SMART CONTRACT SECURITY AUDIT OF

MDTPROTOCOL

Audit Introduction

Auditing Firm	InterFi Network
Audit Architecture	InterFi Echelon Auditing Standard
Language	Solidity
Client Firm	MdtProtocol
Website	https://mdt.gg/
Twitter	https://twitter.com/mdtProtocol/
Github	https://github.com/mdtProtocol/
Discord	https://discord.gg/mdtProtocol/
Report Date	March 03, 2022

About MdtProtocol

Premier sniper bot that aims for inclusivity rather than exclusivity; just pay for what you use, we don't collect any upfront payment to use our trading bot.

Audit Summary

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

❖ MdtProtocol's smart contract source codes have **LOW RISK SEVERITY**

❖ MdtProtocol's smart contracts use **GNOSIS MULTISIG** to reduce centralization risk

❖ Privileges to watch – **MINT, BLACKLIST, ADD & REMOVE RELAYER**

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.

🔰 mdtERC20 Contract address: **0x1c6ab21DEf24F6361873191e3C52Fc76C6e0CA15**

🔰 mdtLogic Contract address: **0xF4b0Aad56c9d1752983881c1F40F9a5AaE790c11**

🔰 mdtProxy Contract address: **0xc72cB9584bf1C7d1A9422466823c562077126f65**

🔀 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 MdtProtocol to conduct the smart contract security audit of their solidity source codes. The audit scope of work is strictly limited to the mentioned solidity file(s) only:

- ❖ ImdtLogic.sol
- ❖ mdtERC20.sol
- ❖ mdtLogic.sol
- ❖ mdtProxy.sol

Solidity Source Code On GitHub

// Private code repository

SHA-1 Hash

Solidity source code is audited at hash #21447048d8af1e2ee387724c7a9a1f82f11d0940

Audit Methodology

The scope of this report is to audit the smart contract source codes of MdtProtocol. 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

```
| **ImdtLogic** | Interface |  |||
| └ | gnosis | External ❗ |   |NO❗ |
||||||
| **mdtERC20** | Implementation | Ownable, ERC20 |||
| └ | <Constructor> | Public ❗ | 🛑  | ERC20 |
| └ | mint | External ❗ | 🛑  | onlyOwner |
| └ | burn | External ❗ | 🛑  | onlyOwner |
| └ | _approve | Internal 🔒 | 🛑  | |
| └ | _beforeTokenTransfer | Internal 🔒 | 🛑  | |
||||||
| **mdtLogic** | Implementation | Initializable, ReentrancyGuard, Context |||
| └ | <Constructor> | Public ❗ | 🛑  | initializer |
| └ | initialize | External ❗ | 🛑  | initializer |
| └ | <Receive Ether> | External ❗ | 💲 |NO❗ |
| └ | recoverEth | External ❗ | 🛑  | onlyGnosis |
| └ | recoverToken | External ❗ | 🛑  | onlyGnosis |
| └ | rate | External ❗ |   |NO❗ |
| └ | setRate | External ❗ | 🛑  | onlyGnosis |
| └ | gnosis | External ❗ |   |NO❗ |
| └ | dollar | External ❗ |   |NO❗ |
| └ | royalty | External ❗ |   |NO❗ |
| └ | credit | External ❗ |   |NO❗ |
| └ | router | External ❗ |   |NO❗ |
| └ | share | External ❗ |   |NO❗ |
| └ | setShare | External ❗ | 🛑  | onlyGnosis |
| └ | _setShareDefault | Internal 🔒 | 🛑  | |
| └ | governance | External ❗ |   |NO❗ |
| └ | setGovernance | External ❗ | 🛑  | onlyGnosis |
| └ | _setGovernanceDefault | Internal 🔒 | 🛑  | |
| └ | discounts | External ❗ |   |NO❗ |
| └ | setDiscounts | External ❗ | 🛑  | onlyGnosis |
```

