InterFi NETWORK

@INTERFINETWORK

SMART CONTRACT SECURITY AUDIT

InfinityPad

SMART CONTRACT AUDIT | TEAM KYC | PROJECT EVALUATION

RELENTLESSLY SECURING THE PUBLIC BLOCKCHAIN | MADE IN CANADA

Summary

Auditing Firm	InterFi Network
Architecture	InterFi "Echelon" Auditing Standard
Smart Contract Audit Approved By	Chris | Blockchain Specialist at InterFi Network
Project Overview Approved BY	Albert | Project Specialist at InterFi Network
Platform	Solidity
Audit Check (Mandatory)	Static, Software, Auto Intelligent & Manual Analysis
Project Check (Optional)	KYC, Website & Socials Analysis (Not Applicable)
Consultation Request Date	September 28, 2021
Report Date	September 30, 2021

Audit Summary

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

- **InfinityPad Token smart contract source code has LOW RISK SEVERITY.**
- **InfinityPad IDO smart contract source code has LOW RISK SEVERITY.**
- **Both of InfinityPad's solidity codes have successfully PASSED the smart contract audits.**

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

Table Of Contents

Project Overview

InterFi was consulted by InfinityPad on September 28, 2021 to conduct a smart contract security audit of **InfinityPad Token source code & InfinityPad IDO source code.**

InfinityPad is a next level launchpad to help raise capital across multiple blockchains on a single platform in a fully transparent and decentralized way by connecting projects and communities.

Project	Infinity Pad
Blockchain	InfinityPad is a multi-chain project **(Currently deployed on BSC)**
Language	Solidity
InfinityPad Token	0xfe82eff54a58c21ffc9523c4998d5dad84dcbd50
InfinityPad IDO	Not Deployed
Official Website	https://www.infinitypad.io/
Whitepaper	https://www.infinitypad.io/assets/images/InfinityPad_Deck.pdf
Twitter	https://twitter.com/InfinityPad_io
Telegram	https://t.me/infinitypad

Public logo

InfinityPad Token Source Code On Blockchain (BscScan Verified With Exact Match)

https://bscscan.com/address/0xfe82eff54a58c21ffc9523c4998d5dad84dcbd50#code

Compiler Version: v0.8.0+commit.c7dfd78e

Optimization Enabled: Yes with 200 runs

InfinityPad Token Source Code On InterFi GitHub

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

InfinityPad IDO Source Code On InterFi GitHub

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

GitHub Commit For Verification

710bceefd193f63bd69c954214842c943c17e38d

Audit Scope & Methodology

The scope of this report is to audit the smart contract codes of **InfinityPad Token & IDO**

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

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

InterFi has scanned and reviewed both the contracts for vulnerabilities, exploits, hacks, and back-doors. Below is the list of commonly known smart contract vulnerabilities, exploits, and hacks:

Category	
Smart Contract Vulnerabilities	❖ Re-entrancy (RE) ❖ Unhandled Exceptions (UE) ❖ Transaction Order Dependency (TO) ❖ Integer Overflow (IO) ❖ Unrestricted Action (UA) ❖ Ownership Takeover ❖ Gas Limit and Loops
Source Code Review	❖ Deployment Consistency ❖ Repository Consistency ❖ Data Consistency ❖ Token Supply Manipulation
Functional Assessment	❖ Access Control and Authorization ❖ Operations Trail and Event Generation ❖ Assets Manipulation ❖ Liquidity Access

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 Automated AI 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
- ❖ Consensys Surya
- ❖ 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.

