

SMART CONTRACT SECURITY AUDIT OF COW STAKE



SMART CONTRACT AUDIT | SOLIDITY DEVELOPMENT & TESTING | PROJECT EVALUATION

RELENTLESSLY SECURING THE PUBLIC BLOCKCHAIN



Summary

Auditing Firm InterFi Network

Client Firm Cow Stake

Architecture InterFi "Echelon" Auditing Standard

Language Solidity

Mandatory Audit Check Static, Software, Auto Intelligent & Manual Analysis

Final Report Date January 19, 2022

<u>Audit Summary</u>

InterFi team has performed a line-by-line manual analysis and automated review of the smart contracts. Smart contracts were analyzed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks. According to the smart contract audit:

- Cow Stake's smart contract source codes have LOW RISK SEVERITY
- Cow Stake's smart contracts have an ACTIVE OWNERSHIP
- Cow Stake's contracts under scope:
 - CowBondDepository.sol
 - CowStaking.sol
 - Scow.sol
 - ScowCalculator.sol

For the detailed understanding of risk severity, source code vulnerability, and functional test, kindly refer to the audit.

✓ Verify the authenticity of this report on InterFi's GitHub: https://github.com/interfinetwork



Table Of Contents

Project Information

Overview	4
InterFi "Echelon" Audit Standard	
Audit Scope & Methodology	6
InterFi's Risk Classification	8
Smart Contract Risk Assessment	
Static Analysis	9
Inheritance Graph	15
Manual Analysis	17
SWC Attacks	19
Risk Status & Radar Chart	21
Report Summary	
Auditor's Verdict	22
<u>Legal Advisory</u>	
Important Disclaimer	23
About InterFi Network	24



Project Overview

InterFi was consulted by Cow Stake to conduct the smart contract security audit of their solidity source codes.

About Cow Stake

CowStake is a defi 2.0 project. The initial intention of the project was to solve the problem of spamming of aircoin in the defi 1.0 project. To this end, we designed a set of ingenious economic models. Our goal is to keep our supporters profitable in the DEFI world.

Project	Cow Stake
Blockchain	Binance Smart Chain
Language	Solidity
Contract	Solidity multi-file format Smart Contract
Website	https://cowstake.money/ Security Audit
Telegram	https://t.me/CowStake
Twitter	https://twitter.com/cow_stake
Medium	https://medium.com/@cowstake4.4
Discord	https://discord.com/invite/KjmWPdS8C9



Project Logo



Solidity Source Codes Under Scope

https://github.com/milkCowOrg/test_contract

- CowBondDepository.sol
- CowStaking.sol
- Scow.sol
- ScowCalculator.sol



Smart Contract

Solidity Source Code On InterFi GitHub

https://github.com/interfinetwork/audited-codes/blob/main/CowStake.sol

SHA-1 Hash

Solidity source codes are audited at hash #bdf201fb58b5a8e8e85a348f82b5f014e82682cf



Audit Scope & Methodology

The scope of this report is to audit the smart contract source codes of Cow Stake. InterFi has scanned the contracts and reviewed the project for common vulnerabilities, exploits, hacks, and back-doors. Below is the list of commonly known smart contract vulnerabilities, exploits, and hacks:

Category

- Re-entrancy
- Unhandled Exceptions
- Transaction Order Dependency
- Integer Overflow
- Unrestricted Action
- Incorrect Inheritance Order
- Typographical Errors
- Requirement Violation
- Ownership Takeover
- Gas Limit and Loops
- Deployment Consistency
- Repository Consistency
- Data Consistency
- Token Supply Manipulation
- Access Control and Authorization
- Operations Trail and Event Generation
- Assets Manipulation
- Liquidity Access

Smart Contract Vulnerabilities

Source Code Review

Functional Assessment



InterFi's Echelon Audit Standard

The aim of InterFi's "Echelon" standard is to analyze the smart contract and identify the vulnerabilities and the hacks in the smart contract. Mentioned are the steps used by ECHELON-1 to assess the smart contract:

