InterFi NETWORK

@INTERFINETWORK

SMART CONTRACT SECURITY AUDIT OF

CRYPTO POOL

SMART CONTRACT AUDIT | SOLIDITY DEVELOPMENT & TESTING | PROJECT EVALUATION

RELENTLESSLY SECURING THE PUBLIC BLOCKCHAIN

Audit Introduction

Auditing Firm	InterFi Network
Audit Architecture	InterFi Echelon Auditing Standard
Language	Solidity
Client Firm	Crypto Pool
Website	https://www.crypto8pool.com/
Telegram	https://t.me/CryptopoolToken/
Twitter	https://twitter.com/Cryptopool_App/
Instagram	https://www.instagram.com/Cryptopool_App/
Report Date	March 12, 2022

About Crypto Pool

Cryptopool is a pool game and revolutionary PvP token where players can either try simple 1-vs-1 matches. play and challenge people to win $CP tokens, it's possible to customize the tables and to challenge your friends. Or course, there is a level up system in the game that keeps everything interesting, and that makes sure that people are being paired correctly. the team behind Crypto Pool has decades of experience in gaming, blockchain, and numerous other industries. We are united by our passion for blockchain technology and how it could revolutionize the gaming industry.

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:

- Crypto Pool's solidity source code has **LOW RISK SEVERITY**
- Crypto Pool's smart contract has an **ACTIVE OWNERSHIP**
- Important owner privileges – **BULK BLACKLIST, MAX TX & WALLET LIMIT, SET FEES, SET COOLDOWN, SET TRADING STATUS**
- Crypto Pool's smart contract owner has multiple "Write Contract" privileges. Centralization risk correlated to the active owner is **HIGH**

Be aware that smart contracts deployed on the blockchain aren't resistant to internal exploit, external vulnerability, or hack. For a detailed understanding of risk severity, source code vulnerability, functional hack, and audit disclaimer, kindly refer to the audit.

🚨 Token Contract address: **0x0EFd918A9A5dD198fD83E3a2D20Db3a9e0002092**

⛓️ Blockchain: **Binance Smart Chain**

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

Table Of Contents

Audit Scope

InterFi was consulted by Crypto Pool to conduct the smart contract security audit of their solidity source code. The audit scope of work is strictly limited to the mentioned solidity file(s) only:

- CryptoPool.sol

Solidity Source Code On Blockchain (Verified Contract Source Code)

https://bscscan.com/address/0x0efd918a9a5dd198fd83e3a2d20db3a9e0002092#code

Contract Name: Cryptopool

Compiler Version: v0.8.7

Optimization Enabled: Yes with 200 runs

Solidity Source Code On InterFi GitHub

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

SHA-1 Hash

Solidity source code is audited at hash #97023eb2a55ebb0d3c1230477c1651c5fa3e9cc0

Audit Methodology

The scope of this report is to audit the smart contract source code of Crypto Pool. InterFi has scanned the contract 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	
Smart Contract Vulnerabilities	❖ Re-entrancy ❖ Unhandled Exceptions ❖ Transaction Order Dependency ❖ Integer Overflow ❖ Unrestricted Action ❖ Incorrect Inheritance Order ❖ Typographical Errors ❖ Requirement Violation ❖ 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-by-line to identify potential vulnerabilities.
2. Static, Manual, and Software analysis:
   - ❖ Test coverage analysis 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 is analyzing a program to determine what inputs cause 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

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 that requires owning 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
! High	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.
! Medium	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
! Low	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.
! Informational	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

