InterFi NETWORK

SMART CONTRACT AUDIT

interfinetwork

hello@interfi.network

https://interfi.network

PREPARED FOR

MIRROR PROTOCOL STAKING CONTRACT

INTRODUCTION

Auditing Firm	InterFi Network
Client Firm	Mirror Protocol
Methodology	Automated Analysis, Manual Code Review
Language	Solidity
Implementation	0xf312Eb258470151A43bEf7bbBe43B65198e412BD
Proxy	0x1Fb81E728979cD4948bAf75955E7Dd36e0c49c69
Blockchain	Binance Smart Chain
Centralization	Active ownership with multi-sig approach
Commit	872d49d50ab7e82753f0113c2add65574eebbe26
Website	https://themirrorprotocol.com/
Report Date	April 18, 2023

ℹ️ Verify the authenticity of this report on our website: https://www.github.com/interfinetwork

EXECUTIVE SUMMARY

InterFi has performed the automated and manual analysis of solidity codes. Solidity codes were reviewed for common contract vulnerabilities and centralized exploits. Here's a quick audit summary:

Status	Critical 🔴	Major 🟠	Medium 🟡	Minor 🟢	Unknown 🟤
Open	0	0	1	2	0
Acknowledged	0	1	0	2	1
Resolved	1	0	0	0	0
Noteworthy Privileges	**Authorize Upgrade, Pause Contract,** Set Escrow Bonus Percentages, Set Rewards Distribution Duration, Set Last Time, Set Fees, Set Pools, Set Tokens				

ℹ️ Please note that smart contracts deployed on blockchains aren't resistant to exploits, vulnerabilities and/or hacks. Blockchain and cryptography assets utilize new and emerging technologies. These technologies present a high level of ongoing risks. For a detailed understanding of risk severity, source code vulnerability, and audit limitations, kindly review the audit report thoroughly.

ℹ️ Please note that centralization privileges regardless of their inherited risk status - constitute an elevated impact on smart contract safety and security.

TABLE OF CONTENTS

SCOPE OF WORK

InterFi was consulted by Mirror Protocol to conduct the smart contract audit of their solidity source codes. The audit scope of work is strictly limited to mentioned solidity file(s) only:

- MPStaking.sol

ℹ️ If source codes are not deployed on the main net, they can be modified or altered before main-net deployment. Verify the contract's deployment status below:

Public Contract Link	
https://bscscan.com/address/0xf312eb258470151a43bef7bbbe43b65198e412bd#code	
Contract Name	MPStaking
Compiler Version	0.8.12
License	MIT

AUDIT METHODOLOGY

Smart contract audits are conducted using a set of standards and procedures. Mutual collaboration is essential to performing an effective smart contract audit. Here's a brief overview of InterFi's auditing process and methodology:

CONNECT

- The onboarding team gathers source codes, and specifications to make sure we understand the size, and scope of the smart contract audit.

AUDIT

- Automated analysis is performed to identify common contract vulnerabilities. We may use the following third-party frameworks and dependencies to perform the automated analysis:
  - Remix IDE Developer Tool
  - Open Zeppelin Code Analyzer
  - SWC Vulnerabilities Registry
  - DEX Dependencies, e.g., Pancakeswap, Uniswap
- Simulations are performed to identify centralized exploits causing contract and/or trade locks.
- A manual line-by-line analysis is performed to identify contract issues and centralized privileges. We may inspect below mentioned common contract vulnerabilities, and centralized exploits:

Centralized Exploits	○ Token Supply Manipulation ○ Access Control and Authorization ○ Assets Manipulation ○ Ownership Control ○ Liquidity Access ○ Stop and Pause Trading ○ Ownable Library Verification

Common Contract Vulnerabilities	○ Integer Overflow ○ Lack of Arbitrary limits ○ Incorrect Inheritance Order ○ Typographical Errors ○ Requirement Violation ○ Gas Optimization ○ Coding Style Violations ○ Re-entrancy ○ Third-Party Dependencies ○ Potential Sandwich Attacks ○ Irrelevant Codes ○ Divide before multiply ○ Conformance to Solidity Naming Guides ○ Compiler Specific Warnings ○ Language Specific Warnings

REPORT

- The auditing team provides a preliminary report specifying all the checks which have been performed and the findings thereof.
- The client's development team reviews the report and makes amendments to solidity codes.
- The auditing team provides the final comprehensive report with open and unresolved issues.

