

# SMART CONTRACT SECURITY AUDIT OF **CX1**



# Summary

Auditing Firm	<b>InterFi Network</b>
Client Firm	<b>CX1</b>
Language	<b>Solidity</b>
Mandatory Audit Check	<b>Static, Software, Auto Intelligent &amp; Manual Analysis</b>
Final Report Date	<b>April, 2022</b>

## Audit Summary

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

- ❖ CX1's smart contract source code has **LOW RISK SEVERITY**
- ❖ CX1 has **PASSED** the smart contract audit
- ❖ CX1 no uses any mint function.

For the detailed understanding of risk severity, source code vulnerability, and functional test, kindly refer to the audit. Please note, only a number of CX1 contracts from their repository are audited, the contracts make external calls and import various code packages to work effectively, InterFi does not provide explicit guarantee on the safety and security of these calls/packages.

 MacroSwap.app DEX is a fork of MMFinance DEX but on Cronos blockchain

 **Blockchain:** Cronos Chain

 **Verify the authenticity of this report on InterFi's GitHub:** <https://github.com/interfiaudit>



# Table Of Contents

## Project Information

<b>Overview .....</b>	<b>4</b>
-----------------------	----------

## InterFi “Echelon” Audit Standard

<b>Audit Scope &amp; Methodology .....</b>	<b>6</b>
--	----------

<b>InterFi’s Risk Classification .....</b>	<b>8</b>
--	----------

## Smart Contract Risk Assessment

<b>Static Analysis .....</b>	<b>9</b>
------------------------------	----------

<b>Software Analysis .....</b>	<b>13</b>
--------------------------------	-----------

<b>Manual Analysis.....</b>	<b>16</b>
-----------------------------	-----------

<b>SWC Attacks .....</b>	<b>19</b>
--------------------------	-----------

<b>Risk Status &amp; Radar Chart .....</b>	<b>21</b>
--	-----------

## Report Summary

<b>Auditor’s Verdict .....</b>	<b>22</b>
--------------------------------	-----------

## Legal Advisory

<b>Important Disclaimer .....</b>	<b>23</b>
-----------------------------------	-----------

<b>About InterFi Network.....</b>	<b>24</b>
-----------------------------------	-----------



# Project Overview

InterFi was consulted by CX1 to conduct the smart contract security audit of their solidity source codes.

## About CX1 - MacroSwap

**CX1 DEX – The DEX is a fork of PancakeSwap but on SmartBCH. There will be one tokens by the DEX:**

**a) an Exchange token: CX1**

**The CX1 team comprises experienced MacroSwap developers, and seasoned blockchain product and project leads.**

Project	CX1
Blockchain	Cronos Chain
Language	Solidity
Contracts	<a href="https://cronoscan.com/address/0x06a58fc0defd68f08f64dc8a126326d4e92bb3c6">cronoscan.com/address/0x06a58fc0defd68f08f64dc8a126326d4e92bb3c6</a>
Website	<a href="https://macroswap.app/">https://macroswap.app/</a>
Telegram	<a href="https://t.me/macroswap">https://t.me/macroswap</a>
Twitter	<a href="https://twitter.com/macroswapapp">https://twitter.com/macroswapapp</a>
Instagram	<a href="https://instagram.com/macrsoswap">https://instagram.com/macrsoswap</a>
Medium	<a href="https://macroswap.medium.com/">https://macroswap.medium.com/</a>



## Project Logo



## Solidity Source Codes On Cronoscan

<https://cronoscan.com/address/0x06a58fc0defd68f08f64dc8a126326d4e92bb3c6>

## Solidity Source Codes Under Scope

- ❖ **MasterChef.sol**
- ❖ **CRR0Token.sol**
- ❖ **CROVault.sol**
- ❖ **MMFFactory.sol**
- ❖ **MMFPair.sol**
- ❖ **MMFRouter.sol**
- ❖ **MMFLibrary.sol**
- ❖ **SmartChefFactory.sol**
- ❖ **SmartChefInitializable.sol**

