

# SMART CONTRACT AUDIT



interfinetwork



hello@interfi.network



<https://interfi.network>

PREPARED FOR

**STORAGE CHAIN STAKING CONTRACT**



# INTRODUCTION

Auditing Firm	InterFi Network
Client Firm	Storage Chain
Methodology	Automated Analysis, Manual Code Review
Language	Solidity
Contract	
Blockchain	Ethereum Chain
Centralization	Active Ownership
Commit	6e9c04f31ea842716be818f6ca5522df3a1d325
Website	<a href="https://storagechain.io/">https://storagechain.io/</a>
Telegram	<a href="https://t.me/s/storagechain">https://t.me/s/storagechain</a>
X (Twitter)	<a href="https://twitter.com/StorageChain">https://twitter.com/StorageChain</a>
Report Date	November 04, 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 <span style="color: red;">●</span>	Major <span style="color: orange;">●</span>	Medium <span style="color: yellow;">●</span>	Minor <span style="color: green;">●</span>	Unknown <span style="color: brown;">●</span>
Open	0	0	0	1	1
Acknowledged	0	1	0	3	1
Resolved	0	0	1	1	0
Important Functions	Add Node, Delete Node				
Noteworthy Privileges	Deposit, Feed Node Record				

 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
MANUAL REVIEW.....	11
DISCLAIMERS .....	22
ABOUT INTERFI NETWORK .....	25

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 Storage Chain 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:

- StakingContract.sol

 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	
Contract Name	StakingContract
Compiler Version	0.8.17
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"><li>○ Token Supply Manipulation</li><li>○ Access Control and Authorization</li><li>○ Assets Manipulation</li><li>○ Ownership Control</li><li>○ Liquidity Access</li><li>○ Stop and Pause Trading</li><li>○ Ownable Library Verification</li></ul>
----------------------	---



## 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

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:

Risk Type	Definition
Critical <span style="color: red;">●</span>	These risks pose immediate and severe threats, such as asset theft, data manipulation, or complete loss of contract functionality. They are often easy to exploit and can lead to significant, irreparable damage. Immediate fix is required.
Major <span style="color: orange;">●</span>	These risks can significantly impact code performance and security, and they may indirectly lead to asset theft and data loss. They can allow unauthorized access or manipulation of sensitive functions if exploited. Fixing these risks are important.
Medium <span style="color: yellow;">●</span>	These risks may create attack vectors under certain conditions. They may enable minor unauthorized actions or lead to inefficiencies that can be exploited indirectly to escalate privileges or impact functionality over time.
Minor <span style="color: green;">●</span>	These risks may include inefficiencies, lack of optimizations, code-style violations. These should be addressed to enhance overall code quality and maintainability.
Unknown <span style="color: brown;">●</span>	These risks pose uncertain severity to the contract or those who interact with it. Immediate fix is required to mitigate risk uncertainty.

All statuses which are identified in the audit report are categorized here:

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

```

| **StakingContract** | Implementation | ReentrancyGuard |||
| L | <Constructor> | Public ! |  | NO ! |
| L | deposit | External ! |  | onlyOwner |
| L | FeedNodeRecord | External ! |  | onlyOwner |
| L | addNode | External ! |  | nonReentrant |
| L | deleteNode | External ! |  | nonReentrant |
| L | checkContractBalance | External ! | NO ! |
| L | totalStakedStorCoins | External ! | NO ! |
| L | nodeExists | External ! | NO ! |
| L | totalStakeOfUser | External ! | NO ! |
| L | getNodeIds | External ! | NO ! |
| L | getStakeAmountOfNode | External ! | NO ! |
| L | hasStaked | External ! | NO ! |
| L | renounceOwnership | Public ! |  | onlyOwner |
| L | owner | External ! | NO ! |
| L | <Receive Ether> | External ! |  | NO ! |

```

INTERFI  
CONFIDENTIAL



## MANUAL REVIEW

Identifier	Definition	Severity
CEN-01	Centralized privileges	Major 🟡

Important onlYowner centralized privileges are listed below:

`deposit()`

`FeedNodeRecord()`

`renounceOwnership()`

`FeedNodeRecord()` allows for mass updating of records and can potentially lead to misuse if not properly protected. It updates a lot of critical state variables and it could be misused if owner's address is compromised.

### RECOMMENDATION

Securing private keys or access credentials of deployers, contract owners, operators, and other roles with privileged access is crucial to prevent single points of failure that can compromise contract security.


Use of multi-signature wallets is recommended – These wallets require multiple authorizations to execute sensitive contract functions, reducing the risk associated with single-party control.

Use of decentralized governance model is recommended – This model allows token holders and stakeholders to actively participate in decision-making, such as contract upgrades and parameter adjustments, enhancing overall security and resilience.

### ACKNOWLEDGEMENT

StorageChain team argued that centralized and controlled privileges are used as required. StorageChain team will implement Gnosis Multi-sig wallet upon wrapper contract deployment.



Identifier	Definition	Severity
LOG-01	Inadequate input validation	Below 

There are no checks on the lengths of arrays provided to the `feedNodeRecord()`. Inconsistent array lengths can lead to unexpected behavior or runtime errors.

## RECOMMENDATION


Add require check to ensure `_userAddresses`, `_nodeIds`, and `_stakedAmounts` have same length.

```
require(_userAddresses.length == _nodeIds.length && _nodeIds.length == _stakedAmounts.length,  
"Array lengths mismatch");
```

## NOTE

Updated review Below .



