



SMART CONTRACT AUDIT



interfinetwork



hello@interfi.network



<https://interfi.network>

PREPARED FOR

PAYVERTISE



INTRODUCTION

Auditing Firm	InterFi Network
Client Firm	Payvertise
Methodology	Automated Analysis, Manual Code Review
Language	Solidity
Contract	0xb4E14166F6dE109f800C52A84a434C383137C8dC
Blockchain	Binance Smart Chain
Centralization	Active Ownership
Commit	b8777c91d5788bb36bb45f41l88541c063ddee5e
Website	https://payvertise.com/
Telegram	https://t.me/payvertisechat/ https://t.me/payvertiseann/
X (Twitter)	https://twitter.com/payvertise_/
Discord	https://discord.com/invite/payvertise/
Report Date	January 10, 2024


 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	0	0
Acknowledged	0	1	0	6	1
Resolved	1	0	0	2	0
Important Privileges	Launch, Rescue Token, Add Pair, Del Pair, Set Rate Drop, Set Rate Stake				
Important Functions	Unstake Airdrop, Unstake Staking				

 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.

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

 Please note that smart contract relies on external stake contract for staking functionalities.



TABLE OF CONTENTS

TABLE OF CONTENTS	4
SCOPE OF WORK.....	5
AUDIT METHODOLOGY	6
RISK CATEGORIES	8
CENTRALIZED PRIVILEGES	9
AUTOMATED ANALYSIS.....	10
INHERITANCE GRAPH	14
MANUAL REVIEW	15
DISCLAIMERS	28
ABOUT INTERFI NETWORK	31



SCOPE OF WORK

InterFi was consulted by Payvertise 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:

- PAYVERTISE.sol

i 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/0xb4e14166f6de109f800c52a84a434c383137c8dc#code	
Contract Name	PAYVERTISE
Compiler Version	0.8.19
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	<ul style="list-style-type: none">○ 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.

 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.

 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

| ****ERC20**** | Implementation | Context, IERC20, IERC20Metadata |||

| ^L | <Constructor> | Public  |  | NO  |

| ^L | name | Public  | | NO  |


| ^L | symbol | Public  | | NO  |

| ^L | decimals | Public  | | NO  |




| ^L | totalSupply | Public  | | NO  |

| ^L | getCirculatingSupply | Internal  | | |




| ^L | balanceOf | Public  | | NO  |




| ^L | getAirdropRewards | Public  | | NO  |




| ^L | getStakedRewards | Public  | | NO  |

| ^L | transfer | Public  |  | NO  |



| ^L | allowance | Public  | | NO  |



| ^L | approve | Public  |  | NO  |



| ^L | transferFrom | Public  |  | NO  |



| ^L | increaseAllowance | Public  |  | NO  |

| ^L | decreaseAllowance | Public  |  | NO  |

| ^L | transfer | Internal  |  | |

| ^L | _setexclusion | Internal  |  | |

| ^L | _setairdropcontract | Internal  |  | |

| ^L | _setstakecontract | Internal  |  | |



```

| L | _setAirdropRate | Internal | 🔒 | 🔴 | | |
| L | _setStakingRate | Internal | 🔒 | 🔴 | | |
| L | _setMaxSupply | Internal | 🔒 | 🔴 | | |
| L | _unstakeAirdrop | Internal | 🔒 | 🔴 | | |
| L | _unstakeStaking | Internal | 🔒 | 🔴 | | |
| L | _mint | Internal | 🔒 | 🔴 | | |
| L | _burn | Internal | 🔒 | 🔴 | | |
| L | _approve | Internal | 🔒 | 🔴 | | |
| L | _spendAllowance | Internal | 🔒 | 🔴 | | |
| L | _beforeTokenTransfer | Internal | 🔒 | 🔴 | | |
| L | _afterTokenTransfer | Internal | 🔒 | 🔴 | | |