- 1. Solidity smart contract source code reviewal:
 - Review of the specifications, sources, and instructions provided to InterFi to make sure we understand the size, scope, and functionality of the smart contract.
 - Manual review of code, which is the process of reading source code line-byline to identify potential vulnerabilities.
- 2. Static, Manual, and Software analysis:
 - Test coverage analysis, which is the process of determining whether the test cases are covering the code and how much code is exercised when we run those test cases.
 - Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
- 3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
- 4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts

Automated 3P frameworks used to assess the smart contract vulnerabilities

- Slither
- Consensys MythX, Mythril
- SWC Registry
- Solidity Coverage
- Open Zeppelin Code Analyzer
- Solidity Code Complier



InterFi's Risk Classification

Smart contracts are generally designed to manipulate and hold funds denominated in ETH/BNB. This makes them very tempting attack targets, as a successful attack may allow the attacker to directly steal funds from the contract. Below are the typical risk levels of a smart contract:

Vulnerable: A contract is vulnerable if it has been flagged by a static analysis tool as such. As we will see later, this means that some contracts may be vulnerable because of a false-positive.

Exploitable: A contract is exploitable if it is vulnerable and the vulnerability could be exploited by an external attacker. For example, if the "vulnerability" flagged by a tool is in a function which requires to own the contract, it would be vulnerable but not exploitable.

Exploited: A contract is exploited if it received a transaction on the main network which triggered one of its vulnerabilities. Therefore, a contract can be vulnerable or even exploitable without having been exploited.

	of fight Colleges
Risk severity	Meaning Security Audit
! Critical	This level vulnerabilities could be exploited easily, and can lead to asset loss,
	data loss, asset, or data manipulation. They should be fixed right away.
! High	This level vulnerabilities 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	This level vulnerabilities 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	This level vulnerabilities can be ignored. They are code style violations, and
	informational statements in the code. They may not affect the smart contract
	execution