┗	_setDiscountsDefault	Internal 🔒	🔴	
┗	loyalty	External ❗		NO❗
┗	setLoyalty	External ❗	🔴	onlyGnosis
┗	_setLoyaltyDefault	Internal 🔒	🔴	
┗	relayer	External ❗		NO❗
┗	addRelayer	External ❗	🔴	onlyGnosis
┗	removeRelayer	External ❗	🔴	onlyGnosis
┗	blacklist	External ❗		NO❗
┗	addBlacklist	External ❗	🔴	onlyRelayer
┗	removeBlacklist	External ❗	🔴	onlyRelayer
┗	special	External ❗		NO❗
┗	_addSpecial	Internal 🔒	🔴	
┗	_removeSpecial	Internal 🔒	🔴	
┗	upgradeSpecial	External ❗	🔴	onlyRelayer
┗	diamond	External ❗		NO❗
┗	_addDiamond	Internal 🔒	🔴	
┗	_removeDiamond	Internal 🔒	🔴	
┗	upgradeDiamond	External ❗	🔴	onlyRelayer
┗	gold	External ❗		NO❗
┗	_addGold	Internal 🔒	🔴	
┗	_removeGold	Internal 🔒	🔴	
┗	upgradeGold	External ❗	🔴	onlyRelayer
┗	silver	External ❗		NO❗
┗	_addSilver	Internal 🔒	🔴	
┗	_removeSilver	Internal 🔒	🔴	
┗	upgradeSilver	External ❗	🔴	onlyRelayer
┗	bronze	External ❗		NO❗
┗	_addBronze	Internal 🔒	🔴	
┗	_removeBronze	Internal 🔒	🔴	
┗	upgradeBronze	External ❗	🔴	onlyRelayer
┗	revokeRoles	External ❗	🔴	onlyRelayer
┗	snipe	External ❗		NO❗
┗	sniper	External ❗		NO❗
┗	sniperLoyaltyPoint	External ❗		NO❗
┗	_addSniperLoyaltyPoint	Internal 🔒	🔴	
┗	GnosisAddSniperLoyaltyPoint	External ❗	🔴	onlyGnosis
┗	_removeSniperLoyaltyPoint	Internal 🔒	🔴	
┗	GnosisRemoveSniperLoyaltyPoint	External ❗	🔴	onlyGnosis
┗	sniperMarksmanStatus	External ❗		NO❗
┗	_updateSniperMarksmanStatus	Internal 🔒	🔴	
┗	GnosisUpdateSniperMarksmanStatus	External ❗	🔴	onlyGnosis
┗	sniperWalletsCount	External ❗		NO❗
┗	_increaseSniperWalletsCount	Internal 🔒	🔴	
┗	GnosisIncreaseSniperWalletsCount	External ❗	🔴	onlyGnosis
┗	_decreaseSniperWalletsCount	Internal 🔒	🔴	
┗	GnosisDecreaseSniperWalletsCount	External ❗	🔴	onlyGnosis
┗	_resetSniperWalletsCount	Internal 🔒	🔴	
┗	GnosisResetSniperWalletsCount	External ❗	🔴	onlyGnosis
┗	blockchainMultiplier	External ❗		NO❗
┗	setBlockchainMultiplier	External ❗	🔴	onlyGnosis

└	sniperRate	Public ❗		NO❗
└	mintCredit	External ❗	🔴	nonReentrant registerSniper
└	redeemLoyaltyPoint	External ❗	🔴	nonReentrant registerSniper
└	frontendMarksman	Public ❗		NO❗
└	activateMarksman	External ❗	🔴	nonReentrant registerSniper
└	frontendWalletsCount	External ❗		NO❗
└	upgradeWalletsCount	External ❗	🔴	nonReentrant registerSniper
└	executeSnipe	External ❗	🔴	nonReentrant registerSnipe registerSniper
└	_sniperRoleId	Internal 🔒		
└	version	External ❗		NO❗
mdtProxy	Implementation	TransparentUpgradeableProxy, Context		
└	<Constructor>	Public ❗	🔴	TransparentUpgradeableProxy

