

SMART CONTRACT AUDIT



interfinetwork



hello@interfi.network



<https://interfi.network>

PREPARED FOR

EKTA BLOCKCHAIN WALLETS



INTRODUCTION

Auditing Firm	InterFi Network
Client Firm	Ekta Blockchain Wallets
Methodology	Automated Analysis, Manual Code Review
Language	Solidity
Contract	Multiple contracts
Blockchain	
Centralization	Active ownership
Commit	8a1d6b9174e5473d65b22a54ebcfdffc21c74b8d
Website	http://ekta.io
Telegram	https://t.me/ekta_community
Twitter	https://twitter.com/EktaChain
Discord	https://discord.com/invite/ektachain
Report Date	April 20, 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	0	0
Acknowledged	0	1	1	3	1
Resolved	0	0	1	3	0

 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.



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.....	29
ABOUT INTERFI NETWORK.....	32


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



SCOPE OF WORK

InterFi was consulted by Ekta Blockchain Wallets 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:

- BlockchainWallet.sol
- BlockchainWalletBeacon.sol
- BlockchainWalletFactory.sol
- EktaRegistry.sol

 If source codes are not deployed on the main net, they can be modified or altered before main-net deployment.

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



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

****EBW_VERSION01_0415****

Wallet

```

| **IRegistry** | Interface | |||
| L | addUserWallet | External ! |  | NO ! |
| L | removeUserWallet | External ! |  | NO ! |
|||||
| **BlockchainWallet** | Implementation | Initializable |||
| L | <Constructor> | Public ! |  | NO ! |
| L | initialize | Public ! |  | initializer validRequirement |
| L | addOwner | Public ! |  | onlyWallet ownerDoesNotExist notNull validRequirement |
| L | removeOwner | Public ! |  | onlyWallet ownerExists |
| L | changeRequirement | Public ! |  | onlyWallet validRequirement |
| L | changeWhitelist | Public ! |  | onlyWallet |
| L | createWhitelistTransaction | Public ! |  | NO ! |
| L | sendEktaTransaction | Public ! |  | NO ! |
| L | sendERC20Transaction | Public ! |  | NO ! |
| L | createAddOwnerTransaction | Public ! |  | NO ! |

```

INTERFI
CONFIDENTIAL



	└		createRemoveOwnerTransaction		Public	!		🔴		NO	!	
	└		createRequirementTransaction		Public	!		🔴		NO	!	
	└		confirmTransaction		Public	!		🔴		ownerExists transactionExists notConfirmed		
	└		revokeConfirmation		Public	!		🔴		ownerExists confirmed notExecuted		
	└		executeTransaction		Public	!		🔴		ownerExists confirmed notExecuted		
	└		external_call		Internal	🔒		🔴				
	└		addTransaction		Internal	🔒		🔴		notNull		
	└		isConfirmed		Public	!				NO	!	
	└		getConfirmationCount		Public	!				NO	!	
	└		getTransactionDetails		Public	!				NO	!	
	└		getTransactionCount		Public	!				NO	!	
	└		getOwners		Public	!				NO	!	
	└		getConfirmations		Public	!				NO	!	
	└		getTransactionIds		Public	!				NO	!	
	└		<Receive Ether>		External	!		🔴		NO	!	

Wallet Beacon

	BlockchainWalletBeacon		Implementation								
	└		<Constructor>		Public	!		🔴		validRequirement	
	└		update		Public	!		🔴		onlyWallet	
	└		implementation		Public	!				NO	!
	└		addOwner		Public	!		🔴		onlyWallet ownerDoesNotExist notNull validRequirement	
	└		removeOwner		Public	!		🔴		onlyWallet ownerExists	
	└		changeRequirement		Public	!		🔴		onlyWallet validRequirement	
	└		createUpgradeImplementationTransaction		Public	!		🔴		NO	!
	└		createAddOwnerTransaction		Public	!		🔴		NO	!
	└		createRemoveOwnerTransaction		Public	!		🔴		NO	!
	└		createRequirementTransaction		Public	!		🔴		NO	!



```

| L | confirmTransaction | Public ! | ● | ownerExists transactionExists notConfirmed |
| L | revokeConfirmation | Public ! | ● | ownerExists confirmed notExecuted |
| L | executeTransaction | Public ! | ● | ownerExists confirmed notExecuted |
| L | external_call | Internal 🔒 | ● | |
| L | addTransaction | Internal 🔒 | ● | notNull |
| L | isConfirmed | Public ! | | NO ! |
| L | getConfirmationCount | Public ! | | NO ! |
| L | getTransactionCount | Public ! | | NO ! |
| L | getOwners | Public ! | | NO ! |
| L | getConfirmations | Public ! | | NO ! |
| L | getTransactionIds | Public ! | | NO ! |