PUBLISH

- The client may use the audit report internally or disclose it publicly.

ℹ️ It is important to note that there is no pass or fail in the audit, it is recommended to view the audit as an unbiased assessment of the safety of solidity codes.

RISK CATEGORIES

Smart contracts are generally designed to hold, approve, and transfer tokens. This makes them very tempting attack targets. A successful external attack may allow the external attacker to directly exploit. A successful centralization-related exploit may allow the privileged role to directly exploit. All risks which are identified in the audit report are categorized here for the reader to review:

Risk Type	Definition
Critical 🔴	These risks could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
Major 🟠	These risks are hard to exploit but very important to fix, they carry an elevated risk of smart contract manipulation, which can lead to high-risk severity.
Medium 🟡	These risks should be fixed, as they carry an inherent risk of future exploits, and hacks which may or may not impact the smart contract execution. Low-risk re-entrancy-related vulnerabilities should be fixed to deter exploits.
Minor 🟢	These risks do not pose a considerable risk to the contract or those who interact with it. They are code-style violations and deviations from standard practices. They should be highlighted and fixed nonetheless.
Unknown 🟤	These risks pose uncertain severity to the contract or those who interact with it. They should be fixed immediately to mitigate the risk uncertainty.

All statuses which are identified in the audit report are categorized here for the reader to review:

Status Type	Definition
Open	Risks are open.
Acknowledged	Risks are acknowledged, but not fixed.
Resolved	Risks are acknowledged and fixed.

CENTRALIZED PRIVILEGES

Centralization risk is the most common cause of cryptography asset loss. When a smart contract has a privileged role, the risk related to centralization is elevated.

There are some well-intended reasons have privileged roles, such as:

- Privileged roles can be granted the power to `pause()` the contract in case of an external attack.
- Privileged roles can use functions like, `include()`, and `exclude()` to add or remove wallets from fees, swap checks, and transaction limits. This is useful to run a presale and to list on an exchange.

Authorizing privileged roles to externally-owned-account (EOA) is dangerous. Lately, centralization-related losses are increasing in frequency and magnitude.

- The client can lower centralization-related risks by implementing below mentioned practices:
- Privileged role's private key must be carefully secured to avoid any potential hack.
- Privileged role should be shared by multi-signature (multi-sig) wallets.
- Authorized privilege can be locked in a contract, user voting, or community DAO can be introduced to unlock the privilege.
- Renouncing the contract ownership, and privileged roles.
- Remove functions with elevated centralization risk.

ℹ️ Understand the project's initial asset distribution. Assets in the liquidity pair should be locked. Assets outside the liquidity pair should be locked with a release schedule.

AUTOMATED ANALYSIS

Symbol	Definition
🛑	Function modifies state
💸	Function is payable
🔒	Function is internal
🔐	Function is private
!	Function is important

ANALYSIS_V1_0406

IERC20Upgradeable	Interface			
└	totalSupply	External !		NO!
└	balanceOf	External !		NO!
└	transfer	External !	🛑	NO!
└	allowance	External !		NO!
└	approve	External !	🛑	NO!
└	transferFrom	External !	🛑	NO!
PausableUpgradeable	Implementation	Initializable, ContextUpgradeable			
└	__Pausable_init	Internal 🔒	🛑	onlyInitializing
└	__Pausable_init_unchained	Internal 🔒	🛑	onlyInitializing
└	paused	Public !		NO!
└	_requireNotPaused	Internal 🔒		
└	_requirePaused	Internal 🔒		
└	_pause	Internal 🔒	🛑	whenNotPaused
└	_unpause	Internal 🔒	🛑	whenPaused