Software Analysis

Function Signatures

```
50284610  =>  GnosisAddSniperLoyaltyPoint(address[],uint256)
238c8aad  =>  gnosis()
40c10f19  =>  mint(address,uint256)
9dc29fac  =>  burn(address,uint256)
104e81ff  =>  _approve(address,address,uint256)
cad3be83  =>  _beforeTokenTransfer(address,address,uint256)
3641e083  =>  initialize(uint256,address,address,address,address)
bcdb446b  =>  recoverEth()
0d0d6941  =>  recoverToken(IERC20)
2c4e722e  =>  rate()
34fcf437  =>  setRate(uint256)
51adeb57  =>  dollar()
29ee566c  =>  royalty()
a06d083c  =>  credit()
f887ea40  =>  router()
a8d5fd65  =>  share()
6bf8e9cc  =>  setShare(uint256,uint256)
7eafb8ac  =>  _setShareDefault()
5aa6e675  =>  governance()
d6a1056b  =>  setGovernance(uint256,uint256,uint256,uint256,uint256,uint256,uint256)
c628749d  =>  _setGovernanceDefault()
75af65d1  =>  discounts()
a9608210  =>
setDiscounts(uint256,uint256,uint256,uint256,uint256,uint256,uint256,uint256,uint256,uint256)
175a5b1a  =>  _setDiscountsDefault()
8d16a694  =>  loyalty()
3ba41f5a  =>  setLoyalty(uint256,uint256,uint256)
cb6a908e  =>  _setLoyaltyDefault()
c9b5ef8e  =>  relayer(address)
dd39f00d  =>  addRelayer(address)
60f0a5ac  =>  removeRelayer(address)
f9f92be4  =>  blacklist(address)
3d2cc56c  =>  addBlacklist(address[])
7911ef9d  =>  removeBlacklist(address[])
ba791b9a  =>  special(address)
9e66a6f5  =>  _addSpecial(address)
bbf44fa9  =>  _removeSpecial(address)
031d47b2  =>  upgradeSpecial(address[])
d8b40ab7  =>  diamond(address)
923bd18b  =>  _addDiamond(address)
6d62ccdd  =>  _removeDiamond(address)
71a5cb0d  =>  upgradeDiamond(address[])
88bfb4a4  =>  gold(address)
6673ef45  =>  _addGold(address)
369d33ea  =>  _removeGold(address)
```

```
a74d803a  =>  upgradeGold(address[])
36fd4464  =>  silver(address)
9559ae3d  =>  _addSilver(address)
2cd24f24  =>  _removeSilver(address)
40d40652  =>  upgradeSilver(address[])
62213f8b  =>  bronze(address)
9d39f4f4  =>  _addBronze(address)
8103982d  =>  _removeBronze(address)
c379471e  =>  upgradeBronze(address[])
e5ef8405  =>  revokeRoles(address[])
39b0ff21  =>  snipe()
0300b50a  =>  sniper()
166da037  =>  sniperLoyaltyPoint(address)
23cac8a2  =>  _addSniperLoyaltyPoint(address,uint256)
3bf6562a  =>  _removeSniperLoyaltyPoint(address,uint256)
6314fe6a  =>  GnosisRemoveSniperLoyaltyPoint(address[],uint256)
79f79274  =>  sniperMarksmanStatus(address)
23b9de35  =>  _updateSniperMarksmanStatus(address,uint256)
522060da  =>  GnosisUpdateSniperMarksmanStatus(address[],uint256)
002d9463  =>  sniperWalletsCount(address)
2a6e9d3a  =>  _increaseSniperWalletsCount(address)
18eb6cba  =>  GnosisIncreaseSniperWalletsCount(address[])
c1af65c6  =>  _decreaseSniperWalletsCount(address)
c3d8ecc5  =>  GnosisDecreaseSniperWalletsCount(address[])
456c6972  =>  _resetSniperWalletsCount(address)
541fe8a0  =>  GnosisResetSniperWalletsCount(address[])
dd507008  =>  blockchainMultiplier(uint256)
822c6ce8  =>  setBlockchainMultiplier(uint256,uint256)
a4149770  =>  sniperRate(uint256,address,uint256)
16717b31  =>  mintCredit(IERC20,uint256,address)
1d7eb5b4  =>  redeemLoyaltyPoint(uint256,address)
7efa5062  =>  frontendMarksman(address)
4faa87b8  =>  activateMarksman(uint256)
ceebd8bc  =>  frontendWalletsCount(address)
ef35a40e  =>  upgradeWalletsCount()
9d6ec91a  =>  executeSnipe(uint256,uint256,uint256)
93c1297a  =>  _sniperRoleId(address)
54fd4d50  =>  version()
```