```

Wallet Factory

```

| **BlockchainWalletFactory** | Implementation | AccessControl |||
| L | <Constructor> | Public ! | ● | NO ! |
| L | create | External ! | ● | onlyRole |
| L | getImplementation | Public ! | | NO ! |
| L | getBeacon | Public ! | | NO ! |
| L | getWallets | Public ! | | NO ! |
| **IRegistry** | Interface | |||
| L | addUserWallet | External ! | ● | NO ! |
| L | removeUserWallet | External ! | ● | NO ! |

```

Registry

```

| **IUserWalletFactory** | Interface | |||
| L | createUserWallet | External ! | ● | NO ! |
|||||
| **IBlockchainWalletFactory** | Interface | |||

```



| L | create | External ! | ● | NO ! |

|||||

| ****EktaRegistry**** | Implementation | Initializable, OwnableUpgradeable, UUPSUpgradeable |||

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

| L | initialize | Public ! | ● | initializer |

| L | addMerkleRoot | External ! | ● | onlyOwner |

| L | addDappUserMap | External ! | ● | onlyOwner |

| L | verifyDappUserMerkle | External ! | | NO ! |

| L | setWalletFactories | External ! | ● | onlyOwner |

| L | CreateBlockchainWallet | External ! | ● | onlyOwner |

| L | getUserWallets | External ! | | NO ! |

| L | addUserWallet | Public ! | ● | NO ! |

| L | removeUserWallet | Public ! | ● | NO ! |

| L | getUserAddressesFromMail | External ! | | NO ! |

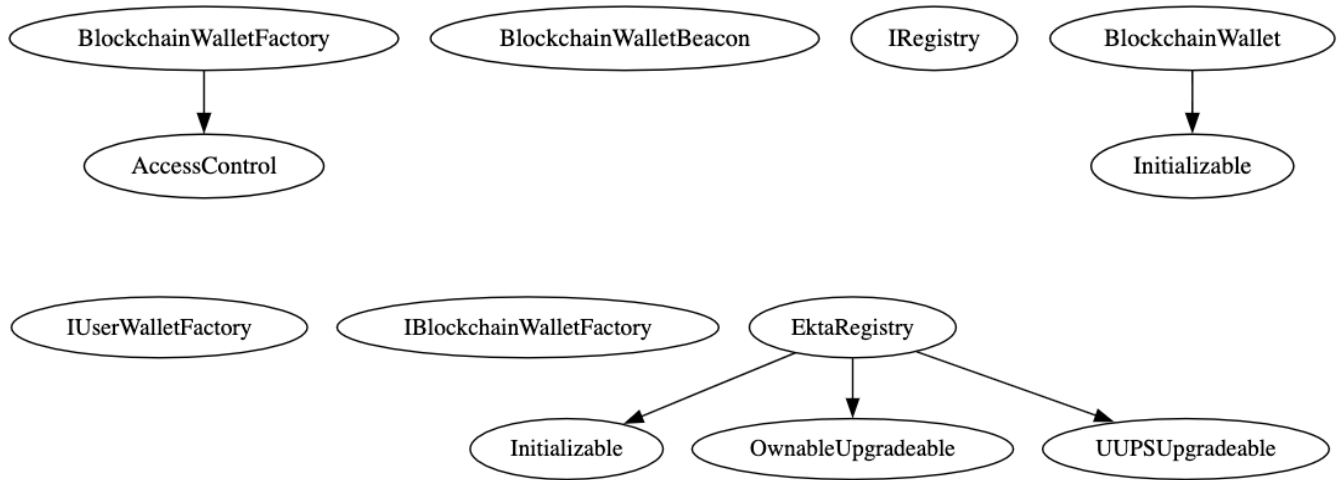
| L | _authorizeUpgrade | Internal 🔒 | ● | onlyOwner |