Identifier	Definition	Severity
LOG-01-01	Incorrect Array Length Validation in <code>feedNodeRecord()</code>	Minor 

Array length validation in `feedNodeRecord()` checks if all arrays are non-empty but allows the possibility of mismatched lengths due to a logical error in this line:

```
if(_userAddresses.length != _nodeIds.length && _nodeIds.length != _stakedAmounts.length){
    revert lengthDontMatch(_userAddresses.length,_nodeIds.length,_stakedAmounts.length);
}
```

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

## RECOMMENDATION

Replace the logical AND (`&&`) with OR (`||`) in the condition to ensure that all provided arrays have exactly the same length.

## ACKNOWLEDGEMENT

StorageChain team commented that use of AND (`&&`) in the condition follows intended logic. Code is kept as-is.



Identifier	Definition	Severity
LOG-02	Potential front-running	Minor ●

Front-running occurs when a transaction is observed in the mempool and another transaction is submitted with a higher gas price to be mined first, exploiting the knowledge of the pending transactions. Key areas vulnerable in your contract include:

Node and Treasury Transactions: Since these involve value transfers, a front-runner could manipulate the order of transactions to their advantage.

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


## RECOMMENDATION

Use commit-reveal schemes, timelocks, or by ensuring that transaction order does not impact the fairness of the contract to protect against front-running.

## ACKNOWLEDGEMENT

Front-running is not avoidable on public blockchains. StorageChain team commented that, most EVM chains are prone to some sort of front-running and external manipulation.



Identifier	Definition	Severity
LOG-03	Re-entrancy	Below 
LOG-04	Checks-Effects-Interactions	

Re-entrancy attacks happen when external contracts are called, which can then call back into the calling contract before the first execution finishes.

Mentioned functions can be vulnerable to re-entrancy attacks as they call an external address while the contract state is still mutable:

```
addNode()  
deleteNode()
```

Vulnerable to re-entrancy because it calls external contract (`payable(msg.sender).call{value: stakedAmount}("")`) before updating internal state (`stake.isNodeAlive` and `stake.stakedAmount`).

## RECOMMENDATION

Use Checks-Effects-Interactions (CEI) pattern when transferring control to external entities. This design pattern ensures that all state changes are completed before external interactions occur. Additionally, implement re-entrancy guard to block recursive calls from external contracts.

## NOTE

Updated review Below .



Identifier	Definition	Severity
LOG-04-01	Checks-Effects-Interactions in deleteNode()	Medium 🟡

deleteNode()

Function uses nonReentrant modifier, which can be effective in preventing re-entrancy attacks. However, the function still performs the external call to transfer Ether using `.call{value: x}()`, which can be exploited. `stakeInfo[msg.sender][_nodeId].stakedAmount` value is reset after making the external call.

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

## RECOMMENDATION

Reorder the operations in `deleteNode()` to follow the Checks-Effects-Interactions pattern strictly. First adjust the state variables, then transfer Ether.

## RESOLUTION

StorageChain team has reordered operations in `deleteNode()` to follow the Checks-Effects-Interactions pattern.





Identifier	Definition	Severity
COD-02	Timestamp dependence	Minor ●

Smart contract uses timestamps for reward transfer timing. Be aware that the timestamp of the block can be manipulated by miners. Since miners can slightly adjust the timestamp, they may influence contract outcomes to their advantage.

## RECOMMENDATION

Avoid relying solely on timestamp of the block for critical contract functions. Follow 15 seconds rule, and scale time dependent events accordingly.

## ACKNOWLEDGEMENT

StorageChain team argued that contract logic depends on use of timestamps for reward transfer timing.



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

Smart contract is interacting with third party protocols e.g., DEX routers, reward token contract, stake token contract, web3 applications, *OpenZeppelin* upgradeable and ERC20 libraries. 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.


## RECOMMENDATION

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

## ACKNOWLEDGEMENT

StorageChain team will inspect third party dependencies regularly, and push upgrades whenever required.



Identifier	Definition	Severity
COD-11	Gas limit vulnerabilities in loops	Minor 

FeedNodeRecord ( ) function iterates over arrays without bounds, which can lead to gas limit issues if the arrays are large enough. This can allow a malicious user to craft transactions that consume all the gas provided by a transaction, effectively making the function fail.

```
for (uint i=0; i < _userAddress.length; i++ )
```

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

## RECOMMENDATION

Set a maximum bound on array sizes or use more efficient data management system to handle such operations without iteration over potentially large arrays.




Identifier	Definition	Severity
COD-13	Note regarding flash loan attack	Unknown 🟤

Flash loans are typically exploited in contracts that rely on external state, such as the prices of tokens or the balances of accounts, which can be manipulated temporarily within a single transaction. Vulnerabilities may arise if smart contract relies on external state or token balances that could be temporarily inflated. For instance, acquiring a large number of tokens to temporarily manipulate the price or liquidity can mislead the contract into issuing more tokens than appropriate.

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
COM-01	Floating pragma	Minor 

Compiler is set to ^0.8.17

INTERFI  
CONFIDENTIAL

INTERFI  
CONFIDENTIAL

## RECOMMENDATION

Pragma should be fixed to stable compiler version. Fixing pragma ensures compatibility and prevents the contract from being compiled with incompatible compiler versions.

## 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](mailto: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