InterFi NETWORK

SMART CONTRACT AUDIT

interfinetwork

hello@interfi.network

https://interfi.network

PREPARED FOR

4M

INTRODUCTION

Auditing Firm	InterFi Network
Client Firm	4M
Methodology	Automated Analysis, Manual Code Review
Language	Solidity
Contract	Multiple contracts
Blockchain	Not available
Centralization	Active ownership
Commit	de6eed417efe9f6dac328adc613cc2b56597b43d
Website	https://www.4m-bsc.com/
Telegram	https://t.me/MMMM_BSC/
X (Twitter)	https://twitter.com/4m_bsc/
Report Date	January 04, 2024

ℹ 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	3	0	2	5	1
Acknowledged	0	0	0	3	1
Resolved	0	0	0	0	0
Noteworthy Privileges	Check PAGE 17 for controlled and privileged roles				

⚠️ In the context of this audit, it is important to note that the source codes under review have not yet been deployed on the main-net. Consequently, they are subject to potential modifications or alterations before their eventual deployment. It is essential to consider this, as any changes made after this audit but before deployment could affect the security and functionality of the smart contracts, thus impacting the audit's relevance and accuracy.

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 4M 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:

- MMMM.sol
- Partner.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.

ℹ 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

MMMM

| **IBEP20** | Implementation | |||

| └ | totalSupply | External ! | | |NO! |

| └ | decimals | External ! | | |NO! |

| └ | symbol | External ! | | |NO! |

| └ | name | External ! | | |NO! |

| └ | balanceOf | External ! | | |NO! |

| └ | transfer | External ! | 🛑 | |NO! |

| └ | allowance | External ! | | |NO! |

| └ | approve | External ! | 🛑 | |NO! |

| └ | transferFrom | External ! | 🛑 | |NO! |

||||||

| **IMinter** | Implementation | |||

| └ | setMinter | External ! | 🛑 | |NO! |

| └ | updateTime | External ! | 🛑 | |NO! |

||||||

| **SafeMath** | Library | |||

| └ | add | Internal 🔒 | | |

| └ | sub | Internal 🔒 | | |

| └ | mul | Internal 🔒 | | |

| └ | div | Internal 🔒 | | |

| └ | mod | Internal 🔒 | | |

||||||

| **Context** | Implementation | |||

| └ | _msgSender | Internal 🔒 | | |

| └ | _msgData | Internal 🔒 | | |

||||||

| **Ownable** | Implementation | Context |||

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

| └ | owner | Public ! | | NO! |

| └ | transferOwnership | Public ! | 🛑 | onlyOwner |

||||||

| **StableSwap** | Interface | |||

| └ | add_liquidity | External ! | 🛑 | NO! |

| └ | remove_liquidity_one_coin | External ! | 🛑 | NO! |

||||||

| **ReentrancyGuard** | Implementation | |||

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

||||||

| **MMMM** | Implementation | Ownable, ReentrancyGuard |||

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

| └ | updateTime | Public ! | 🛑 | NO! |

| └ | transfer | Public ! | 🛑 | onlyOwner |

| └ | register | Public ! | 🛑 | NO! |

| └ | _register | Internal 🔒 | 🛑 | nonReentrantRegister check4MStatus |

| └ | buyTickets | External ! | 🛑 | check4MStatus |

| └ | staticIncomeParams | Internal 🔒 | | |

| └ | _deposit | Private 🔐 | 🛑 | |