Smart Contract - Static Analysis

Meaning
Function can be modified
Function is payable
Function is locked
Function can be accessed
Important functionality

```
| **<mark>ScowCalculator</mark>** | Implementation | Ownable, IScowCalculator, AbstractScow |||
 L | <Constructor> | Public ! | ● |NO! |
| L | rebaseForCal | Public ! | OnlysCowContract | |
| L | getTotalAmount | External ! | NO! |
| L | addStaker | Internal 🗎 | 🛑 | |
 └ | updateUserInfo | Public ! | ● |NO! |
| └ | _updateUserInfo | Internal 🗎 | ● | |
| └ | updateLevelGons | Internal 🗎 | ● | |
| └ | _updateGons | Internal 🗎 | 🔴 | |
| L | getLevelValue | Public ! | NO! |
| └ | _updateUserLevel | Internal 🗎 | ● | lock |
| L | balanceOf | Public ! | | initialized |
| L | gonsForBalance | Public ! | | initialized |
| L | balanceForGons | Public ! | | initialized |
| L | transferFrom | Public ! | OnlysCowContract |
| L | getGonsPerFragment | Private 🚆 | | |
| └ | _perFragment | Private 🔐 | | |
| L | _getUserCurrentLevel | Internal 🗎 |
| L | getMaxLevel | Public ! | NO! |
| L | getInfoList | Public ! | NO! |
| L | getRebaseInfoList | Public ! | NO! |
| L | getLevelSupply | Public ! | NO! |
| L | getUserCountList | Public ! | NO! |
| L | setContract | External ! | 

| onlyManager |
| L | setStakeLevelInfo | Public ! | • | onlyManager |
| L | setLevelValue | External ! | • | onlyManager |
| L | addBeneficiary | Public ! | 🔎 | onlyManager |
| L | setRouter | Public ! | • | onlyManager |
| └ | listContains | Internal 🗎 | | |
```



```
| **IOwnable** | Interface | |||
 L | manager | External ! | NO! |
| └ | renounceManagement | External ! | ● |NO! |
| L | pushManagement | External ! | • | NO! |
| L | pullManagement | External ! | 🔴 | NO! |
111111
| **<mark>Ownable</mark>** | Implementation | IOwnable |||
| └ | <Constructor> | Public ! | ● |NO! |
| L | manager | Public ! | NO! |
| L | renounceManagement | Public ! | OnlyManager |
| └ | <mark>pushManagement</mark> | Public ! | ● | onlyManager |
| L | pullManagement | Public ! | • |NO! |
| **<mark>FullMath</mark>** | Library |  |||
| L | fullMul | Private 🔐 | | |
| <sup>L</sup> | fullDiv | Private 🗎 |
                               III
| L | mulDiv | Internal 🗎 |
| **FixedPoint** | Library |
                             - 111
| L | decode | Internal 🗎 | | |
| L | decode112with18 | Internal 🗎 |
                                       | L | fraction | Internal e | | |
| **IERC20** | Interface | |||
| L | totalSupply | External ! | NO! |
| L | balanceOf | External ! |
                                  |NO ! |
| L | transfer | External ! | 🛑 |NO! | | | |
| L | allowance | External ! |
| L | approve | External ! | P | NO! |
| L | transferFrom | External ! | • |NO! |
| | | | | | | | |
| **<mark>SafeMath</mark>** | Library | |||
| L | tryAdd | Internal 🗎 |
| L | trySub | Internal 🗎 |
| L | tryMul | Internal 🔒 |
| L | tryDiv | Internal 🗎 |
| L | tryMod | Internal 🔒 |
| L | add | Internal 🗎 |
| <sup>L</sup> | sub | Internal 🔒 |
| L | mul | Internal 🗎 |
| <sup>L</sup> | div | Internal 🔒 |
| L | mod | Internal 🗎 |
| <sup>L</sup> | sub | Internal 🔒 |
| L | div | Internal 🗎 |
| <sup>L</sup> | mod | Internal 🗎 |
