InterFi NETWORK

SMART CONTRACT AUDIT

interfinetwork

hello@interfi.network

https://interfi.network

PREPARED FOR

BBTF

(ESCROW & STAKING CONTRACTS)

INTRODUCTION

Auditing Firm	InterFi Network
Client Firm	BBTF
Methodology	Automated Analysis, Manual Code Review
Language	Solidity
Staking Pool	0xB92F3e56489320E44De78c78592068C72425B673
Rewards Pool	0xdb613E0dCD76DcA6a7c907a2B7E7EC60A20f7D9f
Escrow Pool	0x53241a8328DF464ed6f37848e23F42cb76E319d3
Staking Contract	0xaC3d632eB0501B7Fb3945c1c4864B218dd796069
Blockchain	Binance Smart Chain
Centralization	Active ownership
Commit	04e98d76955890e75c685e8fe167434da311336a
Website	https://bbtftoken.com/
Report Date	August 10, 2023

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	0	7	0
Acknowledged	0	1	0	0	1
Resolved	1	0	0	0	0
Escrow Noteworthy Privileges	**Emergency Withdraw Token and BNB, Set Control Contract,** Transfer Token To, Transfer Multi Token with Percentage				
Staking Noteworthy Privileges	**Authorize Upgrade, Pause Contract, Set Parent Token, Emergency Withdraw Token and BNB,** Set Escrow Bonus Percentages, Set Rewards Distribution Duration, Set Fees, Set Pools				

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 MultiRewardStaking 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:

- IEscrow.sol
- Escrow.sol
- MultiRewardStaking.sol
- IMultiRewardStaking.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/0xB92F3e56489320E44De78c78592068C72425B673#code	
Contract Name	Escrow.sol (Staking Pool)

Public Contract Link	
https://bscscan.com/address/0xdb613E0dCD76DcA6a7c907a2B7E7EC60A20f7D9f#code	
Contract Name	Escrow.sol (Rewards Pool)

Public Contract Link	
https://bscscan.com/address/0x53241a8328DF464ed6f37848e23F42cb76E319d3#code	
Contract Name	Escrow.sol (Escrow Pool)

Public Contract Link	
https://bscscan.com/address/0x6574b96181a56924c1d1879a9bbe330ecc77fa7f#code	
Contract Name	MultiRewardStaking.sol

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

Escrow

| **IEscrow** | Interface | |||

| └ | emergencyWithdrawBNB | External ! | 🛑 |NO! |

| └ | emergencyWithdrawToken | External ! | 🛑 |NO! |

| └ | transferMultiTokensToWithPercentage | External ! | 🛑 |NO! |

| └ | transferTokenTo | External ! | 🛑 |NO! |

||||||

| **Escrow** | Implementation | IEscrow, Ownable |||

| └ | <Constructor> | Public ! | 🛑 |NO! |

| └ | emergencyWithdrawBNB | External ! | 🛑 | onlyOwner |

| └ | emergencyWithdrawToken | External ! | 🛑 | onlyOwner |

| └ | transferMultiTokensToWithPercentage | External ! | 🛑 | onlyControlContract |

| └ | transferTokenTo | External ! | 🛑 | onlyControlContract |

| └ | setControlContract | External ! | 🛑 | onlyOwner |

MultiRewardStaking

| **IMultiRewardStaking** | Interface | |||

| └ | emergencyWithdrawBNB | External ! | 🛑 |NO! |

| └ | emergencyWithdrawToken | External ❗ | 🛑 |NO❗ |

| └ | addStakingToken | External ❗ | 🛑 |NO❗ |

| └ | removeStakingToken | External ❗ | 🛑 |NO❗ |

| └ | addRewardToken | External ❗ | 🛑 |NO❗ |

| └ | removeRewardToken | External ❗ | 🛑 |NO❗ |

| └ | batchStakeTokens | External ❗ | 🛑 |NO❗ |

| └ | unStakeWithRewards | External ❗ | 🛑 |NO❗ |

| └ | withdrawRewards | External ❗ | 🛑 |NO❗ |

| └ | compoundRewards | External ❗ | 🛑 |NO❗ |

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

| └ | unstakeTokensWithRewardsFor | External ❗ | 🛑 |NO❗ |

||||||

| **MultiRewardStaking** | Implementation | IMultiRewardStaking, Initializable, PausableUpgradeable, OwnableUpgradeable, ReentrancyGuardUpgradeable, UUPSAccessControlUpgradeable |||