Risk severity	Meaning
! Critical	This level vulnerabilities could be exploited easily, and can lead to asset loss, data loss, asset manipulation, 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 critical risk severity
! Medium	This level vulnerabilities are 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

Symbol	Meaning
🛑	Function can be modified
💵	Function is payable
🔒	Function is locked
🔐	Function can be accessed
!	Important functionality

Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Public !	🛑	NO!
└	owner	Public !		NO!
└	renounceOwnership	Public !	🛑	onlyOwner
└	transferOwnership	Public !	🛑	onlyOwner
└	_setOwner	Private 🔐	🛑	
Address	Library			
└	isContract	Internal 🔒		
└	sendValue	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	functionStaticCall	Internal 🔒		
└	functionStaticCall	Internal 🔒		
└	functionDelegateCall	Internal 🔒	🛑	
└	functionDelegateCall	Internal 🔒	🛑	
└	verifyCallResult	Internal 🔒		
SafeMath	Library			
└	tryAdd	Internal 🔒		
└	trySub	Internal 🔒		
└	tryMul	Internal 🔒		
└	tryDiv	Internal 🔒		
└	tryMod	Internal 🔒		
└	add	Internal 🔒		

```
| └ | sub | Internal 🔒 |   | |
| └ | mul | Internal 🔒 |   | |
| └ | div | Internal 🔒 |   | |
| └ | mod | Internal 🔒 |   | |
| └ | sub | Internal 🔒 |   | |
| └ | div | Internal 🔒 |   | |
| └ | mod | Internal 🔒 |   | |
||||||
| **IERC20** | Interface |  |||
| └ | totalSupply | External ❗ |  |NO❗ |
| └ | balanceOf | External ❗ |  |NO❗ |
| └ | transfer | External ❗ | 🛑 |NO❗ |
| └ | allowance | External ❗ |  |NO❗ |
| └ | approve | External ❗ | 🛑 |NO❗ |
| └ | transferFrom | External ❗ | 🛑 |NO❗ |
||||||
| **ERC20** | Implementation | Context, IERC20, Ownable |||
| └ | <Constructor> | Public ❗ | 🛑 |NO❗ |
| └ | name | Public ❗ |  |NO❗ |
| └ | symbol | Public ❗ |  |NO❗ |
| └ | decimals | Public ❗ |  |NO❗ |
| └ | totalSupply | Public ❗ |  |NO❗ |
| └ | balanceOf | Public ❗ |  |NO❗ |
| └ | transfer | Public ❗ | 🛑 |NO❗ |
| └ | allowance | Public ❗ |  |NO❗ |
| └ | approve | Public ❗ | 🛑 |NO❗ |
| └ | transferFrom | Public ❗ | 🛑 |NO❗ |
| └ | increaseAllowance | Public ❗ | 🛑 |NO❗ |
| └ | decreaseAllowance | Public ❗ | 🛑 |NO❗ |
| └ | _transfer | Internal 🔒 | 🛑 | |
| └ | _mint | Internal 🔒 | 🛑 | |
| └ | _burn | Internal 🔒 | 🛑 | |
| └ | _approve | Internal 🔒 | 🛑 | |
| └ | _setupDecimals | Internal 🔒 | 🛑 | |
| └ | updateTaxFee | Public ❗ | 🛑 | onlyOwner |
| └ | excludeFromFee | Public ❗ | 🛑 | onlyOwner |
| └ | _beforeTokenTransfer | Internal 🔒 | 🛑 | |
||||||
| **InfinityPad** | Implementation | ERC20 |||
| └ | <Constructor> | Public ❗ | 🛑 | ERC20 |
||||||
| **INFP** | Implementation |  |||
| └ | <Constructor> | Public ❗ | 🛑 |NO❗ |
| └ | submitToken | External ❗ | 🛑 | validateIDO |
| └ | emergencyWithdrawTokens | External ❗ | 💵 | validateTokenId onlyOwner |
| └ | updateFees | External ❗ | 🛑 | onlyOwner |
| └ | purchaseToken | External ❗ | 💵 | validateTokenId |
| └ | claimToken | Public ❗ | 🛑 | validateTokenId onlyOwner |
| └ | deleteToken | External ❗ | 🛑 | validateTokenId onlyOwner |
```