| └ | invest | Public ! | 🛑 | nonReentrantInvest check4MStatus validateLadder |

| └ | _saveOrder | Private 🔐 | 🛑 | |

| └ | _unlockOrder | Private 🔐 | 🛑 | |

| └ | _setParentPV | Private 🔐 | 🛑 | |

| └ | _setUserLevel | Private 🔐 | 🛑 | |

| └ | incomeStatic | Public ! | 🛑 | check4MStatus |

| └ | _incomeStatic | Private 🔐 | 🛑 | |

| └ | inviteRate | Private 🔐 | | |

| └ | _incomeInvite | Private 🔐 | 🛑 | |

| └ | _incomeTeam | Private 🔐 | 🛑 | |

| └ | withdraw | Public ! | 🛑 | check4MStatus |

| └ | withdrawPool | External ! | 🛑 | onlyOwner |

| └ | withdrawAll | External ! | 🛑 | onlyOwner |

| └ | _withdraw | Private 🔐 | 🛑 | |

| └ | _restart | Private 🔐 | 🛑 | |

| └ | _compensate | Private 🔐 | 🛑 | |

| └ | setTicketPrice | External ! | 🛑 | onlyOwner |

| └ | setTicketRate | External ! | 🛑 | onlyOwner |

| └ | setPartner | External ! | 🛑 | onlyOwner |

| └ | setCompensateMax | External ! | 🛑 | onlyOwner |

| └ | inviterCode | External ! | | NO! |

| └ | inviter | External ! | | NO! |

| └ | getUser | External ! | | NO! |

| └ | isUser | Public ! | | NO! |

| └ | isActiveUser | External ! | | NO! |

| └ | isPartner | External ! | | NO! |

| └ | leaderNum | External ! | | NO! |

| └ | params | External ! | | NO! |

| └ | fundPools | External ! | | NO! |

| └ | getLastOrder | External ! | | NO! |

| └ | unlockOrderList | External ! | | NO! |

| └ | orderList | External ! | | NO! |

| └ | inviteList | External ! | | NO! |

Partner

| **IBEP20** | Implementation | |||

| └ | totalSupply | External ! | | NO! |

| └ | decimals | External ! | | NO! |

| └ | symbol | External ! | | NO! |

| └ | name | External ! | | NO! |

| └ | balanceOf | External ! | | NO! |

| └ | transfer | External ! | 🛑 | NO! |

| └ | allowance | External ! | | NO! |

| └ | approve | External ! | 🛑 | NO! |

| └ | transferFrom | External ❗ | 🛑 | NO❗ |

||||||

| **SafeMath** | Library | |||

| └ | add | Internal 🔒 | | |

| └ | sub | Internal 🔒 | | |

| └ | mul | Internal 🔒 | | |

| └ | div | Internal 🔒 | | |

| └ | mod | Internal 🔒 | | |

||||||

| **Context** | Implementation | |||

| └ | _msgSender | Internal 🔒 | | |

| └ | _msgData | Internal 🔒 | | |

||||||

| **Ownable** | Implementation | Context |||

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

| └ | owner | Public ❗ | | NO❗ |

| └ | transferOwnership | Public ❗ | 🛑 | onlyOwner |

| └ | isAdmin | Public ❗ | | NO❗ |

| └ | setAdmin | Public ❗ | 🛑 | onlyOwner |

| └ | unsetAdmin | Public ❗ | 🛑 | onlyOwner |

||||||

| **ReentrancyGuard** | Implementation | |||

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

||||||

| **Partner** | Implementation | Ownable, ReentrancyGuard |||

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

| └ | transfer | Public ❗ | 🛑 | onlyOwner |

| └ | register | Public ❗ | 🛑 | NO❗ |

| └ | _register | Internal 🔒 | 🛑 | nonReentrantRegister |

| └ | join | Public ❗ | 🛑 | NO❗ |

| └ | buyAmount | External ❗ | | NO❗ |

| └ | isPartner | External ❗ | | NO❗ |

| └ | inviterCode | External ❗ | | NO❗ |

| └ | inviter | External ❗ | | NO❗ |

| └ | levelList | External ❗ | | NO❗ |

| └ | getLevel | External ❗ | | NO❗ |

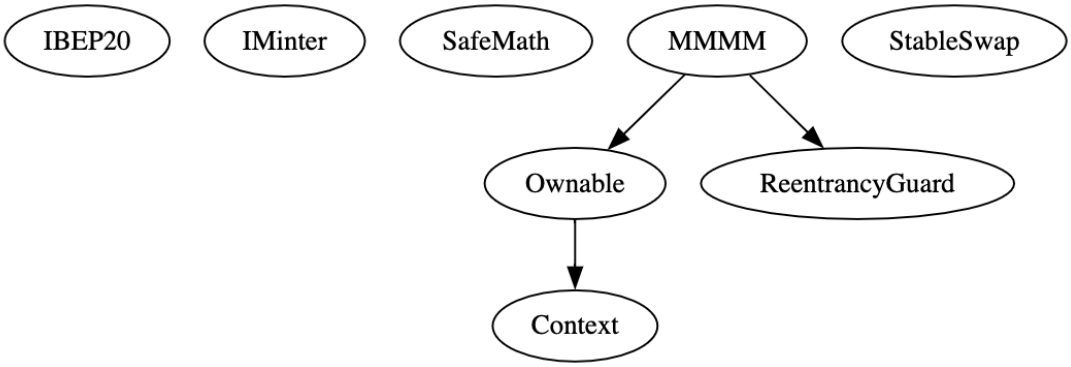
| └ | setLevel | External ❗ | 🛑 | onlyOwner |