Initializable	Implementation			
└	_disableInitializers	Internal 🔒	🛑		
└	_getInitializedVersion	Internal 🔒			
└	_isInitializing	Internal 🔒			
OwnableUpgradeable	Implementation	Initializable, ContextUpgradeable		
└	__Ownable_init	Internal 🔒	🛑	onlyInitializing
└	__Ownable_init_unchained	Internal 🔒	🛑	onlyInitializing
└	owner	Public ❗		NO❗
└	_checkOwner	Internal 🔒			
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	_transferOwnership	Internal 🔒	🛑		
ReentrancyGuardUpgradeable	Implementation	Initializable		
└	__ReentrancyGuard_init	Internal 🔒	🛑	onlyInitializing
└	__ReentrancyGuard_init_unchained	Internal 🔒	🛑	onlyInitializing
└	_nonReentrantBefore	Private 🔐	🛑		
└	_nonReentrantAfter	Private 🔐	🛑		
└	_reentrancyGuardEntered	Internal 🔒			
UUPSUpgradeable	Implementation	Initializable, IERC1822ProxiableUpgradeable, ERC1967UpgradeUpgradeable		
└	__UUPSUpgradeable_init	Internal 🔒	🛑	onlyInitializing
└	__UUPSUpgradeable_init_unchained	Internal 🔒	🛑	onlyInitializing
└	proxiableUUID	External ❗		notDelegated
└	upgradeTo	Public ❗	🛑	onlyProxy
└	upgradeToAndCall	Public ❗	💵	onlyProxy

| └ | _authorizeUpgrade | Internal 🔒 | 🛑 | |

||||||

| **EnumerableSetUpgradeable** | Library | |||

| └ | _add | Private 🔐 | 🛑 | |

| └ | _remove | Private 🔐 | 🛑 | |

| └ | _contains | Private 🔐 | | |

| └ | _length | Private 🔐 | | |

| └ | _at | Private 🔐 | | |

| └ | _values | Private 🔐 | | |

| └ | add | Internal 🔒 | 🛑 | |

| └ | remove | Internal 🔒 | 🛑 | |

| └ | contains | Internal 🔒 | | |

| └ | length | Internal 🔒 | | |

| └ | at | Internal 🔒 | | |

| └ | values | Internal 🔒 | | |

| └ | add | Internal 🔒 | 🛑 | |

| └ | remove | Internal 🔒 | 🛑 | |

| └ | contains | Internal 🔒 | | |

| └ | length | Internal 🔒 | | |

| └ | at | Internal 🔒 | | |

| └ | values | Internal 🔒 | | |

| └ | add | Internal 🔒 | 🛑 | |

| └ | remove | Internal 🔒 | 🛑 | |

| └ | contains | Internal 🔒 | | |

| └ | length | Internal 🔒 | | |

| └ | at | Internal 🔒 | | |

| └ | values | Internal 🔒 | | |

||||||

IUniswapV2Router02	Interface	IUniswapV2Router01		
└	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
MPStaking	Implementation	IMPStaking, Initializable, KeeperCompatible, PausableUpgradeable, OwnableUpgradeable, ReentrancyGuardUpgradeable, UUPSUpgradeable		
└	<Constructor>	Public ❗	🛑	NO❗
└	initialize	External ❗	🛑	initializer
└	_authorizeUpgrade	Internal 🔒	🛑	onlyOwner
└	__MPStaking_init	Internal 🔒	🛑	
└	__MPStaking_init_unchained	Internal 🔒	🛑	
└	<Receive Ether>	External ❗	💵	NO❗
└	emergencyWithdrawBNB	External ❗	🛑	onlyOwner
└	emergencyWithdrawToken	External ❗	🛑	onlyOwner
└	setDistributeRewardsDuration	External ❗	🛑	onlyOwner
└	updateFees	External ❗	🛑	onlyOwner
└	setEscrowBonusPercentage	External ❗	🛑	onlyOwner
└	setEscrowPool	External ❗	🛑	onlyOwner
└	setRewardsPool	External ❗	🛑	onlyOwner
└	setStakingPool	External ❗	🛑	onlyOwner
└	updateRouter	External ❗	🛑	onlyOwner
└	updateOperationsWallet	External ❗	🛑	onlyOwner
└	updateLIOContract	External ❗	🛑	onlyOwner
└	addRewardToken	Public ❗	🛑	onlyOwner
└	addStakingToken	Public ❗	🛑	onlyOwner