## SHA-1 Hash

**Solidity source codes are audited at hash #f7c2bef01d6fcfd87103b0f0cdf4241bb8d0789f**



# Audit Scope & Methodology

The scope of this report is to audit the smart contract source codes of CX1. InterFi has scanned the contract codes and reviewed the project for common vulnerabilities, exploits, hacks, and backdoors. Below is the list of commonly known smart contract vulnerabilities, exploits, and hacks:

## Category

---

Smart Contract Vulnerabilities	❖ <b>Re-entrancy</b>
	❖ <b>Unhandled Exceptions</b>
	❖ <b>Transaction Order Dependency</b>
	❖ <b>Integer Overflow</b>
	❖ <b>Unrestricted Action</b>
	❖ <b>Incorrect Inheritance Order</b>
	❖ <b>Typographical Errors</b>
Source Code Review	❖ <b>Requirement Violation</b>
	❖ <b>Ownership Takeover</b>
	❖ <b>Gas Limit and Loops</b>
	❖ <b>Deployment Consistency</b>
Functional Assessment	❖ <b>Repository Consistency</b>
	❖ <b>Data Consistency</b>
	❖ <b>Token Supply Manipulation</b>
	❖ <b>Access Control and Authorization</b>
	❖ <b>Operations Trail and Event Generation</b>
	❖ <b>Assets Manipulation</b>
	❖ <b>Liquidity Access</b>



## **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 smartcontract:**

**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 CodeAnalyzer**
- ❖ **Solidity Code Compiler**



# InterFi's Risk Classification

**Smart contracts are generally designed to manipulate and hold funds. 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
<b>! Critical</b>	<b>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.</b>
<b>! High</b>	<b>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</b>
<b>! Medium</b>	<b>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.</b>
<b>! Low</b>	<b>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</b>

































































































# Smart Contract – Static Analysis

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

```

| **IMigratorChef** | Interface |    ||| |
|  | migrate | External  |   |  |NO!  |
|||||
| **MasterChef** | Implementation | Ownable |||
|  | <Constructor> | Public  |   |  |NO!  |
|  | updateMultiplier | Public  |   |   | onlyOwner |
|  | updateRewardPerBlock | External  |   |  | onlyOwner |
|  | poolLength | External  |   | |NO!  |
|  | add | Public  |   |  | onlyOwner |
|  | set | Public  |   |  | onlyOwner |
|  | updateStakingPool | Internal  |  |   | |
|  | setMigrator | Public  |   |  | onlyOwner |
|  | migrate | Public  |   |  |NO!  |
|  | migrateControl | Public  |   |  | onlyOwner |
|  | getMultiplier | Public  |   | |NO!  |
|  | pendingCake | External  |   | |NO!  |
|  | massUpdatePools | Public  |   |  |NO!  |
|  | updatePool | Public  |   |  |NO!  |
|  | deposit | Public  |   |  |NO!  |
|  | withdraw | Public  |   |  |NO!  |
|  | enterStaking | Public  |   |  |NO!  |
|  | leaveStaking | Public  |   |  |NO!  |
|  | emergencyWithdraw | Public  |   |  |NO!  |
|  | safeCakeTransfer | Internal  |  |   | |
|  | setDevAddress | Public  |   |  |NO!  |
|  | setDevShare | Public  |   |  |NO!  |
|  | setShareTo | Public  |   |  |NO!  |
|||||
| **CROToken** | Implementation | BEP20 |||
|  | mint | Public  |   |  | onlyOwner |
|  | safeCakeTransfer | Public  |   |  | onlyOwner |

