InterFi NETWORK

SMART CONTRACT AUDIT

interfinetwork

hello@interfi.network

https://interfi.network

PREPARED FOR

SOCIALPAL STAKE CONTRACT

INTRODUCTION

Auditing Firm	InterFi Network
Client Firm	SocialPal
Methodology	Automated Analysis, Manual Code Review
Language	Solidity
Contract	
Blockchain	
Centralization	Active ownership
Commit	6cfcef4d69cb5ccf578342c8d7735e29ffed2200
Website	
Report Date	April 24, 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	0	0	0	2	0
Acknowledged	0	1	0	2	1
Resolved	0	1	1	3	0
Important Privileges	Pause, Fund, Update Tier Reward Rate, Update Tier State				
Important Functions	Fund, Deposit Locked, Withdraw Locked				

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 SocialPal 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:

- Staking.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	
Contract Name	Staking
Compiler Version	0.8.24
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	○ 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.

ℹ 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 issues are exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed immediately.
Major 🟠	These issues are exploited with a simple trick, they carry an elevated risk of data loss and manipulation, which leads to major risk. They should be fixed immediately.
Medium 🟡	These issues should be fixed, as they carry an inherent risk of straight-forward exploits, and hacks which may or may not impact the smart contract execution. Minor-risk re-entrancy-related vulnerabilities should be fixed to deter exploits.
Minor 🟢	These issues 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 issues pose uncertain risks 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.

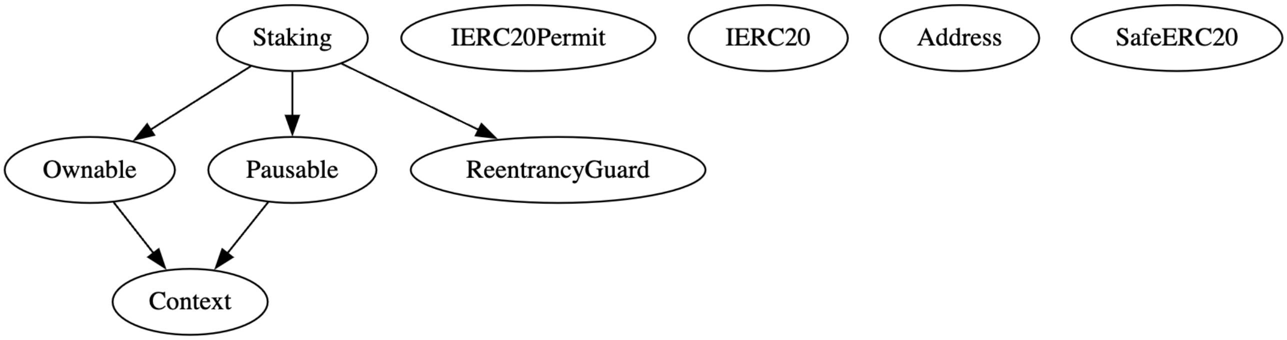
ℹ Understand the project's initial asset distribution. Assets in the liquidity pair should be locked. Assets outside the liquidity pair should be locked with a release schedule.