111111
| **Address** | Library | |||
| L | isContract | Internal 🗎 | | |
| L | sendValue | Internal 🗎 | 🔴 | |
| └ | functionCall | Internal 🏻 | 🔎 | |
```



```
| └ | functionCall | Internal 🔒 | 🛑 | |
 └ | functionCallWithValue | Internal 🗎 | 🕌 | |
| L | functionCallWithValue | Internal 🗎 | 🔴 | | |
| L | functionStaticCall | Internal 🗎 | | | |
| └ | functionDelegateCall | Internal 🏻 | ● | |
| L | functionDelegateCall | Internal 🗎 | 🛑 | |
| L | _verifyCallResult | Private 🔒 | | | |
IIIIIII
| **<mark>SafeERC20</mark>** | Library | |||
| └ | safeTransferFrom | Internal 🍙 | 🔴 | |
| └ | safeApprove | Internal 🗎 | 🔴 | |
| └ | safeIncreaseAllowance | Internal 🏻 | ● | |
| L | safeDecreaseAllowance | Internal 🗎 | 🛑 | |
IIIIIII
| **ITreasury** | Interface | |||
| └ | mintRewards | External ! | ● |NO! |
| L | deposit | External ! | 🔎 |NO! |
| L | valueOf | External ! | NO! |
| **IStakingHelper** | Interface | |||
| L | stake | External ! | ● |NO! |
| **IStaking** | Interface | |||
| L | stake | External ! | 🔴 |NO! |
IIIIIII
| **IBondCalculator** | Interface | |||
| L | valuation | External ! | NO! |
| L | markdown | External ! | NO! |
| **ICOW20** | Interface | |||
| L | decimals | External ! | NO! |
| L | balanceOf | External ! | NO! |
| L | burnFrom | External ! | O | NO! |
IIIIIII
| **IBondPriceHelper** | Interface | ||| | | |
| L | calculateBondPrice | External ! | • | NO! |
| L | currentBondPrice | External ! | NO! |
| L | getCowPrice | External ! | NO! |
| **IRouter** | Interface | |||
| L | inviteUser | External ! | O | NO! |
| | | | | | | | |
| **CowBondDepository** | Implementation | Ownable |||
| L | <Constructor> | Public ! | • | NO! |
| └ | initializeBondTerms | External ! | ● | onlyManager |
| └ | setBondTerms | External ! | ● | onlyManager |
| └ | setStaking | External ! | ● | onlyManager |
```



```
| L | setRouter | Public ! | 🔴 | onlyManager |
 └ | deposit | External ! | ● |NO! |
| L | redeem | External ! | P | NO! |
| └ | stakeOrSend | Internal 🍙 | 🔴 | |
 └ | decayDebt | Internal 🔒 | 🔴 | |
 L | maxPayout | Public ! | NO!
| L | payoutFor | Public ! |
                          |N0 ! |
 L | bondPrice | Public ! | NO! |
| └ | _updateBondPrice | Internal 🗎 | 🛑 | |
| L | bondPriceInUSD | Public ! | NO! |
| L | debtRatio | Public ! | NO! |
 L | standardizedDebtRatio | External ! |
                                     |N0 ! |
| L | currentDebt | Public ! | NO! | | | | |
| L | debtDecay | Public ! | NO! |
| L | percentVestedFor | Public ! | NO! |
| L | pendingPayoutFor | External ! | NO! |
| L | recoverLostToken | External ! | • | NO! |
| L | setDiscount | External ! | 📦 | onlyManager |
| └ | setFund | External ! | ● | onlyManager |
| **IsCOW** | Interface | |||
| L | circulatingSupply | External ! | NO! |
| L | balanceOf | External ! | NO! |
| L | gonsForBalance | External ! | NO! |
| L | balanceForGons | External ! | NO! |
| **EnumerableSet** | Library | |||
| L | _remove | Private 🔐 | 🛑 | |
| └ | _contains | Private 🔒 | | |
| L | _length | Private 🔐 | | |
| L | add | Internal | | | | | |
| L | remove | Internal 🗎 | 🛑 | |
| L | contains | Internal 🔒 | | |