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 🔒 |  | |
||||||
| **IERC20** | Interface |  |||
| └ | totalSupply | External ! |  |NO! |
| └ | balanceOf | External ! |  |NO! |
| └ | transfer | External ! | 🛑 |NO! |
| └ | allowance | External ! |  |NO! |
| └ | approve | External ! | 🛑 |NO! |
| └ | transferFrom | External ! | 🛑 |NO! |
||||||
| **IERC20Metadata** | Interface | IERC20 |||
| └ | name | External ! |  |NO! |
| └ | symbol | External ! |  |NO! |
| └ | decimals | External ! |  |NO! |
||||||
| **ERC20** | Implementation | Context, IERC20, IERC20Metadata |||
| └ | <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 🔒 | 🛑 | |
```

└	_tokengeneration	Internal 🔒	🛑		
└	_burn	Internal 🔒	🛑		
└	_approve	Internal 🔒	🛑		
└	_beforeTokenTransfer	Internal 🔒	🛑		
Address	Library				
└	sendValue	Internal 🔒	🛑		
Ownable	Implementation	Context			
└	<Constructor>	Public ❗	🛑	NO❗	
└	owner	Public ❗		NO❗	
└	renounceOwnership	Public ❗	🛑	onlyOwner	
└	transferOwnership	Public ❗	🛑	onlyOwner	
└	_setOwner	Private 🔐	🛑		
IFactory	Interface				
└	createPair	External ❗	🛑	NO❗	
IRouter	Interface				
└	factory	External ❗		NO❗	
└	WETH	External ❗		NO❗	
└	addLiquidityETH	External ❗	💵	NO❗	
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗	
Cryptopool	Implementation	ERC20, Ownable			
└	<Constructor>	Public ❗	🛑	ERC20	
└	approve	Public ❗	🛑	NO❗	
└	transferFrom	Public ❗	🛑	NO❗	
└	increaseAllowance	Public ❗	🛑	NO❗	
└	decreaseAllowance	Public ❗	🛑	NO❗	
└	transfer	Public ❗	🛑	NO❗	
└	_transfer	Internal 🔒	🛑		
└	handle_fees	Private 🔐	🛑	mutexLock	
└	swapTokensForBNB	Private 🔐	🛑		
└	addLiquidity	Private 🔐	🛑		
└	updateLiquidityProvide	External ❗	🛑	onlyOwner	
└	updateLiquidityTreshhold	External ❗	🛑	onlyOwner	
└	updateTaxes	External ❗	🛑	onlyOwner	
└	updateSellTaxes	External ❗	🛑	onlyOwner	
└	updateRouterAndPair	External ❗	🛑	onlyOwner	
└	updateTradingEnabled	External ❗	🛑	onlyOwner	
└	updateMarketingWallet	External ❗	🛑	onlyOwner	
└	updateDevWallet	External ❗	🛑	onlyOwner	
└	updateOperationsWallet	External ❗	🛑	onlyOwner	
└	updateCooldown	External ❗	🛑	onlyOwner	
└	updateIsBlacklisted	External ❗	🛑	onlyOwner	
└	bulkIsBlacklisted	External ❗	🛑	onlyOwner	
└	updateAllowedTransfer	External ❗	🛑	onlyOwner	
└	bulkAllowedTransfer	External ❗	🛑	onlyOwner	
└	updateExemptFee	External ❗	🛑	onlyOwner	

└	bulkExemptFee	External ❗	🛑	onlyOwner
└	updateMaxTxLimit	External ❗	🛑	onlyOwner
└	updateMaxWalletlimit	External ❗	🛑	onlyOwner
└	rescueBNB	External ❗	🛑	onlyOwner
└	rescueBEP20	External ❗	🛑	onlyOwner
└	<Receive Ether>	External ❗	💵	NO❗

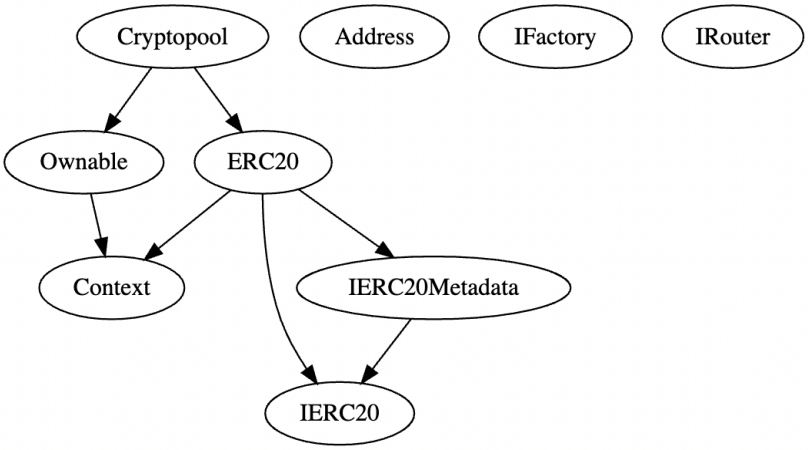
Software Analysis

Function Signatures

