g. 48-72 hours for awareness of privileged operations
- Introduce a DAO/Governance/Voting module to increase transparency and user involvement
- Consider Renouncing the ownership so that the owner cannot modify any state variables of the contract anymore. Make sure to set up everything before renouncing.

Audit Results

#1 | Approved Addresses can Claim for All Wallets

File	Severity	Location	Status
JuniorVault.sol	Medium	L140	Open

Description - The owner-approved HANDLER addresses can manually claim rewards for users in their account before time, this may lead to an issue where the users will not get the rewards out of their own free will.

#2 | Missing Timelock

File	Severity	Location	Status
RewardDistributor	Medium	L67	Open

Description - The contract misses a timelock in the claim function. This means that the claim function can be called recursively.

Remediation - We recommend putting a timelock so that the claim function cannot be called by an external contract recursively and only legitimate users will be able to claim.

#3 | Owner can Lock Borrow

File	Severity	Location	Status
SeniorConfig	Medium	L72	Open

Description - The admin address of the contract is able to set the max borrowing limit to a very small value which may lead to users not being able to borrow anymore that have already borrowed because, on the L58 of the "SeniorVaultImp" contract, the capacity variable will be zero.

Remediation - Make sure that it is not possible to set the value to a very small number instead try using a percentage of some value maximum value which may or may not fluctuate as per the users subjectively.

#4 | Logical Error

File	Severity	Location	Status
JuniorConfig	Medium	L27	Open

Description - DEFAULT_ADMIN must be set in the constructor explicitly

#5 | Missing Zero Address Validation

File	Severity	Location	Status
JuniorConfig	Low	L27	Open

Description

- Make sure to validate that the address passed in the function parameters is “non-zero”.

#6 | Missing Events

File	Severity	Location	Status
RewardController	Low	L60—83	Open

Description - Make sure to emit events for all the critical parameter changes in the contract to ensure the transparency and trackability of all the state variable changes in the contract.

#7 | NatSpec documentation missing

File	Severity	Location	Status
All	Informational	—	Open

Description

- If you started to comment on your code, also comment on all other functions, variables etc.



Recommendation for the Project Developers - As we have observed that the contracts are upgradeable in nature we strongly advise the team to make sure to disable the initializers in all the contracts once they are initialized otherwise there will be a risk of losing the ownership of the contracts.

Moreover, it has also come to our attention that some router contracts still have “TODO” comments and we would strongly advise the team to address them properly.

Legend for the Issue Status

Attribute or Symbol	Meaning
Open	The issue is not fixed by the project team.
Fixed	The issue is fixed by the project team.
Acknowledged(ACK)	The issue has been acknowledged or declared as part of business logic.







**Blockchain Security | Smart Contract Audits | KYC
Development | Marketing**

MADE IN GERMANY