| └ | getUser | External ❗ | | NO❗ |

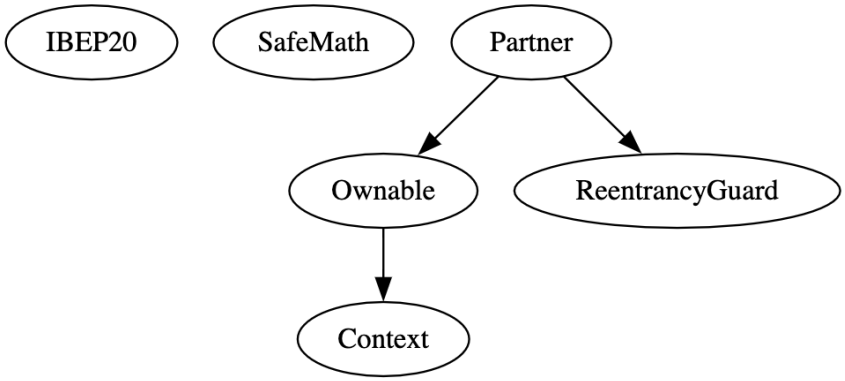
| └ | inviteList | External ❗ | | NO❗ |

INHERITANCE GRAPH

MMMM



Partner



MANUAL REVIEW

Identifier	Definition	Severity
CEN-01	Centralized privileges	Critical 🔴
4M-01	Contract owner can transfer any BEP20 token from contract with `transfer` function in `MMMM` and `Partner` contracts	
4M-02	Contract owner can withdraw funds from specified pools without checks with `withdrawPool` function in `MMMM` contract	
4M-03	Contract owner can withdraw all the Swap LP tokens from the contract with `withdrawAll` function in `MMMM` contract	

Important `onlyOwner` centralized privileges are listed below:

MMMM

`transferOwnership`
`transfer`
`withdrawPool`
`withdrawAll`
`setTicketPrice`
`setTicketRate`
`setPartner`
`setCompensateMax`

Partner

`transferOwnership`
`setAdmin`
`unsetAdmin`
`transfer`
`setLevel`

RECOMMENDATION

Deployers', owners', administrators', 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.

Identifier	Definition	Severity
4M-04	Lack of appropriate access restrictions	Critical 🔴

Below mentioned functions are set without appropriate access restrictions, meaning anyone can call these functions:

MMMM

`updateTime`
`register`

Partner

`register`
`join`

RECOMMENDATION

Ensure that functions performing critical operations have adequate access control to prevent unauthorized use.

Identifier	Definition	Severity
LOG-01	Lack of appropriate input validation	Medium 🟡

Below mentioned functions are set without appropriate input restrictions:

MMMM

`setTicketPrice`
`setTicketRate`
`setPartner`
`setCompensateMax`
`buyTickets`
`register`
`withdrawPool`
`updateTime`

Partner

`setLevel`

RECOMMENDATION

These functions should be provided appropriate input restrictions to allow value change within set parameters.

Identifier	Definition	Severity
LOG-02	Potential front-running	Minor

Potential front-running happens when an attacker observes a transaction swapping tokens or adding liquidity without setting restrictions on slippage or minimum output amount. The attacker can manipulate the exchange rate by front-running a transaction to purchase assets and make profits by back-running a transaction to sell assets. Below mentioned functions are called without setting restrictions on slippage or minimum output:

MMMM

```
register
buyTickets
invest
withdraw
withdrawPool
```

Partner

```
register
join
transfer
```

RECOMMENDATION

Use mechanisms like commit-reveal schemes, time locks, or oracle-based price feeds to make the outcomes less predictable.