```

```

|||||

```

```

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

```

```

| L | <Constructor> | Public | ! | 🔴 | NO! |

```

```

| L | owner | Public | ! | NO! |

```

```

| L | _checkOwner | Internal | 🔒 | | |

```

```

| L | renounceOwnership | Public | ! | 🔴 | onlyOwner |

```

```

| L | transferOwnership | Public | ! | 🔴 | onlyOwner |

```

```

| L | _transferOwnership | Internal | 🔒 | 🔴 | | |

```

```

|||||

```

```

| **SafeERC20** | Library | |||

```

```

| L | safeTransfer | Internal | 🔒 | 🔴 | | |

```

```

| L | safeTransferFrom | Internal | 🔒 | 🔴 | | |

```

```

| L | safeApprove | Internal | 🔒 | 🔴 | | |

```

```

| L | safeIncreaseAllowance | Internal | 🔒 | 🔴 | | |

```

```

| L | safeDecreaseAllowance | Internal | 🔒 | 🔴 | | |

```

```

| L | safePermit | Internal | 🔒 | 🔴 | | |

```

```

| L | _callOptionalReturn | Private | 🔒 | 🔴 | | |

```

```

|||||

```

```

| **EnumerableSet** | Library | |||

```

```

| L | _add | Private | 🔒 | 🔴 | | |



```

```

| L | _remove | Private | 🔒 | 🔴 | | |

```



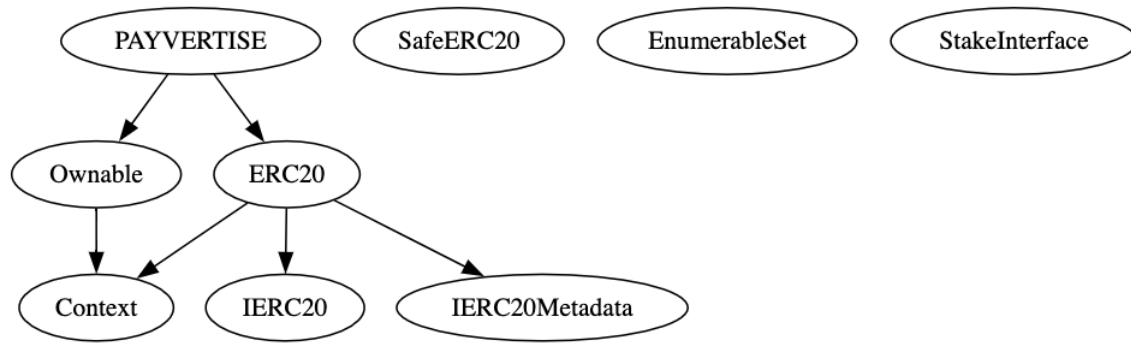
```
| **StakeInterface** | Interface | |||
| ^ | StakedAmount | External ! | |NO ! |
| ^ | getAirdropUnstakePercent | External ! | |NO ! |
| ^ | getStakingUnstakePercent | External ! | |NO ! |
| ^ | StakedAirdrop | External ! | |NO ! |
| ^ | unStakeAirdrop | External ! |  |NO ! |
| ^ | unStakeStaking | External ! |  |NO ! |
| | | | |
```



Function	Visibility	State	Access	Notes
PAYVERTISE	Implementation	ERC20, Ownable		
<Constructor>	Public	!	●	ERC20
decimals	Public	!		NO!
transfer	Public	!	●	NO!
transferFrom	Public	!	●	NO!
_pvtTransfer	Internal	🔒	●	
launched	Internal	🔒		
buyFees	Internal	🔒	●	
sellFees	Internal	🔒	●	
takeAdvertisingFee	Internal	🔒	●	
circulatingSupply	Public	!		NO!
getMinterLength	Public	!		NO!
getPair	Public	!		NO!
isPair	Public	!		NO!
unstakeairdrop	Public	!	●	NO!
unstakestaking	Public	!	●	NO!
launch	Public	!	●	onlyOwner
rescueToken	External	!	●	onlyOwner
setIsFeeExempt	External	!	●	onlyOwner
setPresaleWallet	External	!	●	onlyOwner
setFeeSettings	External	!	●	onlyOwner
setAdvertisingWallet	External	!	●	onlyOwner
addPair	Public	!	●	onlyOwner
delPair	Public	!	●	onlyOwner
setexclusion	External	!	●	onlyOwner
setairdropcontract	External	!	●	onlyOwner
setstakecontract	External	!	●	onlyOwner
setratedrop	External	!	●	onlyOwner
setratestake	External	!	●	onlyOwner
setfutureAirdropUnstakeDate	External	!	●	onlyOwner
setfutureStakingUnstakeDate	External	!	●	onlyOwner



INHERITANCE GRAPH



INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT



MANUAL REVIEW

Identifier	Definition	Severity
CEN-01	Centralized privileges	Major 🟡
PAY-01	Trading must be enabled to allow EOAs to transfer/trade assets	
PAY-02	Privileged role can remove active contract pair	

Important onLyOwner centralized privileges are listed below:

```

renounceOwnership()
transferOwnership()
launch()
rescueToken()
setIsFeeExempt()
setPresaleWallet()
setFeeSettings()
setAdvertisingWallet()
addPair()
delPair()
setexclusion()
setairdropcontract()
setstakecontract()
setratedrop()
setratestake()
setfutureAirdropUnstakeDate()
setfutureStakingUnstakeDate()

```

INTERFI
CONFIDENTIAL

INTERFI
CONFIDENTIAL



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. It is recommended to:

Implement multi-signature wallets: Require multiple signatures from different parties to execute certain sensitive functions within contracts. This spreads control and reduces the risk of a single party having complete authority.

Use a decentralized governance model: Implement a governance model that enables token holders or other stakeholders to participate in decision-making processes. This can include voting on contract upgrades, parameter changes, or any other critical decisions that impact the contract's functioning.

ACKNOWLEDGEMENT

Payvertise project has confirmed that privileged roles are used as intended.