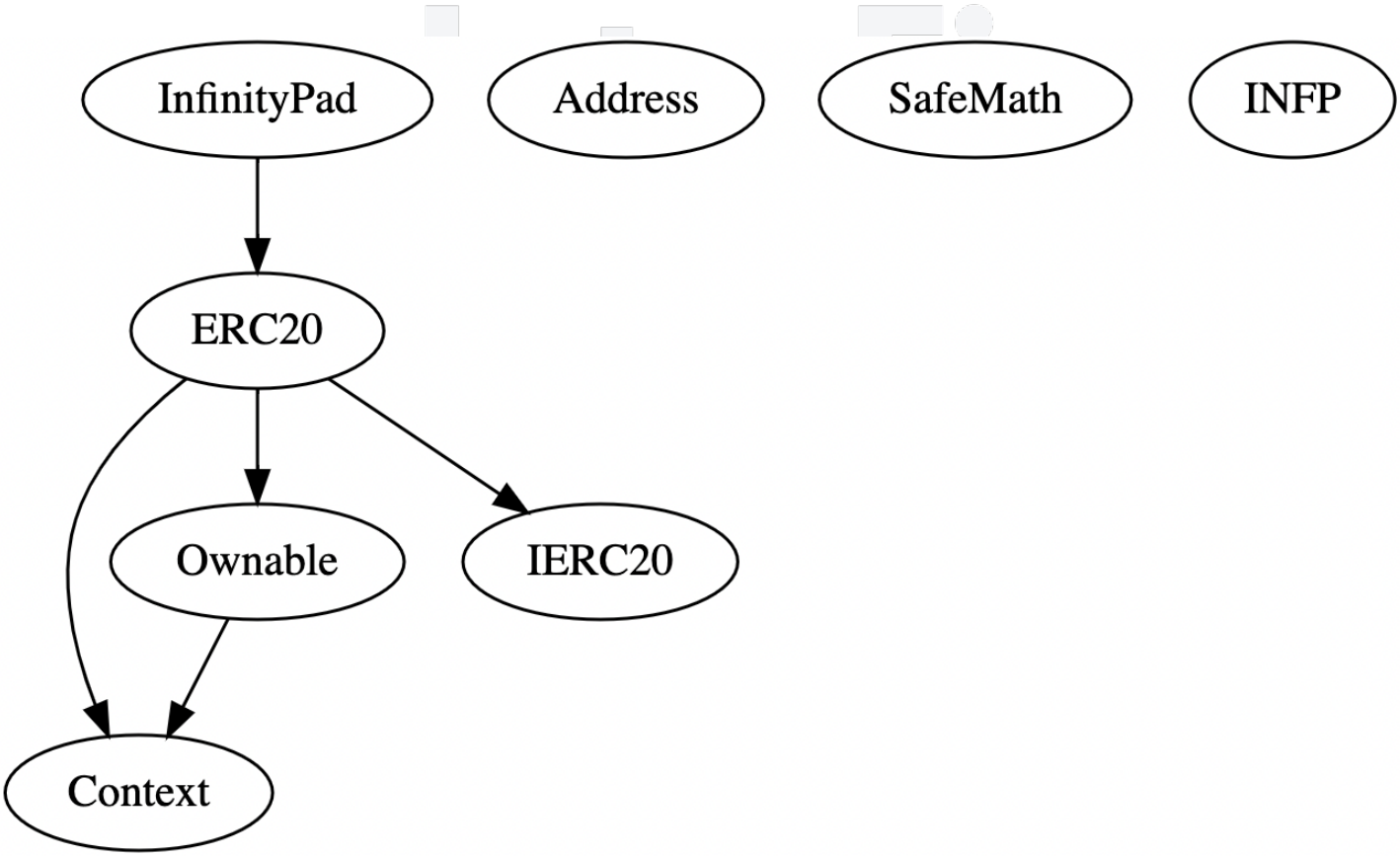
Smart Contract – Software Analysis

Callout functions – Sighash

```
Sighash   |   Function Signature
========================
16279055  =>  isContract(address)
39509351  =>  increaseAllowance(address,uint256)
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
8da5cb5b  =>  owner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
fc201122  =>  _setOwner(address)
24a084df  =>  sendValue(address,uint256)
a0b5ffb0  =>  functionCall(address,bytes)
241b5886  =>  functionCall(address,bytes,string)
2a011594  =>  functionCallWithValue(address,bytes,uint256)
d525ab8a  =>  functionCallWithValue(address,bytes,uint256,string)
c21d36f3  =>  functionStaticCall(address,bytes)
dbc40fb9  =>  functionStaticCall(address,bytes,string)
ee33b7e2  =>  functionDelegateCall(address,bytes)
57387df0  =>  functionDelegateCall(address,bytes,string)
946b5793  =>  verifyCallResult(bool,bytes,string)
884557bf  =>  tryAdd(uint256,uint256)
a29962b1  =>  trySub(uint256,uint256)
6281efa4  =>  tryMul(uint256,uint256)
736ecb18  =>  tryDiv(uint256,uint256)
38dc0867  =>  tryMod(uint256,uint256)
771602f7  =>  add(uint256,uint256)
b67d77c5  =>  sub(uint256,uint256)
c8a4ac9c  =>  mul(uint256,uint256)
a391c15b  =>  div(uint256,uint256)
f43f523a  =>  mod(uint256,uint256)
e31bdc0a  =>  sub(uint256,uint256,string)
b745d336  =>  div(uint256,uint256,string)
71af23e8  =>  mod(uint256,uint256,string)
18160ddd  =>  totalSupply()
70a08231  =>  balanceOf(address)
a9059cbb  =>  transfer(address,uint256)
dd62ed3e  =>  allowance(address,address)
095ea7b3  =>  approve(address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
06fdde03  =>  name()
95d89b41  =>  symbol()
313ce567  =>  decimals()
a457c2d7  =>  decreaseAllowance(address,uint256)
30e0789e  =>  _transfer(address,address,uint256)
4e6ec247  =>  _mint(address,uint256)
```

```
6161eb18  =>  _burn(address,uint256)
104e81ff  =>  _approve(address,address,uint256)
61e9edb2  =>  _setupDecimals(uint8)
b5505335  =>  updateTaxFee(uint256)