TERFI
CONFIDENTIAL

INTERFI
CONFIDENTIAL



INHERITANCE GRAPH



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



MANUAL REVIEW

Identifier	Definition	Severity
CEN-01	Possible centralization	Major 🟡
CEN-07	Authorizations and access controls	

Wallet

onlyWallet ensures that the transaction is sent by wallet contract. This access control is provided to:

```
addOwner()
removeOwner()
changeRequirement()
changeWhitelist()
```

Wallet Beacon

onlyWallet ensures that the transaction is sent by wallet beacon contract. This access control is provided to:

```
update()
addOwner()
removeOwner()
changeRequirement()
```

Wallet Factory

onlyRole access control is provided to below mentioned function:

```
create()
```

Registry

onlyOwner centralized privileges are provided to below mentioned functions:

```
addMerkleRoot()
addDappUserMap()
setWalletFactories()
CreateBlockchainWallet()
_authorizeUpgrade()
```



UNIQUE MODIFIERS

1. **onlyWallet** - This modifier restricts the execution of a function to be called only if the caller (`msg.sender`) is the address of the contract itself. This is useful for ensuring that only internal contract functions can call certain functions.
2. **ownerExists** - This modifier checks if the given address is an owner. It requires that the `_owner` address is an existing owner by checking if the `isOwner` mapping returns true for the given address. Functions with this modifier can only be executed by an existing owner.
3. **ownerDoesNotExist** - This modifier checks if the given address is not an owner. It requires that the `_owner` address is not an existing owner by checking if the `isOwner` mapping returns false for the given address. Functions with this modifier can only be executed if the caller is not an existing owner.
4. **transactionExists** - This modifier checks if a transaction with the given transaction ID exists. It requires that the transaction with the given `_transactionId` exists in the transactions mapping. Functions with this modifier can only be executed if the transaction exists.
5. **confirmed** - This modifier checks if a transaction is confirmed by a specific owner. It requires that the given `_transactionId` is confirmed by the owner specified in the `_owner` parameter. Functions with this modifier can only be executed if the transaction is confirmed by the specified owner.
6. **notConfirmed** - This modifier checks if a transaction is not confirmed by a specific owner. It requires that the given `_transactionId` is not confirmed by the owner specified in the `_owner` parameter. Functions with this modifier can only be executed if the transaction is not confirmed by the specified owner.
7. **notExecuted** - This modifier checks if a transaction is not executed. It requires that the given `_transactionId` is not marked as executed in the transactions mapping. Functions with this modifier can only be executed if the transaction is not executed.



RECOMMENDATION

Contract creators, contract owners, administrators, and all other privileged roles' private keys should be secured carefully. Wallet and wallet beacon contracts allow multiple owners to manage centralization, which mitigate some centralization risks. However, it is still essential to consider that owners are in control of contracts and their behaviors. Centralization risks depend on how the owners are chosen and managed. If the owners are from a single organization or if there is collusion among them, there might still be some centralization risks involved.

ACKNOWLEDGEMENT

Ekta project team uses Gnosis multi-sig safe to manage centralization related risks.

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
CEN-09	Use of proxy and upgradeable contracts	Medium ●

Smart contracts use BeaconProxy and UUPSUpgradeable mechanism. Privileged roles can initiate new implementation. Contract upgradeability allows privileged roles to change current contract implementation.

In Wallet Beacon, UpgradeableBeacon() allows to change the implementation contract address, but this change will not affect the main contract's logic.

RECOMMENDATION

Test and validate current contract thoroughly before deployment. Future contract upgradeability negatively elevates centralization risk. Integrate proxy only when necessary. While proxy contracts are great for robust deployments while maintaining the upgradeable flexibility, proxy codes are prone to new security or logical issues that may compromise the project.

ACKNOWLEDGEMENT

Ekta Blockchain Wallets project is under development phase. Contract upgradeability is required to optimize code, tackle novel vulnerabilities, and introduce new utilities.