```



```

| L | burn | Public  | ! | NO! |
| L | delegates | External  | ! | NO! |
| L | delegate | External  | ! | NO! |
| L | delegateBySig | External  | ! | NO! |
| L | getCurrentVotes | External  | ! | NO! |
| L | getPriorVotes | External  | ! | NO! |
| L | _delegate | Internal  |  |  |
| L | _moveDelegates | Internal  |  |  |
| L | _writeCheckpoint | Internal  |  |  |
| L | safe32 | Internal  |  |  |
| L | getChainId | Internal  |  |  |
|||||
| **CROVault** | Implementation | Ownable, Pausable |||
| L | <Constructor> | Public  | ! | NO! |
| L | deposit | External  | ! | whenNotPaused notContract |
| L | withdrawAll | External  | ! | notContract |
| L | harvest | External  | ! | notContract whenNotPaused |
| L | setAdmin | External  | ! | onlyOwner |
| L | setTreasury | External  | ! | onlyOwner |
| L | setPerformanceFee | External  | ! | onlyAdmin |
| L | setCallFee | External  | ! | onlyAdmin |
| L | setWithdrawFee | External  | ! | onlyAdmin |
| L | setWithdrawFeePeriod | External  | ! | onlyAdmin |
| L | emergencyWithdraw | External  | ! | onlyAdmin |
| L | inCaseTokensGetStuck | External  | ! | onlyAdmin |
| L | pause | External  | ! | onlyAdmin whenNotPaused |
| L | unpause | External  | ! | onlyAdmin whenPaused |
| L | calculateHarvestCakeRewards | External  | ! | NO! |
| L | calculateTotalPendingCakeRewards | External  | ! | NO! |
| L | getPricePerFullShare | External  | ! | NO! |
| L | withdraw | Public  | ! | notContract |
| L | available | Public  | ! | NO! |
| L | balanceOf | Public  | ! | NO! |
| L | _earn | Internal  |  |  |
| L | _isContract | Internal  |  |  |
|||||
| **MMFFactory** | Implementation | IPancakeFactory |||
| L | <Constructor> | Public  | ! | NO! |
| L | allPairsLength | External  | ! | NO! |
| L | createPair | External  | ! | NO! |
| L | setFeeTo | External  | ! | NO! |
| L | setExchangeFee | External  | ! | NO! |
| L | getExchangeFee | External  | ! | NO! |
| L | setFeeShare | External  | ! | NO! |
| L | setFeeAdmin | External  | ! | NO! |
|||||
| **MMFPair** | Implementation | IPancakePair, PancakeERC20 |||
| L | getReserves | Public  | ! | NO! |
| L | _safeTransfer | Private  |  |  |
| L | <Constructor> | Public  | ! |  |

```



```

|  | initialize | External  | !  | NO  |
|  | _update | Private  |  |  |
|  | _mintFee | Private  |  |  |
|  | mint | External  | !  | lock |
|  | burn | External  | !  | lock |
|  | swap | External  | !  | lock |
|  | skim | External  | !  | lock |
|  | sync | External  | !  | lock |
|||||
| **MMFRouter** | Implementation | IPancakeRouter02 |||
|  | <Constructor> | Public  | !  | NO!  |
|  | <Receive Ether> | External  | !  | NO  |
|  | _addLiquidity | Internal  |  |  |
|  | addLiquidity | External  | !  | ensure |
|  | addLiquidityETH | External  | !  | ensure |
|  | removeLiquidity | Public  | !  | ensure |
|  | removeLiquidityETH | Public  | !  | ensure |
|  | removeLiquidityWithPermit | External  | !  | NO!  |
|  | removeLiquidityETHWithPermit | External  | !  | NO!  |
|  | removeLiquidityETHSupportingFeeOnTransferTokens | Public  | !  | ensure |
|  | removeLiquidityETHWithPermitSupportingFeeOnTransferTokens | External  | !  | NO!  |
|  | _swap | Internal  |  |  |
|  | swapExactTokensForTokens | External  | !  | ensure |
|  | swapTokensForExactTokens | External  | !  | ensure |
|  | swapExactETHForTokens | External  | !  | ensure |
|  | swapTokensForExactETH | External  | !  | ensure |
|  | swapExactTokensForETH | External  | !  | ensure |
|  | swapETHForExactTokens | External  | !  | ensure |
|  | _swapSupportingFeeOnTransferTokens | Internal  |  |  |
|  | swapExactTokensForTokensSupportingFeeOnTransferTokens | External  | !  | ensure |
|  | swapExactETHForTokensSupportingFeeOnTransferTokens | External  | !  | ensure |
|  | swapExactTokensForETHSupportingFeeOnTransferTokens | External  | !  | ensure |
|  | quote | Public  | !  | NO  |
|  | getAmountOut | Public  | !  | NO  |
|  | getAmountIn | Public  | !  | NO  |
|  | getAmountsOut | Public  | !  | NO  |
|  | getAmountsIn | Public  | !  | NO  |
|||||
| **MMFLibrary** | Library | |||
|  | sortTokens | Internal  |  |  |
|  | pairFor | Internal  |  |  |
|  | getReserves | Internal  |  |  |
|  | quote | Internal  |  |  |
|  | getAmountOut | Internal  |  |  |
|  | getAmountIn | Internal  |  |  |
|  | getAmountsOut | Internal  |  |  |
|  | getAmountsIn | Internal  |  |  |
|||||
|  | <Constructor> | Public  | !  | !