| └ | removeRewardToken | External ❗ | 🛑 | onlyOwner |

| └ | removeStakingToken | External ❗ | 🛑 | onlyOwner |

| └ | updateStakingRewards | Private 🔐 | 🛑 | |

| └ | addBonusRewardsFromEscrow | Internal 🔒 | 🛑 | |

| └ | setLastTime | External ❗ | 🛑 | onlyOwner |

| └ | checkUpkeep | External ❗ | | NO❗ |

| └ | performUpkeep | External ❗ | 🛑 | NO❗ |

| └ | addNewStaker | Internal 🔒 | 🛑 | |

| └ | removeStaker | Internal 🔒 | 🛑 | |

| └ | reintroduceFee | Internal 🔒 | 🛑 | |

| └ | stakeToken | Public ❗ | 🛑 | NO❗ |

| └ | batchStakeTokens | External ❗ | 🛑 | whenNotPaused nonReentrant |

| └ | unStakeWithRewards | External ❗ | 🛑 | whenNotPaused nonReentrant |

| └ | updateStakerExclusions | Private 🔐 | 🛑 | |

| └ | _resetStakerExclusions | Private 🔐 | 🛑 | |

| └ | _calculateRewardsForStake | Private 🔐 | 🛑 | |

| └ | calculateRewardsForToken | Private 🔐 | | |

| └ | _calculatePercentageRewardsForToken | Private 🔐 | 🛑 | |

| └ | claimRewards | Internal 🔒 | 🛑 | |

| └ | withdrawRewards | External ❗ | 🛑 | whenNotPaused nonReentrant |

| └ | compoundRewards | External ❗ | 🛑 | whenNotPaused nonReentrant |

| └ | sendFees | Internal 🔒 | 🛑 | |

| └ | swapTokenToBNB | Internal 🔒 | 🛑 | |

| └ | getTotalStakedAmount | Public ❗ | | NO❗ |

| └ | getTokenRewardsOfUser | Public ❗ | | NO❗ |

| └ | getTokenRewardsOfUserWithDecimals | Public ❗ | | NO❗ |

| └ | getRemainingRewardsPool | Public ❗ | | NO❗ |

| └ | getAvailableRewardTokens | Public ❗ | | NO❗ |

| └ | getAvailableStakingTokens | Public ❗ | | NO❗ |

| └ | getBNBBalanceOfWallet | Public ❗ | | NO❗ |

| └ | getTokenBalanceOfStakingPool | Public ❗ | | NO❗ |

| └ | getTokenBalanceOfRewardsPool | Public ❗ | | NO❗ |

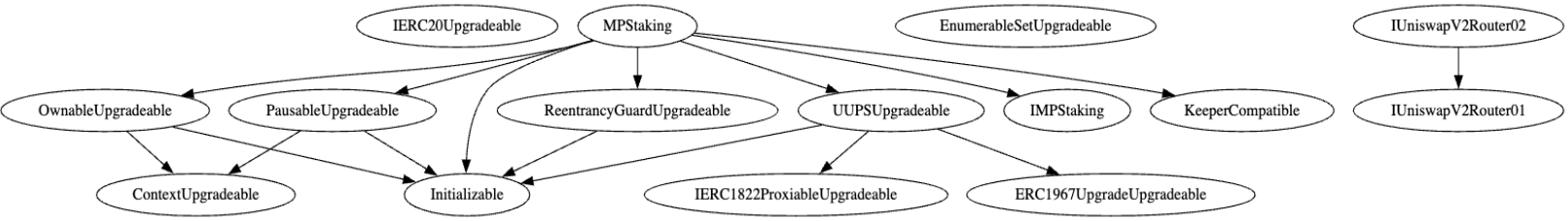
| └ | getStakedTokenAmount | External ❗ | | NO❗ |

| └ | pause | External ❗ | 🛑 | onlyOwner |

| └ | unpause | External ❗ | 🛑 | onlyOwner |

| └ | getTokenStakerCount | External ❗ | | NO❗ |

INHERITANCE GRAPH



MANUAL REVIEW

Identifier	Definition	Severity
CEN-01	Centralized privileges	Major 🟠
CEN-05	Privileged role using pause as a circuit breaker	

Important `onlyOwner` centralized privileges are listed below:

```
transferOwnership()
_authorizeUpgrade()
emergencyWithdrawBNB()
emergencyWithdrawToken()
setDistributeRewardsDuration()
updateFees()
setEscrowBonusPercentage()
setEscrowPool()
setRewardsPool()
setStakingPool()
updateRouter()
updateOperationsWallet()
updateLIOContract()
addRewardToken()
addStakingToken()
removeRewardToken()
removeStakingToken()
setLastTime()
pause()
unpause()
```

RECOMMENDATION

Contract creator, contract owner, administrator and all privileged roles' private keys should be secured carefully. Please refer to PAGE-09 CENTRALIZED PRIVILEGES for a detailed understanding.

ACKNOWLEDGEMENT

Mirror Protocol team acknowledged to use Gnosis multi-sig protocol to manage centralized privileges.