AUTOMATED ANALYSIS

Symbol	Definition
🛑	Function modifies state
💵	Function is payable
🔒	Function is internal
🔐	Function is private
❗	Function is important

```
| **Context** | Implementation |  |||
| └ | _msgSender | Internal 🔒 |   | |
| └ | _msgData | Internal 🔒 |   | |
| └ | _contextSuffixLength | Internal 🔒 |   | |
||||||
| **Ownable** | Implementation | Context |||
| └ | <Constructor> | Public ❗ | 🛑  |NO❗ |
| └ | owner | Public ❗ |   |NO❗ |
| └ | _checkOwner | Internal 🔒 |   | |
| └ | renounceOwnership | Public ❗ | 🛑  | onlyOwner |
| └ | transferOwnership | Public ❗ | 🛑  | onlyOwner |
| └ | _transferOwnership | Internal 🔒 | 🛑  | |
||||||
| **IERC20Permit** | Interface |  |||
| └ | permit | External ❗ | 🛑  |NO❗ |
| └ | nonces | External ❗ |   |NO❗ |
| └ | DOMAIN_SEPARATOR | External ❗ |   |NO❗ |
||||||
| **IERC20** | Interface |  |||
| └ | totalSupply | External ❗️ |   |NO❗️ |
| └ | balanceOf | External ❗️ |   |NO❗️ |
| └ | transfer | External ❗️ | 🛑  |NO❗️ |
| └ | allowance | External ❗️ |   |NO❗️ |
| └ | approve | External ❗️ | 🛑  |NO❗️ |
| └ | transferFrom | External ❗️ | 🛑  |NO❗️ |
||||||
| **Address** | Library |  |||
| └ | sendValue | Internal 🔒 | 🛑  | |
| └ | functionCall | Internal 🔒 | 🛑  | |
| └ | functionCallWithValue | Internal 🔒 | 🛑  | |
| └ | functionStaticCall | Internal 🔒 |   | |
| └ | functionDelegateCall | Internal 🔒 | 🛑  | |
| └ | verifyCallResultFromTarget | Internal 🔒 |   | |
| └ | verifyCallResult | Internal 🔒 |   | |
| └ | _revert | Private 🔐 |   | |
||||||
| **SafeERC20** | Library |  |||
| └ | safeTransfer | Internal 🔒 | 🛑  | |
| └ | safeTransferFrom | Internal 🔒 | 🛑  | |
| └ | safeIncreaseAllowance | Internal 🔒 | 🛑  | |
| └ | safeDecreaseAllowance | Internal 🔒 | 🛑  | |
| └ | forceApprove | Internal 🔒 | 🛑  | |
| └ | _callOptionalReturn | Private 🔐 | 🛑  | |
| └ | _callOptionalReturnBool | Private 🔐 | 🛑  | |
||||||
| **Pausable** | Implementation | Context |||
| └ | <Constructor> | Public ❗️ | 🛑  |NO❗️ |
| └ | paused | Public ❗ |   |NO❗ |
| └ | _requireNotPaused | Internal 🔒 |   | |
| └ | _requirePaused | Internal 🔒 |   | |
| └ | _pause | Internal 🔒 | 🛑  | whenNotPaused |
| └ | _unpause | Internal 🔒 | 🛑  | whenPaused |
||||||
| **ReentrancyGuard** | Implementation |  |||
| └ | <Constructor> | Public ❗ | 🛑  |NO❗ |
| └ | _nonReentrantBefore | Private 🔐 | 🛑  | |
| └ | _nonReentrantAfter | Private 🔐 | 🛑  | |
| └ | _reentrancyGuardEntered | Internal 🔒 |   | |
||||||
| **Staking** | Implementation | Ownable, ReentrancyGuard, Pausable |||
| └ | <Constructor> | Public ❗ | 🛑  | Ownable |
| └ | fund | External ❗ | 🛑  | nonReentrant onlyOwner whenNotPaused |
| └ | depositLocked | External ❗ | 🛑  | nonReentrant whenNotPaused |
| └ | withdrawLocked | External ❗ | 🛑  | nonReentrant whenNotPaused |
| └ | getLockedUserData | Public ❗ |   |NO❗ |
| └ | updateTierRewardRate | External ❗ | 🛑  | onlyOwner whenNotPaused |
| └ | updateTierState | External ❗ | 🛑  | onlyOwner whenNotPaused |
| └ | pause | External ❗ | 🛑  | onlyOwner |
| └ | unpause | External ❗ | 🛑  | onlyOwner |
```

INHERITANCE GRAPH



‘

MANUAL REVIEW

Identifier	Definition	Severity
CEN-01	Centralized privileges	Major ●

Important `onlyOwner` centralized privileges are listed below:

```
renounceOwnership
transferOwnership
fund
updateTierRewardRate
updateTierState
pause
unpause
```

RECOMMENDATION

Deployers, contract owners, operators', access controlled, 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.

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

Social Pal team argued that centralized privileged are required by the design.

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

When creating a new deposit or executing a withdrawal, transaction can be seen by miners before it's included in a block, allowing malicious actors to front-run transactions.

```
fund
depositLocked
withdrawLocked
updateTierState
```

RECOMMENDATION

Use commit-reveal scheme to hide transactions until completed. This should lower front-running risk.

ACKNOWLEDGEMENT

Social Pal team has acknowledged this finding. Additionally, front-running is an unavoidable issue on EVM-based chains.