Inheritance Graph
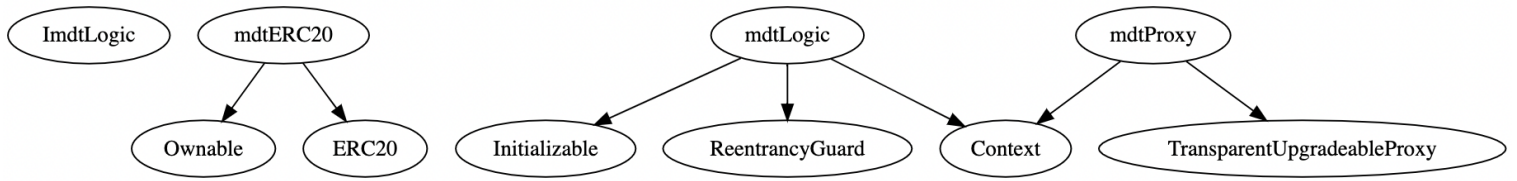
Manual Analysis

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
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	Yes	! Informational
Blacklist	stops specified wallets from interacting with the smart contract function modules	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

Notes

❖ Smart contract uses **gnosis multi-sig wallet authorizations** to reduce the centralization risk.

❖ Only Gnosis can **add or remove contract relayer.**

❖ Only Relayer can **blacklist** wallets from interacting with the contract function modules.

❖ mdtProxy.sol smart contract has an upgradability mechanism. The smart contract utilizes **Transparent Upgradeable Proxy.**

❖ mdtProxy.sol is the owner of mdtERC20.sol, and only the owner can **mint**. mdtLogic.sol is the implementation for the proxy. In the dapp, users can spend any BEP20 tokens (token must be paired with the dollar in contract), and call the mintCredit function in the proxy.

❖ mdtLogic.sol smart contract **utilizes "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.

SWC Attacks

| SWC ID	Description	Verdict
SWC-101	Integer Overflow and Underflow	Passed
SWC-102	Outdated Compiler Version	Passed
SWC-103	Floating Pragma	Passed
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	No low severity issues identified
	❖ Please Review Report
! Informational	1 informational severity issues identified
Verified	54 functions and instances verified and checked

Score out of 100
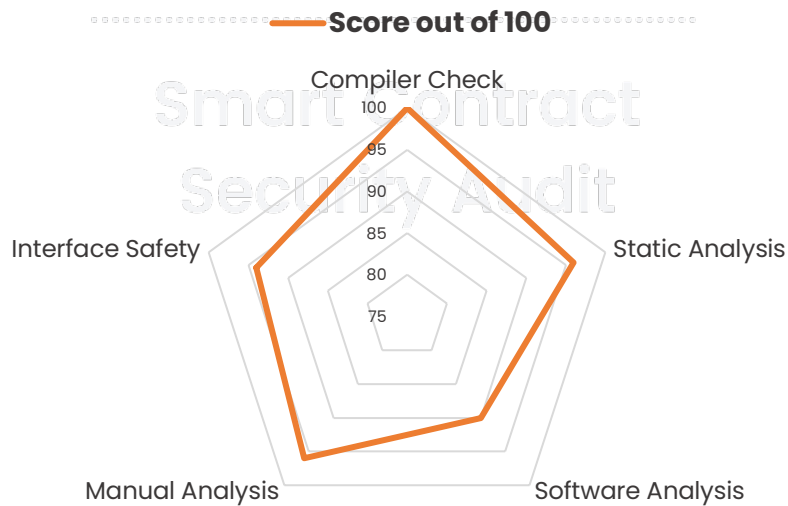
Auditor's Verdict

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

❖ MdtProtocol's smart contract source codes have **LOW RISK SEVERITY**

❖ MdtProtocol's smart contracts use **GNOSIS MULTISIG** to reduce centralization risk

Note for stakeholders

❖ Be aware that active smart contract owner privileges constitute an elevated impact on smart contract safety and security.

❖ Make sure that the project team's KYC/identity is verified by an independent firm.

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 🇨🇦