Identifier	Definition	Severity
CEN-09	Use of proxy and upgradeable contracts	Critical 🔴

Privileged role can initiate contract implementation. Contract upgradeability allows privileged roles to change current contract implementation.

```
function _authorizeUpgrade (address newImplementation) internal override onlyOwner {}
```


RECOMMENDATION

Test and validate current contract thoroughly before deployment. Future contract upgradeability negatively elevates centralization risk.

RESOLUTION

Mirror Protocol team has set contract upgradeability in place to deter future exploitability, and streamline current contracts with their future contracts.

Identifier	Definition	Severity
MST-01	Re-entrancy	Medium 🟡

Below mentioned functions are used with `nonReentrant` modifier to protect against re-entrancy:

```
batchStakeTokens()
unStakeWithRewards()
withdrawRewards()
compoundRewards()
```

In `stakeToken(), external` call is made using `IERC20Upgradeable(_token).transferFrom(msg.sender, address(stakingPool), _stakeAmount),` which transfers the staking tokens from the user's address to the staking pool. Before this external call, the function updates the user's staking information by either adding a new staker or updating the existing staker's information. The user's staking information is updated in `addNewStaker` or `updateStakerExclusions` functions. This ensures that the state is updated before any `external` call is made, making it resistant to re-entrancy attacks. However, re-entrancy attacks can happen in unusual and novel ways, it is strongly recommended to add `nonReentrant` modifier to this function.

RECOMMENDATION

Add `nonReentrant` modifier to `stakeToken().`

Identifier	Definition	Severity
LOG-01	Lack of appropriate arbitrary boundaries	Minor

Below mentioned functions are set with high arbitrary boundaries:

`updateFees`
`setEscrowBonusPercentage`

RECOMMENDATION

These functions should be provided appropriate upper and lower boundaries.

Identifier	Definition	Severity
LOG-02	Front-running	Minor

`swapTokenToBNB` swaps `_token` for ETH using the Uniswap V2 router. Swap function called, `swapExactTokensForETHSupportingFeeOnTransferTokens`, is designed to handle tokens with a transfer fee (or tax) mechanism. The front-running risk exists because an attacker can observe the pending transactions in the mempool and submit a transaction with a higher gas price to have their transaction executed before the original one.

RECOMMENDATION

`swapTokenToBNB` functions should be provided reasonable minimum output amounts, instead of zero. Introduce commit reveal scheme to mitigate front-running. Keep in mind, front-running is unavoidable on public blockchains, and each solution comes with a trade-off.

Identifier	Definition	Severity
COD-02	Possible timestamp manipulation via `block.timestamp`	Minor 🟢

Timestamp of a block can be manipulated by a miner to an extent. Below mentioned functions use `block.timestamp`:

`checkUpkeep()` and `performUpkeep()` uses timestamp to check if the contract needs up-keep or not. Purpose is to distribute rewards and update staking rewards at regular intervals.

`stakeToken()` and `compoundRewards()` uses timestamp to set the `startTS` field in the `StakeInfo` struct when a user stakes or compounds rewards.

RECOMMENDATION

To maintain block integrity, follow 15 seconds rule, and scale time dependent events accordingly.

Identifier	Definition	Severity
COD-10	Contract dependencies	Unknown

Smart contract is interacting with protocols e.g., Market Makers, Decentralized Applications, Open Zeppelin, MP staking and Escrow interfaces, Chainlink Keeper tools. The scope of the audit treats these entities as black boxes and assumes their functional correctness. However, in the real world, they can be compromised, and exploited. Moreover, upgrades in these entities can create severe impacts, e.g., increased transactional fees, deprecation of previous routers, etc.

RECOMMENDATION

Inspect dependencies regularly, and mitigate severe impacts whenever necessary.

ACKNOWLEDGEMENT

Mirror Protocol team will inspect these dependencies to minimize downtime from external intervention.

Identifier	Definition	Severity
COM-01	Floating compiler status	

Compiler is set to `^0.8.9`

RECOMMENDATION

Pragma should be fixed to the version that you're indenting to deploy your contracts with.

RESOLUTION

Smart contract is deployed with stable compiler version.