Identifier	Definition	Severity
LOG-03	Re-entrancy	Critical 🔴

Below mentioned functions are used without re-entrancy guard:

MMMM

```
buyTickets
withdraw
_withdraw
```

Partner

```
join
```


RECOMMENDATION

Use Checks Effects Interactions pattern when handing over the flow to an external entity and/or guard functions against re-entrancy attacks. Re-entrancy guard is used to prevent re-entrant calls. Learn more: https://consensys.github.io/smart-contract-best-practices/attacks/reentrancy/

Identifier	Definition	Severity
COD-02	Reliance on `block.timestamp`	Minor

Be aware that the timestamp of the block can be manipulated by a miner. When the contract uses the timestamp to seed a random number, the miner can actually post a timestamp within 15 seconds of the block being validated, effectively allowing the miner to precompute an option more favorable to their chances.

RECOMMENDATION

Use more reliable time sources or add logic to mitigate small miner manipulations.

Identifier	Definition	Severity
COD-03	Potential price manipulation in `_deposit`	Medium 🟡

`_deposit` function interacts with an external swap contract without checking slippage, which is exploited through price manipulation.

RECOMMENDATION

Implement slippage protection or price checks.

Identifier	Definition	Severity
COD-04	Lack of appropriate visibility identifiers	Minor

Below mentioned functions are missing appropriate visibility identifiers:

`Partner`

`_market` array

RECOMMENDATION

Make `_market` private if its visibility is not required by design.

Identifier	Definition	Severity
COD-05	Use of fixed addresses in `_market` array	Unknown 🟤
COD-06	Unknown external addresses	

Smart contract uses a fixed list of addresses in `_market` array for transferring funds. This is a point of centralized manipulation if these addresses are controlled by a single entity.

Partner

```
0x8eC45058C5CDdD0ecc40918092122798846C1F45,
0xD9D274473735e6b4943a93A662a6D5E170DFAA81,
0xce97C1d7E64695820C448230fd267B3388F42963,
0x365fF2690aC0b5d183Fa61FB5Dd5e1Ff441dCd38,
0x2BDAAc2C534c02E5D779EAcB1b302c95DFb3ca13,
0xA7aaE0cbB10725f759EE715115C2D03522A68998,
0xD428d539B8AF6C35b8B6f5C505D5CA94774d0120,
0x444B96DDAe773C9f4AA303DE0Dccd56938A7e48b,
0x8adD3c4Da47c3a81133a04FA35De365B91b9e7d3,
0x4E3138aF6D96e5cAf10772CDe01F1E3C13Fd83a7
```


RECOMMENDATION

Implement a mechanism to update these addresses or decentralize the decision of where the funds are sent.

Identifier	Definition	Severity
COD-10	Direct and indirect dependencies	Unknown

Smart contracts interact with third-party protocols such as Market Makers, specific BEP20 token contracts (e.g., `_USDT`), and external smart contracts like `StableSwap` and `IMinter` in `MMMM` contract. These external entities are treated as black boxes in this audit, with an assumption of their reliability. However, real-world risks include potential compromises or changes in these entities, leading to impacts like increased fees or deprecated functionalities. Continuous monitoring and adaptability are essential due to these potential external changes affecting the contracts' operations.

RECOMMENDATION

Regularly review and audit external contracts for security updates and changes, and implement circuit breakers or pause functions in smart contracts to quickly respond to external changes or threats.

Identifier	Definition	Severity
COD-11	Uncontrolled update of `_marketIndex`	Minor

`_marketIndex` is incremented without bounds and resets to 0 when it exceeds the `_market` array length.

This behavior can be manipulated.

RECOMMENDATION

Implement better mechanism for managing `_marketIndex` updates.

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	Severity
VOL-01	Irrelevant code	Minor

Redundant code in `SafeMath`

RECOMMENDATION

Remove redundant code.

Identifier	Definition	Severity
COM-01	Floating compilers	Minor

Compilers are set to `^0.8.10`

RECOMMENDATION

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

Identifier	Definition	Severity
COM-04	Unbounded loops	Minor

Below mentioned functions contain loops that may iterate over large datasets in future, which could result in out-of-gas errors:

MMMM

```
_incomeInvite
_incomeTeam
```


RECOMMENDATION

Implement pagination or limit the number of iterations per transaction.

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