df8408fe  =>  excludeFromFee(address,bool)
cad3be83  =>  _beforeTokenTransfer(address,address,uint256)
f58a8430  =>  submitToken(IBEP20,uint256,uint256,uint256,uint256)
3e6df504  =>  emergencyWithdrawTokens(uint256)
78dacee1  =>  updateFees(uint256)
c2db2c42  =>  purchaseToken(uint256)
a9e7c2e5  =>  claimToken(uint256)
6297c16c  =>  deleteToken(uint256)
```

Callout functions – Inheritance Graph



Smart Contract – Manual Analysis

Function	Description	Tested	Verdict
TotalSupply	provides information about the total token supply	Yes	**Passed**
BalanceOf	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**
Ownership	executes renounce or transfer of the smart contract ownership	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**
Mint	executes minting of specified number of tokens and transfers them to the owner's account	NA	NA
Burn	executes transfers of a specified number of tokens to a burn address	NA	NA

Verified

- Owner cannot mint tokens after deployment.
- **InterFi was notified that the tokens will be locked on https://team.finance/ for trust and accountability.**

Important Information

InfinityPad Token & IDO smart contracts utilize "SafeMath" function to prevent known vulnerabilities.

```
library SafeMath {
function add(uint256 a, uint256 b) internal pure returns (uint256) {
    uint256 c = a + b;
    require(c >= a, 'SafeMath: addition overflow');

    return c;
  }
  function sub(uint256 a, uint256 b) internal pure returns (uint256) {
    return sub(a, b, 'SafeMath: subtraction overflow');
  }
  uint256 c = a * b;
    require(c / a == b, 'SafeMath: multiplication overflow');

    return c;
  }
function mod(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b > 0, "SafeMath: modulo by zero");
        return a % b;
    }
function mod(uint256 a, uint256 b) internal pure returns (uint256) {
    return mod(a, b, 'SafeMath: modulo by zero');
  }
```


InfinityPad Token & IDO smart contracts have 1 low severity issue which may or may not create any functional vulnerability.