```
39509351  =>  increaseAllowance(address,uint256)
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
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)
c143c0de  =>  _tokengeneration(address,uint256)
6161eb18  =>  _burn(address,uint256)
104e81ff  =>  _approve(address,address,uint256)
cad3be83  =>  _beforeTokenTransfer(address,address,uint256)
24a084df  =>  sendValue(address,uint256)
8da5cb5b  =>  owner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
fc201122  =>  _setOwner(address)
c9c65396  =>  createPair(address,address)
c45a0155  =>  factory()
ad5c4648  =>  WETH()
f305d719  =>  addLiquidityETH(address,uint256,uint256,uint256,address,uint256)
791ac947  =>
swapExactTokensForETHSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
cd7a9c18  =>  handle_fees(uint256,Taxes)
56c3726b  =>  swapTokensForBNB(uint256)
9cd441da  =>  addLiquidity(uint256,uint256)
1340538f  =>  updateLiquidityProvide(bool)
42b6fa11  =>  updateLiquidityTreshhold(uint256)
c4c2ff4f  =>  updateTaxes(Taxes)
ddd8489e  =>  updateSellTaxes(Taxes)
40b28c2f  =>  updateRouterAndPair(address,address)
3d30d20b  =>  updateTradingEnabled(bool,uint256,uint256)
aacebbe3  =>  updateMarketingWallet(address)
1816467f  =>  updateDevWallet(address)
30d5d18d  =>  updateOperationsWallet(address)
e517f2b9  =>  updateCooldown(bool,uint256)
5b24ea5e  =>  updateIsBlacklisted(address,bool)
13f97a8e  =>  bulkIsBlacklisted(address[],bool)
b5c57145  =>  updateAllowedTransfer(address,bool)
```