Identifier	Definition	Severity
CEN-02	Initial asset distribution	Minor ●

All of the initially minted assets are sent to the project owner when deploying the contract. This can be an issue as the project owner can distribute tokens without consulting the community.

```
uint256 _circulatingSupply = 80_000_000 * 1e18; //Current Circulating supply
according to tokenomics.
_mint(_msgSender(), _circulatingSupply);
```


RECOMMENDATION

Project must communicate with stakeholders and obtain the community consensus while distributing assets.

ACKNOWLEDGEMENT

Payvertise project will distribute tokens after acquiring broader consensus, as per their pre-determined tokenomics.



Identifier	Definition	Severity
CEN-11	Token mint after initial deploy	Minor 

Smart contract has `_setMaxSupply()` function, which declares maximum supply. `maxSupply` is not modifiable after contract deployment, which is a good practice.

`_circulatingSupply` is a private variable. `_circulatingSupply` is increased every time the `_mint()` function is called and when rewards are paired to balances in `_transfer()`.

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL

RECOMMENDATION


Add a safeguard in the `_mint()` function to prevent minting more tokens than `maxSupply()`.

ACKNOWLEDGEMENT

`maxSupply` is intended to cap the total amount of tokens that can ever be minted. Payvertise team argued that this is required by design to cater for stake rewards.

```
_setMaxSupply(_totalSupply);
uint256 _totalSupply = 100_000_000 * 1e18; //total MAX supply
```



Identifier	Definition	Severity
LOG-01	Lack of appropriate input limit	Minor 

Below mentioned functions are set without any arbitrary upper limits:

```
setratedrop()  
setratestake()  
setfutureAirdropUnstakeDate()  
setfutureStakingUnstakeDate()
```

RECOMMENDATION

These functions should be provided appropriate upper input boundaries.

ACKNOWLEDGEMENT

Payvertise project has argued that these value changes are possible by contract owner only.



Identifier	Definition	Severity
LOG-02	Potential front-running	Minor 

Potential front-running also classified as – sandwich attack 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:

```
transfer()
transferFrom()
```

Since smart contract modifies state based on external input (StakeInterface), it is vulnerable to front-running attacks.

RECOMMENDATION

Use commit-reveal scheme to deter front-runners.

ACKNOWLEDGEMENT

Payvertise project has argued that front-running is a by-product of many EVM chains, such as Ethereum, Binance Chain. Contract functions such as transaction tax should lower front-running viability.



Identifier	Definition	Severity
LOG-03	Re-entrancy	Critical ●
LOG-03-01	Re-entrancy in BSC#0xb4E14166F6dE109f800C52A84a434C383137C8dC	Medium ●

Below mentioned functions are used without re-entrancy guard:

```
unstakeairdrop()
unstakestaking()
```

These functions are potentially risky if the external contracts are malicious.

RECOMMENDATION

Implement re-entrancy guard, especially in above mentioned functions.

Rearrange the order of operations in `unstakeairdrop()` and `unstakestaking()` to follow the Checks-Effects-Interactions pattern – first update the contract's state, and then interact with external contracts.


LOG-03 RESOLUTION

Payvertise project has added critical require statements in `unstakeairdrop()` and `unstakestaking()` functions to follow the Checks-Effects-Interactions pattern. This should lower re-entrancy risk.

LOG-03-01 RECOMMENDATION

Implement standard OpenZeppelin **ReentrancyGuard** library.



Identifier	Definition	Severity
LOG-04	ERC20 compliance	Minor 

Smart contract deviates from the standard ERC20 implementation by adding additional logic in `transfer()` and `transferFrom()` functions. This is not directly an issue but note that these changes may contradict with other contracts expecting standard ERC20 behavior.

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL

RECOMMENDATION

Use standard ERC20 implementation.

ACKNOWLEDGEMENT

Payvertise project has argued that both `_transfer()`, and `_pvtTransfer()` functions have unique execution styles, and ERC20 library is modified as required by the design.



Identifier	Definition	Severity
COD-02	Timestamp manipulation via <code>block.timestamp</code>	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

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

RESOLUTION

Payvertise project has argued that timestamp of a block is not being used to generate random numbers, miner manipulation should be minimal.



Identifier	Definition
COD-06	Hardcoded fee

Advertising fee is hardcoded and cannot be changed.

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL



Identifier	Definition	Severity
COD-10	Direct and indirect dependencies	Unknown 🟤

Smart contract is interacting with third party protocols e.g., Market Makers, External Contracts, External implementation of staking functionalities, 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.

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL


RECOMMENDATION

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

ACKNOWLEDGEMENT

Payvertise will inspect third party dependencies regularly, and push updates as required.



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

Smart contract uses 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.

ACKNOWLEDGEMENT

Payvertise project has argued that events are added as required. *No other changes are made to the code.*



Identifier	Definition	Severity
COM-01	Floating compiler status	Minor 

Compiler is set to:

```
pragma solidity ^0.8.9;
```

TERFI
CONFIDENTIALINTERFI
CONFIDENTIAL

RECOMMENDATION

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

RESOLUTION

Smart contract will be deployed with stable compiler.



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.

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL



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