| <sup>L</sup> | length | Internal <sup>□</sup> | | |
| L | add | Internal 🗎 | 🛑 | |
| L | remove | Internal 🗎 | 🛑 | |
| L | contains | Internal 🗎 | | |
| <sup>L</sup> | at | Internal <sup>@</sup> | | |
| L | add | Internal = | e | |
| L | remove | Internal 🗎 | 🛑 | |
| └ | contains | Internal 🔒 | | |
| L | length | Internal 🗎 | | |
| L | at | Internal = | | | |
```



```
| **IRouter** | Interface | |||
| └ | inviteUser | External ! | ● |NO! |
| **CowStaking** | Implementation | Ownable |||
| L | <Constructor> | Public ! | • | NO! |
| L | stake | External ! | Wo! |
| L | claim | Public ! | • | NO! |
| L | forfeit | External ! | P | NO! |
| └ | toggleDepositLock | External ! | ● |NO! |
| L | unstake | External ! | 🔎 |NO! |
| L | unstakeFromCaller | External ! | • | onlyCaller |
| L | rebase | Public ! | WO! |
| L | contractBalance | Public ! | NO! |
| L | giveLockBonus | External ! | • |NO! |
| L | returnLockBonus | External ! | • |NO! |
| └ | setContract | External ! | ● | onlyManager |
| L | setRouter | Public ! | 🔴 | onlyManager |
| L | setWarmup | External ! | 🔎 | onlyManager |
| L | addCaller | Public ! | 📦 | onlyManager |
| L | delCaller | Public ! | 🔎 | onlyManager |
| L | isCaller | Public ! | NO! |
| L | getCaller | Public ! | NO! |
| L | getCallerLength | Public ! | NO! |
| **AbstractScow** | Implementation | |||
111111
| **IScowDepository** | Interface | |||
| L | rebaseForCal | External ! | • |NO! |
| L | getTotalAmount | External ! | NO! |
| **IScowCalculator** | Interface | IScowDepository |||
| L | balanceOf | External ! | NO! |
| L | gonsForBalance | External ! |
                                   |NO ! |
| L | balanceForGons | External ! | NO! | | |
| L | transferFrom | External ! | P | NO! |
| **FullMath** | Library | |||
| L | fullMul | Private 🔒 | | | |
| <sup>L</sup> | fullDiv | Private 🔐 |
| L | mulDiv | Internal 🗎 |
| **FixedPoint** | Library | |||
| <sup>L</sup> | decode | Internal 🔒 |
                            | L | decode112with18 | Internal 🗎 |
                                     | \cdot |
| L | fraction | Internal 🗎 | | |
111111
| **Scow** | Implementation | ERC20Permit, Ownable, AbstractScow |||
| └ | <Constructor> | Public ! | ● | ERC20 ERC20Permit |
| L | initialize | External ! | 🛑 |NO! |
| L | setSCowCalculator | External ! | OnlyManager |
```



```
| L | addRecipient | Public ! | 🔴 | onlyManager |
| L | setRecipientRation | Public ! | • | onlyManager |
| L | rebase | Public ! | • | onlyStakingContract calculatorInitialized |
| L | getAllRecipientAmount | Public ! | NO! |
| L | getRecipientAmount | Public ! | NO! |
| L | balanceOf | Public ! | calculatorInitialized |
| L | gonsForBalance | Public ! | calculatorInitialized |
| L | balanceForGons | Public ! | calculatorInitialized |
| L | circulatingSupply | Public ! | NO! |
| L | transfer | Public ! | O | calculatorInitialized |
| L | allowance | Public ! | NO! |
| └ | transferFrom | Public ! | ● | calculatorInitialized |
| L | approve | Public ! | • | NO! |
| L | increaseAllowance | Public ! | Governormal | NO! |
| L | decreaseAllowance | Public ! | • | NO! |
| └ | listContains | Internal 🗎 | | |
```