| └ | <Constructor> | Public ❗ | 🛑 |NO❗ |

| └ | initialize | External ❗ | 🛑 | initializer |

| └ | __MultiRewardStaking_init | Internal 🔒 | 🛑 | onlyInitializing |

| └ | __MultiRewardStaking_init_unchained | Internal 🔒 | 🛑 | onlyInitializing |

| └ | <Receive Ether> | External ❗ | 💵 |NO❗ |

| └ | setParentToken | External ❗ | 🛑 | onlyOwner |

| └ | 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 | Internal 🔒 | 🛑 | |

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

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

| └ | updateStakerExclusions | Private 🔐 | 🛑 | |

| └ | _resetStakerExclusions | Private 🔐 | 🛑 | |

| └ | _calculateRewardsForStake | Private 🔐 | 🛑 | |

| └ | calculateRewardsForToken | Private 🔐 | | |

| └ | _calculatePercentageRewardsForToken | Private 🔐 | 🛑 | |

| └ | claimRewards | Internal 🔒 | 🛑 | |

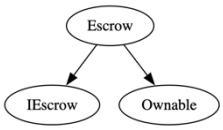
| └ | withdrawRewards | External ❗ | 🛑 | whenNotPaused nonReentrant |

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

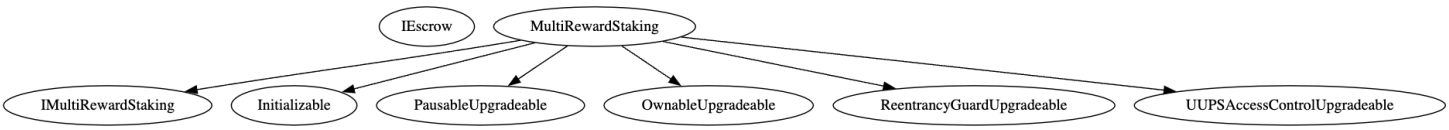
└	sendFees	Internal 🔒	🛑	
└	swapTokenToBNB	Internal 🔒	🛑	
└	getTotalStakedAmount	Public ❗		NO❗
└	getTokenRewardsOfUser	Public ❗		NO❗
└	getTokenRewardsOfUserWithDecimals	Public ❗		NO❗
└	getRemainingRewardsPool	Public ❗		NO❗
└	getAvailableRewardTokens	Public ❗		NO❗
└	getAvailableStakingTokens	Public ❗		NO❗
└	getBNBBalanceOfWallet	Public ❗		NO❗
└	getTokenBalanceOfStakingPool	Public ❗		NO❗
└	getTokenBalanceOfRewardsPool	Public ❗		NO❗
└	getStakedTokenAmount	External ❗		NO❗
└	pause	External ❗	🛑	onlyOwner
└	unpause	External ❗	🛑	onlyOwner
└	getTokenStakerCount	External ❗		NO❗
└	unstakeTokensWithRewardsFor	External ❗	🛑	nonReentrant onlyToken

INHERITANCE GRAPH

Escrow



MultiRewardStaking



MANUAL REVIEW

Identifier	Definition	Severity
CEN-01	Centralized privileges	Major 🟠
CEN-07	Authorizations and access controls	
MRS-01	Privileged roles pause staking feature in **`MultiRewardStaking`**	
MRS-02	Privileged roles withdraw contract balance	

`Escrow`

`onlyOwner` centralized privileges are applied to:

`emergencyWithdrawBNB`
`emergencyWithdrawToken`
`setControlContract`

`onlyControlContract` access control is provided to:

`transferMultiTokensToWithPercentage`
`transferTokenTo`

`MultiRewardStaking`

`onlyOwner` centralized privileges are applied to:

`setParentToken`
`emergencyWithdrawBNB`
`emergencyWithdrawToken`
`setDistributeRewardsDuration`
`updateFees`
`setEscrowBonusPercentage`
`setEscrowPool`
`setRewardsPool`
`setStakingPool`
`updateRouter`

```
updateOperationsWallet
updateLIOContract
addRewardToken
addStakingToken
removeRewardToken
removeStakingToken
setLastTime
pause
unpause
```

In **Escrow** and **MultiRewardStaking**, `emergencyWithdrawBNB()` and `emergencyWithdrawToken()` functions are callable by the contract owner. Make sure these functions are called only when absolutely required, and aren't maliciously used to withdraw contract balance.

RECOMMENDATION

Deployers, contract owners, administrators, control contracts, restricted, and all other privileged roles' private-keys/access-keys/admin-keys should be secured carefully. These entities can have a single point of failure that compromises the security of the project. Manage centralized and privileged roles carefully, review PAGE 09 for more information.