- Expected pragma, import directive or contract/interface/library definition [solc19,1]

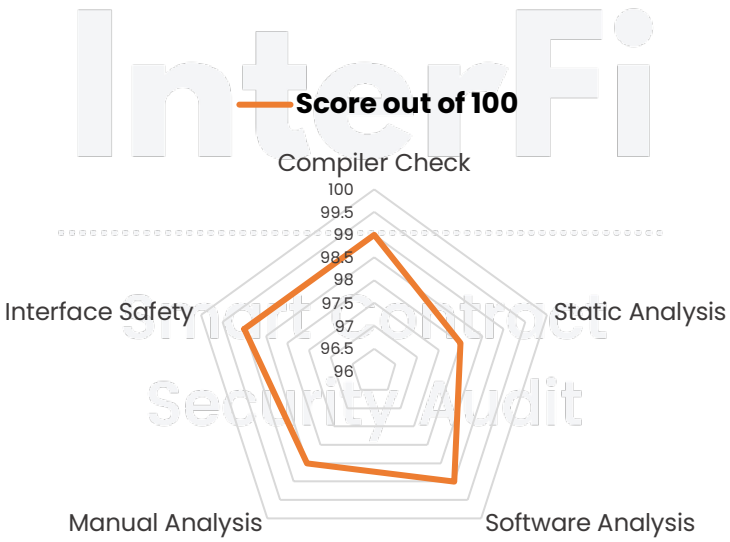
Smart Contract – SWC Attacks

SWC ID	Description	Verdict
SWC-101	Integer Overflow and Underflow	**Passed**
SWC-102	Outdated Compiler Version	**! Low**
SWC-103	Floating Pragma	**! Low**
SWC-104	Unchecked Call Return Value	**Passed**
SWC-105	Unprotected Ether Withdrawal	**Passed**
SWC-106	Unprotected SELFDESTRUCT Instruction	**Passed**
SWC-107	Re-entrancy	**Passed**
SWC-108	State Variable Default Visibility	**Passed**
SWC-109	Uninitialized Storage Pointer	**Passed**
SWC-110	Assert Violation	**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)	**Passed**
SWC-136	Unencrypted Private Data On-Chain	**Passed**

Smart Contract - Risk Status & Radar Chart

Risk Severity	Status
! Critical	None critical severity issues identified
! High	None high severity issues identified
! Medium	None medium severity issues identified
! Low	**2 Low severity issues identified**
Passed	**40 functions and instances verified and passed**



Compiler Check	99
Static Analysis	98
Software Analysis	99
Manual Analysis	98.5
Interface Safety	99

Auditor's Verdict

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

InfinityPad Token smart contract source code has LOW RISK SEVERITY.

InfinityPad IDO smart contract source code has LOW RISK SEVERITY.

Both of InfinityPad's solidity source codes have successfully PASSED the smart contract audits.

General Note:

- Be aware that active smart contract owner privileges constitute an elevated impact to smart contract's safety and security.
- Owner or developer KYC isn't checked and verified due to out of scope.
- Project's liquidity pair isn't checked and verified due to out of scope.
- Project website is not checked due to out of scope. The website hasn't been reviewed for SSL and lighthouse report.

Important Disclaimer

InterFi Network provides contract auditing and project verification 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

@INTERFINETWORK

RELENTLESSLY SECURING THE PUBLIC BLOCKCHAIN | MADE IN CANADA