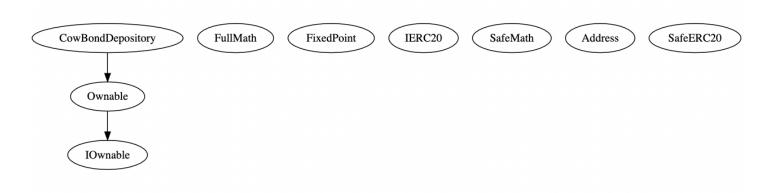
Interfi

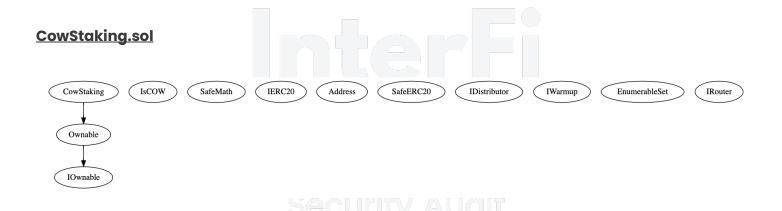
Smart Contract Security Audit



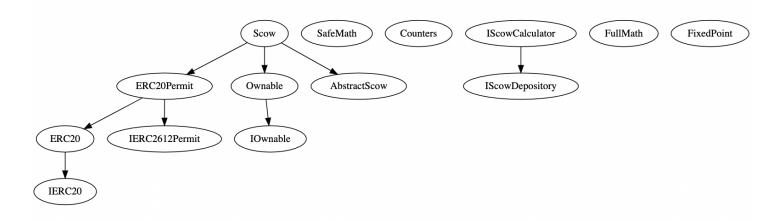
Smart Contract - Inheritance Graph

CowBondDepository.sol



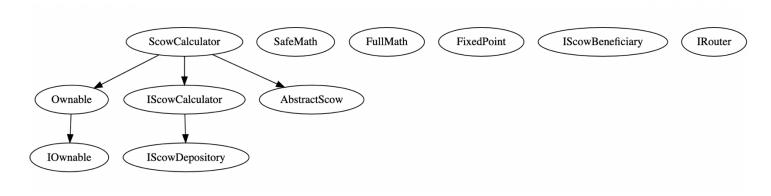


Scow.sol





ScowCalculator.sol





Smart Contract Security Audit



Smart Contract – Manual Analysis

Function	Description	Tested	Verdict
Total Supply	provides information about the total token supply	Yes	Passed
Balance Of	provides account balance of the owner's account	Yes	Passed
Transfer	executes transfers of a specified number of tokens to a specified address	Yes	Passed
Approve	allow a spender to withdraw a set number of tokens from a specified account	Yes	Passed
Allowance	returns a set number of tokens from a spender to the owner	Yes	Passed
Staking	returns a set number of tokens to staking wallet as rewards	Yes	Passed
Transfer Ownership	executes transfer of contract ownership to a specified wallet	Yes	Passed
Renounce Ownership	executes transfer of contract ownership to a dead address	Yes	Passed





- Be aware that active smart contract owner privileges constitute an elevated impact to smart contract's safety and security.
- CowStake's smart contracts does not utilize "ReentrancyGuard" to prevent reentrant calls to a function. Reentrancy Guard is a contract module that helps prevent reentrant calls to a function. Inheriting from Reentrancy Guard will make the nonReentrant modifier available, which can be applied to functions to make sure there are no nested (reentrant) calls to them.
- The smart contract has low severity issue which may or may not create any functional vulnerability.

```
"resource": "/CowStake.sol",

"owner": "_generated_diagnostic_collection_name_#0",

"severity": 8, (! Low Severity)

"Expected pragma, import directive or contract/interface/library definition",

"source": "solc",
```



Smart Contract - SWC Attacks

SWC ID	Description	Verdict
SWC-101	Integer Overflow and Underflow	Passed
SWC-102	Outdated Compiler Version	! Low
swc-103	Floating Pragma	Passed
SWC-104	Unchecked Call Return Value	Passed
SWC-105	Unprotected Ether Withdrawal	Passed
SWC-106	Unprotected SELFDESTRUCT Instruction	Passed
SWC-107	Re-entrancy Company of the Company o	Passed
SWC-108	State Variable Default Visibility	Passed
SWC-109	Uninitialized Storage Pointer	Passed
SWC-110	Assert Violation Smart Contract	Passed
swc-111	Use of Deprecated Solidity Functions	Passed
SWC-112	Delegate Call to Untrusted Callee	Passed
SWC-113	DoS with Failed Call	Passed
SWC-114	Transaction Order Dependence	Passed
SWC-115	Authorization through tx.origin	Passed
SWC-116	Block values as a proxy for time	Passed
SWC-117	Signature Malleability	Passed
SWC-118	Incorrect Constructor Name	Passed