Identifier	Definition	Severity
LOG-03	Checks Effects Interactions	Medium 🟡

Below mentioned functions should be verified for Checks Effects Interactions:

`sendERC20Transaction()` – Make sure the called contract adheres to the ERC20 standard and does not have any malicious 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 Checks Effects Interactions pattern when handing over the flow to an external entity and/or guard functions against re-entrancy attacks.

RESOLUTION

Ekta development team has whitelist check to verify ERC20 receiver address.



Identifier	Definition	Severity
COD-04	Missing or inaccurate error messages	Minor 

Below mentioned functions, modifiers have missing or inaccurate error messages:

```

addUserWallet()
removeUserWallet()
validRequirement()
onlyWallet()
ownerDoesNotExist()
ownerExists()
transactionExists()
confirmed()
notConfirmed()
notExecuted()
notNull()
initialize()

```

RECOMMENDATION

Provide accurate information strings for require related errors.

RESOLUTION

Ekta development team has added error messages for require statements.



Identifier	Definition	Severity
COD-05	Note regarding keccak256 secure hashing	


Note that the keccak256 function is not collision-resistant, and therefore there is a possibility of two different messages producing the same hash. Generating strong random input data, and properly securing and managing keys is recommended for fortification of keccak256.

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

ACKNOWLEDGEMENT

Ekta development team has acknowledged to this note regarding use of keccak256 secure hashing.



Identifier	Definition	Severity
COD-07	Possible DoS attack	Minor 

In the `executeTransaction` function, the `txn.executed` state variable is set to `true` before the actual external call is made. This could lead to potential denial of service (DoS) attacks if the transaction repeatedly fails.

RECOMMENDATION

Consider setting `executed` state after a successful call, and in case of a failure, revert the transaction.

ACKNOWLEDGEMENT

Ekta development team has commented that `txn.execute` is updated beforehand to avoid re-entrancy, and the serialized nature of txns is not enforced to handle DOS.



Identifier	Definition	Severity
COD-10	Third Party Dependencies	Unknown ●

Smart contract is interacting with third party protocols e.g., Decentralized Applications, Open Zeppelin tools, such as MerkleProof contract. The scope of the audit treats third party entities as black boxes and assumes their functional correctness. However, in the real world, third parties can be compromised, and exploited. Moreover, upgrades in third parties 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

Ekta development team will inspect third party dependencies periodically.



Identifier	Definition	Severity
COD-11	Lack of direct access control	Minor 

Mentioned functions can be called by anyone, they rely on the multi-signature mechanism of contracts to enforce access control. Transactions created by these functions will not be executed without the necessary confirmations from the owners, which provides indirect access restriction. However, it is still recommended to provide direct access control:

```
createWhitelistTransaction()
sendEktaTransaction()
sendERC20Transaction()
createAddOwnerTransaction()
createRemoveOwnerTransaction()
createRequirementTransaction()
```

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

ACKNOWLEDGEMENT

Ekta development team has acknowledged that aforementioned functions rely on the multi-signature mechanism of contracts to enforce access control, hence, access control is restricted.



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. Some functions are provided adequate events to track state changes. However, function like `changeWhitelist()` can use events to track changes.

RECOMMENDATION

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

RESOLUTION

Ekta development team has added events to track state changes.



Identifier	Definition	Severity
VOL-01	Commented-out code	Minor ●

CreateUserWallet() is commented out in Registry contract.


RECOMMENDATION

Remove commented-out code if it's not required.

RESOLUTION

Ekta development team has removed redundant code-lines.



Identifier	Definition	Severity
VOL-02	Use of <code>.call</code>	Minor 

`.call` in the `external_call()` does not have a gas limit, which could potentially make contract vulnerable to attacks if the called contract consumes all the available gas.

TERFI
CONFIDENTIALINTERFI
CONFIDENTIAL

RECOMMENDATION

Use `transfer` instead of `.call`.

ACKNOWLEDGEMENT

Use of `transfer` and `call` has their own pros and cons. Ekta development team will keep `.call` in the `external_call()`.



Identifier	Definition	Severity
COM-01	Floating compiler status	

Compilers are set to ^0.8.12

TERFI
CONFIDENTIALINTERFI
CONFIDENTIAL

RECOMMENDATION

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

RESOLUTION

Ekta development team has fixed floating pragma in all contracts under scope.



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