```
81428be1  =>  bulkAllowedTransfer(address[],bool)
355496ca  =>  updateExemptFee(address,bool)
0e375a5c  =>  bulkExemptFee(address[],bool)
59759f61  =>  updateMaxTxLimit(uint256,uint256)
d8672e51  =>  updateMaxWalletlimit(uint256)
441b1d30  =>  rescueBNB(uint256)
3490560d  =>  rescueBEP20(address,uint256)
```

Inheritance Graph



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**
Buy Back	is an action in which the project buys back its tokens from the existing holders usually at a market price	NA	NA
Burn	executes transfers of a specified number of tokens to a burn address	Yes	**Passed**
Mint	executes the creation of a specified number of tokens and adds it to the total supply	NA	NA
Rebase	circulating token supply adjusts (increases or decreases) automatically according to a token's price fluctuations	NA	NA
Blacklist	stops specified wallets from interacting with the smart contract function modules	Yes	**! Low**
Lock	stops or locks all function modules of the smart contract	NA	NA

Function	Description	Tested	Verdict
Dividend	executes transfers of a specified dividend token to a specified address	NA	NA
Airdrop	executes transfers of a specified number of tokens to a specified address	NA	NA
Max Transaction	a non-whitelisted wallet can only transfer a specified number of tokens	Yes	**! Low**
Max Wallet	a non-whitelisted wallet can only hold a specified number of tokens	Yes	**! Low**
Cooldown Timer	functionality to limit the number of transactions that a wallet can make within 24-hours	Yes	**! Low**
Anti Bot	stops some or all bot wallets from interacting with the smart contract	NA	NA
Anti Snipe	prevents bots from making transaction at "addLiquidity" block	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**

Best Practices ✅

- Owner cannot mint tokens after initial contract creation/deployment.

Note ⚠️

- Active smart contract owner: 0x12ef15a447163f8f11ef68b18ed3b2a27f427d3f
- ***Be aware that active smart contract owner privileges constitute an elevated impact to smart contract safety and security.***
- Smart contract owner can ***change trading status***, this function module can be used to stop the users from buying or selling the assets.
- Smart contract can ***burn*** tokens to decrease the total supply.
- Smart contract utilizes ***anti-snipe*** function module to prevent bots from making transactions at "addLiquidity" block.
- Smart contract owner can ***bulk blacklist*** certain wallets from interacting with the contract function modules.
- Smart contract owner can ***change transaction fees***. This function module can be used to impose extraordinary transaction fees. No arbitrary limit set.
- Smart contract owner can ***change max buy, sell, and wallet limit***. The smart contract owner can change the value to "zero". No arbitrary limit set.
- Smart contract has a ***low severity issue*** which may or may not create any functional vulnerability.

{

"resource": " /CryptoPool.sol",

"severity": 8, (! Low Severity)

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

}

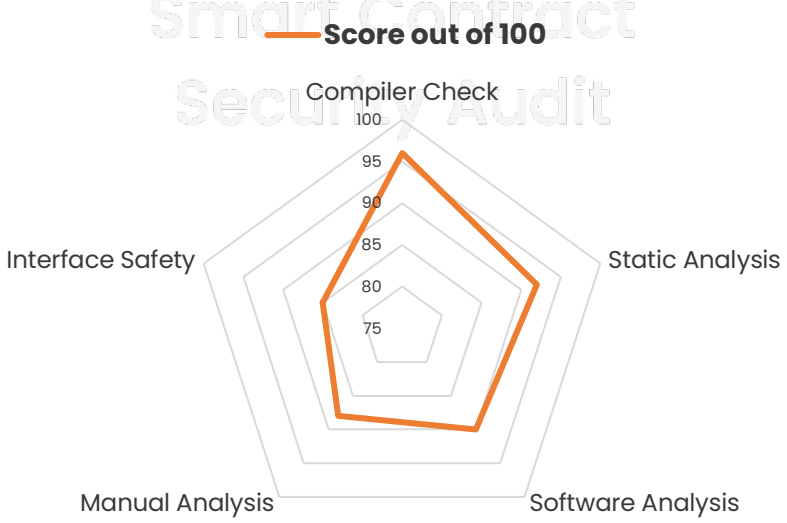
SWC Attacks

SWC ID	Description	Verdict
SWC-101	Integer Overflow and Underflow	**Passed**
SWC-102	Outdated Compiler Version	**! Informational**
SWC-103	Floating Pragma	**! Low**
SWC-104	Unchecked Call Return Value	**Passed**
SWC-105	Unprotected Ether Withdrawal	**Passed**
SWC-106	Unprotected SELF-DESTRUCT 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 the hardcoded gas amount	**Passed**
SWC-135	Code With No Effects (Irrelevant/Dead Code)	**Passed**
SWC-136	Unencrypted Private Data On-Chain	**Passed**

Risk Status & Radar Chart

Risk Severity	Status
! High	No high severity issues identified
! Medium	No medium severity issues identified
! Low	5 low severity issues identified ❖ Please Review Report
! Informational	2 informational severity issues identified ❖ Active Ownership ❖ Outdated Compiler
Verified	54 functions and instances verified and checked



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.

- Crypto Pool's smart contract source code has **LOW RISK SEVERITY**
- Crypto Pool's smart contract has an **ACTIVE OWNERSHIP**
- Crypto Pool's smart contract owner has multiple "Write Contract" privileges. Centralization risk correlated to the active owner is **HIGH**

Note for stakeholders

- Be aware that active smart contract owner privileges constitute an elevated impact on smart contract safety and security.
- If the smart contract is not deployed on any blockchain at the time of the audit, the contract can be modified or altered before blockchain development. Verify contract's deployment status in the audit report.
- 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 the 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.
- 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.

Important Disclaimer

InterFi Network provides contract development, testing, auditing and project evaluation services for blockchain projects. The purpose of the audit is to analyze the on-chain smart contract source code and to provide a basic overview of the project. **This report should not be transmitted, disclosed, referred to, or relied upon by any person for any purpose 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 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 to external vulnerability, or a hack. Be aware that active smart contract owner privileges constitute an elevated impact on smart contract 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 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. Do conduct your 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