Identifier	Definition	Severity
COM-04	Potential resource exhaustion errors	Minor

Mentioned loops may throw out of gas errors upon executing:

`currentStakingTokens`
`currentRewardsTokens`
`stakingTokenAddresses`
`rewardsTokenAddresses`
`stakeTokenAddress`

In `updateStakingRewards()`, expression `(totalNewRewards[j] * accuracyFactor / stakingTokenCount / totalStake[i])` is calculated multiple times. You can calculate it once and store it in a variable before the loop, which will save gas.

In `updateStakingRewards()`, nested loop iterates through `stakingTokenAddresses` and `rewardsTokenAddresses`. You can reduce the gas cost by breaking the nested loop into two separate

RECOMMENDATION

Optimize contract to save transaction related costs.

DISCLAIMERS

InterFi Network provides the easy-to-understand audit of solidity source codes (commonly known as smart contracts).

The smart contract for this particular audit was analyzed for common contract vulnerabilities, and centralization exploits. This audit report makes no statements or warranties on the security of the code. This audit report does not provide any warranty or guarantee regarding the absolute bug-free nature of the smart contract analyzed, nor do they provide any indication of the client's business, business model or legal compliance. This audit report does not extend to the compiler layer, any other areas beyond the programming language, or other programming aspects that could present security risks. Cryptographic tokens are emergent technologies, they carry high levels of technical risks and uncertainty. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. This audit report could include false positives, false negatives, and other unpredictable results.

CONFIDENTIALITY

This report is subject to the terms and conditions (including without limitations, description of services, confidentiality, disclaimer and limitation of liability) outlined in the scope of the audit provided to the client. This report should not be transmitted, disclosed, referred to, or relied upon by any individual for any purpose without InterFi Network's prior written consent.

NO FINANCIAL ADVICE

This audit report does not indicate the endorsement of any particular project or team, nor guarantees its security. No third party should rely on the reports in any way, including to make any decisions to buy or sell a product, service or any other asset. The information provided in this report does not constitute investment advice, financial advice, trading advice, or any other sort of advice and you should not treat any of the report's content as such. This audit report should not be used in any way

to make decisions around investment or involvement. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort.

FOR AVOIDANCE OF DOUBT, SERVICES, INCLUDING ANY ASSOCIATED AUDIT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

TECHNICAL DISCLAIMER

ALL SERVICES, AUDIT REPORTS, SMART CONTRACT AUDITS, OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, INTERFI NETWORK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO SERVICES, AUDIT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, INTERFI NETWORK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM THE COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

WITHOUT LIMITING THE FOREGOING, INTERFI NETWORK MAKES NO WARRANTY OF ANY KIND THAT ALL SERVICES, AUDIT REPORTS, SMART CONTRACT AUDITS, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET THE CLIENT'S OR ANY OTHER INDIVIDUAL'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE.

TIMELINESS OF CONTENT

The content contained in this audit report is subject to change without any prior notice. InterFi Network does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following the publication.

LINKS TO OTHER WEBSITES

This audit report provides, through hypertext or other computer links, access to websites and social accounts operated by individuals other than InterFi Network. Such hyperlinks are provided for your reference and convenience only and are the exclusive responsibility of such websites' and social accounts' owners. You agree that InterFi Network is not responsible for the content or operation of such websites and social accounts and that InterFi Network shall have no liability to you or any other person or entity for the use of third-party websites and social accounts. You are solely responsible for determining the extent to which you may use any content at any other websites and social accounts to which you link from the report.

ABOUT INTERFI NETWORK

InterFi Network provides intelligent blockchain solutions. We provide solidity development, testing, and auditing services. We have developed 150+ solidity codes, audited 1000+ smart contracts, and analyzed 500,000+ code lines. We have worked on major public blockchains e.g., Ethereum, Binance, Cronos, Doge, Polygon, Avalanche, Metis, Fantom, Bitcoin Cash, Velas, Oasis, etc.

InterFi Network is built by engineers, developers, UI experts, and blockchain enthusiasts. Our team currently consists of 4 core members, and 6+ casual contributors.

Website: https://interfi.network

Email: hello@interfi.network

GitHub: https://github.com/interfinetwork

Telegram (Engineering): https://t.me/interfiaudits

Telegram (Onboarding): https://t.me/interfisupport

interfinetwork

hello@interfi.network

https://interfi.network

SMART CONTRACT AUDITS | SOLIDITY DEVELOPMENT AND TESTING

RELENTLESSLY SECURING PUBLIC AND PRIVATE BLOCKCHAINS