ACKNOWLEDGEMENT

Project team acknowledged to use Gnosis multi-sig protocol to manage centralized privileges and argued that centralization and access-controlled privileges are used as required.

Identifier	Definition	Severity
CEN-09	Use of proxy and upgradeable contracts in **`MultiRewardStaking`**	Critical 🔴

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

```
contract MultiRewardStaking is
    IMultiRewardStaking,
    Initializable,
    PausableUpgradeable,
    OwnableUpgradeable,
    ReentrancyGuardUpgradeable,
    UUPSAccessControlUpgradeable
{
```

Project team has added `_disableInitializers` in upgradeable implementation to add a safety measure that prevents initializer functions from being called more than once, reducing the risk of unintended behavior or vulnerabilities.


```
_authorizeUpgrade()
```

RECOMMENDATION

Test and validate current contract thoroughly before deployment. While proxy contracts are great for robust deployments while maintaining the upgradeable flexibility, *proxy codes are prone to new security or logical issues that will compromise the project.*

RESOLUTION

Project team iterated that contract uses proxy mechanism to have future contract upgradeability, and contract flexibility.

Identifier	Definition	
MRS-03	Re-entrancy in **`MultiRewardStaking`**	

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

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

Identifier	Definition	Severity
LOG-01	Lack of appropriate arbitrary boundaries in **`MultiRewardStaking`**	Minor 🟢

Below mentioned functions are set with high arbitrary boundaries:

`setDistributeRewardsDuration`
`updateFees`
`setEscrowBonusPercentage`

RECOMMENDATION

These functions should be provided appropriate upper and lower boundaries.

Identifier	Definition	Severity
LOG-02	Front-running in **`MultiRewardStaking`**	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
LOG-04	Decimal issue in percentage calculation in **`MultiRewardStaking`**	Minor 🟢

In `transferMultiTokensToWithPercentage`, calculation `tokenBalance * _percentage / _denominator` might lead to a rounding error due to the way Solidity handles division.

RECOMMENDATION

Alter the equation to: `(tokenBalance * _percentage) / _denominator`.

Identifier	Definition	Severity
COD-01	Unchecked return values	Minor

Smart contracts use external calls (transfers) to move tokens, but it doesn't check their return values.

RECOMMENDATION

Always check return value of transfer.

Identifier	Definition	
COD-02	Upkeep mechanism in **`MultiRewardStaking`**	

Smart contract relies on external callers to trigger the `performUpkeep` function to update staking rewards. If it's not called frequently, rewards won't get updated.

RECOMMENDATION

Use automated system to trigger the upkeep and update rewards.

Identifier	Definition	
COD-04	Potential flash loan vulnerabilities	

`MultiRewardStaking` and **`Escrow`** contracts do not appear to have direct flash loan vulnerabilities. Flash loan attacks typically exploit functions that rely on external data, e.g., oracles or functions that can be gamed within a single transaction. Smart contracts are making plentiful external calls.

NOTE

Due to the interconnected nature of DeFi contracts, one contract's vulnerability can be exploited using another contract. Be cautious while interacting with third-party contracts.

Identifier	Definition	Severity
COD-05	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	Direct and indirect dependencies	Unknown

Smart contract is interacting with third party protocols e.g., Market Makers, Control Contract, External Contracts and Interfaces, Web 3 Applications, Open Zeppelin tools. The scope of the audit treats these entities as black boxes and assumes their functional correctness. However, in the real world, all of them can be compromised, and exploited. Moreover, upgrades in these entities can create severe impacts, e.g., increased transactional fees, deprecation of previous routers, etc.

It's crucial to be aware that the control contract (or any entity with control of it) has a high degree of power over the assets in the **`Escrow`**. Make sure the control contract is secure.

RECOMMENDATION

Inspect third party dependencies regularly, and mitigate severe impacts whenever necessary.

ACKNOWLEDGEMENT

Project team will inspect third party dependencies to minimize downtime from third-party intervention. Since **`MultiRewardStaking`** contract is upgradeable, project team can push updates when required.

Identifier	Definition	Severity
COD-12	Lack of event-driven architecture	Minor

Smart contracts use function calls to update state, which can make it difficult to track and analyze changes to the contract over time.

RECOMMENDATION

Use events to track state changes. Events improve transparency and provide a more granular view of contract activity.

Identifier	Definition	
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 contracts are deployed with stable compiler version.

Identifier	Definition	Severity
COM-04	Potential resource exhaustion errors in **`MultiRewardStaking`**	Minor 🟢

Loops may throw out of gas errors upon executing. Some functions may become gas-costly, if size of arrays grow:

`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 calls.

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