SWC-119	Shadowing State Variables	Passed
SWC-120	Weak Sources of Randomness from Chain Attributes	Passed
SWC-121	Missing Protection against Signature Replay Attacks	Passed
SWC-122	Lack of Proper Signature Verification	Passed
SWC-123	Requirement Violation	Passed
SWC-124	Write to Arbitrary Storage Location	Passed
SWC-125	Incorrect Inheritance Order	Passed
SWC-126	Insufficient Gas Griefing	Passed
SWC-127	Arbitrary Jump with Function Type Variable	Passed
SWC-128	DoS With Block Gas Limit	Passed
SWC-129	Typographical Error	Passed
SWC-130	Right-To-Left-Override control character (U+202E)	Passed
SWC-131	Presence of unused variables	Passed
SWC-132	Unexpected Ether balance	Passed
SWC-133	Hash Collisions With Multiple Variable Length Arguments	Passed
SWC-134	Message call with hardcoded gas amount	Passed
SWC-135	Code With No Effects (Irrelevant/Dead Code)	! Low
SWC-136	Unencrypted Private Data On-Chain	Passed



Smart Contract - Risk Status & Radar Chart

Risk Severity Status ! Critical None critical severity issues identified None high severity issues identified ! High ! Medium None medium severity issues identified 2 low severity issues identified ! Low Verified 54 functions and instances verified and checked Score out of 100 Compiler Check 85 Interface Safety Static Analysis 80 Manual Analysis Software Analysis



Auditor's Verdict

InterFi team has performed a line-by-line manual analysis and automated review of the smart contracts. The smart contracts were analyzed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks.

- Cow Stake's smart contract source codes have LOW RISK SEVERITY
- Cow Stake's smart contracts have an ACTIVE OWNERSHIP



Note for stakeholders



- Be aware that active smart contract owner privileges constitute an elevated impact on smart contract's safety and security.
- ❖ Make sure that the project team's KYC/identity is verified by an independent firm.
- Always check if the contract's liquidity is locked. A longer liquidity lock plays an important role in project's longevity. It is recommended to have multiple liquidity providers.
- Examine the unlocked token supply in the owner, developer, or team's private wallets.
 Understand the project's tokenomics, and make sure the tokens outside of the LP Pair are vested or locked for a longer period of time.
- Ensure that the project's official website is hosted on a trusted platform, and is using an active SSL certificate. The website's domain should be registered for a longer period of time.



Important Disclaimer

InterFi Network provides contract development, testing, auditing and project evaluation services for blockchain projects. The purpose of the audit is to analyse the on-chain smart contract source code, and to provide basic overview of the project. This report should not be transmitted, disclosed, referred to, or relied upon by any person for any purposes without InterFi's prior written consent.

InterFi provides the easy-to-understand assessment of the project, and the smart contract (otherwise known as the source code). The audit makes no statements or warranties on the security of the code. It also cannot be considered as an enough assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. While we have used all the data at our disposal to provide the transparent analysis, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. Be aware that smart contracts deployed on a blockchain aren't resistant from external vulnerability, or a hack. Be aware that active smart contract owner privileges constitute an elevated impact to smart contract's safety and security. Therefore, InterFi does not guarantee the explicit security of the audited smart contract.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

This report should not be considered as an endorsement or disapproval of any project or team.

The information provided on 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. Do conduct your own due diligence and consult your financial advisor before making any investment decisions.



About InterFi Network

InterFi Network provides intelligent blockchain solutions. InterFi is developing an ecosystem that is seamless and responsive. Some of our services: Blockchain Security, Token Launchpad, NFT Marketplace, etc. InterFi's mission is to interconnect multiple services like Blockchain Security, DeFi, Gaming, and Marketplace under one ecosystem that is seamless, multi-chain compatible, scalable, secure, fast, responsive, and easy-to-use.

InterFi is built by a decentralized team of UI experts, contributors, engineers, and enthusiasts from all over the world. Our team currently consists of 6+ core team members, and 10+ casual contributors. InterFi provides manual, static, and automatic smart contract analysis, to ensure that project is checked against known attacks and potential vulnerabilities.

To learn more, visit https://interfi.network

To view our audit portfolio, visit https://github.com/interfinetwork

To book an audit, message https://t.me/interfiaudits





RELENTLESSLY SECURING THE PUBLIC BLOCKCHAIN | MADE IN CANADA