Identifier	Definition
LOG-03	Re-entrancy

Re-entrancy guard is used to prevent re-entrant calls. Below mentioned functions are used with re-entrancy guard:

```
fund
depositLocked
withdrawLocked
```

Identifier	Definition	Severity
COD-02	Potential timestamp dependence	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.

Use of `block.timestamp` for calculating rewards (`calculateRewards`) may introduce vulnerabilities related miner manipulation.

RECOMMENDATION

Significant manipulation of a block is difficult, however, be aware of potential miner manipulation when using of `block.timestamp`.

RESOLUTION

Timestamp of a block is not used to generate random numbers. Miner manipulation should be minimal.

Identifier	Definition	Severity
COD-03	Precision loss in reward calculation	Medium 🟡

Smart contract uses integer division for calculating reward rates extensively, which may lead to significant precision loss.

For example, in mentioned line, formula can be rearranged to minimize early division.

```
rewards = ((stakedBalance[staker] * (rewardsPerc  / 10 ** 18)  ) / 100);
```

Rearranged formula.

```
rewards = (stakedBalance[staker] * rewardsPerc) / (10 ** 18 * 100);
```

RECOMMENDATION

Use higher precision for intermediate calculations or adjust formula to minimize early division.

RESOLUTION

Social Pal team has amended integer division for calculating reward rates. Integer division is carried out initially, followed by calculation of decimals.

Identifier	Definition	Severity
COD-03-01	Precision loss in `SECONDS_IN_YEAR`	Minor 🟢

Reward rate division by `SECONDS_IN_YEAR` can lead to precision loss in integer division.

RECOMMENDATION

Use higher precision for intermediate calculations or adjust formula to minimize early division.

Identifier	Definition	Severity
COD-05	Non-conforming logic in withdrawal	Major ●

Ability to withdraw funds when tiers are disabled may allow users to bypass the lock period.

Disabling tier allows immediate withdrawals regardless of the lock period. This may not be intended design as it can be exploited to bypass lock periods.

RECOMMENDATION

Add a specific requirement that must be met for early withdrawal when a tier is disabled, e.g.,

```
require(block.timestamp >= TierData[_tier].userData[msg.sender].unlockTime ||
(Tiers[_tier].enabled == false && conditionForEarlyWithdrawal), "NOT_CLAIMABLE");
```

RESOLUTION

Social Pal team has amended withdrawal logic.

Identifier	Definition	Severity
COD-07	Inadequate checks	Minor 🟢

`depositLocked` function deducts rewards from `rewardFunds` without adequate checks beyond initial setup.

RECOMMENDATION

Use additional checks to monitor and control the distribution of rewards, and make sure that `rewardFunds` does not get empty unexpectedly.

Identifier	Definition	Severity
COD-10	Direct and indirect dependencies	Unknown ●
COD-11	`stakingToken` safety and trustworthiness	

Smart contract is interacting with third party protocols e.g., DEX router, staking token contract, external contracts, web3 applications, *OpenZeppelin* libraries, and various Solidity 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.

RECOMMENDATION

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

ACKNOWLEDGEMENT

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

Identifier	Definition	Severity
VOL-01	Use of `delegatecall`	Minor 🟢

`delegatecall` is present in the smart contract.

RECOMMENDATION

Verify the user input and do not allow contract to perform `delegatecall` calls to untrusted contracts. Use of `delegatecall` in the contract is not recommended, as managing the storage layout in multiple contracts during logic update can be disruptive.

ACKNOWLEDGEMENT

SocialPal team has acknowledged this finding, and argued that OpenZeppelin libraries are trusted, widely used, and audited.

Identifier	Definition	Severity
VOL-02	Assembly code	Minor 🟢

Inline assembly is a way to access the Ethereum Virtual Machine (EVM) at low level. This bypasses several important safety features and checks of Solidity. Moreover, automated and manual checks are not confidently possible for inline assembly codes.

RECOMMENDATION

Use high level Solidity constructs instead of `assembly`.

RESOLUTION

SocialPal team has commented that – main assembly code is used for gas savings in byte manipulation and was written by *Consensys*, and is considered safe.

Identifier	Definition	Severity
COM-01	Floating pragma	Minor 🟢
COM-02	Multiple pragma	

Compiler is set to `^0.8.24`

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 version `0.8.24.`

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