```



| **\*\*SmartChefFactory\*\*** | Implementation | Ownable |||

| <sup>L</sup> | **<Constructor>** | Public   |  



```

|  | allPoolsLength | External  |  |  |  | NO  |  |
|  | deployPool | External  |  |  |  |  | onlyOwner |
|||||

```

```

| **SmartChefInitializable** | Implementation | Ownable, ReentrancyGuard ||| | |
|  | <Constructor> | Public  |  |  | NO  |  |
|  | initialize | External  |  |  | NO  |  |
|  | deposit | External  |  |  | nonReentrant |
|  | withdraw | External  |  |  | nonReentrant |
|  | emergencyWithdraw | External  |  |  | nonReentrant |
|  | emergencyRewardWithdraw | External  |  |  | onlyOwner |
|  | recoverWrongTokens | External  |  |  | onlyOwner |
|  | stopReward | External  |  |  | onlyOwner |
|  | updatePoolLimitPerUser | External  |  |  | onlyOwner |
|  | updateRewardPerBlock | External  |  |  | onlyOwner |
|  | updateStartAndEndBlocks | External  |  |  | onlyOwner |
|  | pendingReward | External  |  |  | NO  |  |
|  | _updatePool | Internal  |  |  |  |
|  | _getMultiplier | Internal  |  |  |  |

```

# InterFi

## Smart Contract Security Audit



# Smart Contract – Software Analysis

## Function Signatures

```

26465826 => setCallFee(uint256)
32749461 => getReserves(address,address,address)
74496190 => setMigrator(IMigratorChef)
ef171420 => migrate(IBEP20)
5ffe6146 => updateMultiplier(uint256)
01f8a976 => updateRewardPerBlock(uint256)
081e3eda => poolLength()
0812e2c5 => add(uint256,IBEP20,bool)
64482f79 => set(uint256,uint256,bool)
9b9c4477 => updateStakingPool()
454b0608 => migrate(uint256)
95f86d80 => migrateControl(address)
8dbb1e3a => getMultiplier(uint256,uint256)
1175a1dd => pendingCake(uint256,address)
630b5ba1 => massUpdatePools()
51eb05a6 => updatePool(uint256)
e2bbb158 => deposit(uint256,uint256)
441a3e70 => withdraw(uint256,uint256)
41441d3b => enterStaking(uint256)
1058d281 => leaveStaking(uint256)
5312ea8e => emergencyWithdraw(uint256)
a2e6ddcc => safeCakeTransfer(address,uint256)
d0d41fe1 => setDevAddress(address)
03c0fa01 => setDevShare(uint256)
bf1bbdae => setShareTo(address)
42966c68 => burn(uint256)
587cde1e => delegates(address)
5c19a95c => delegate(address)
c3cda520 => delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32)
b4b5ea57 => getCurrentVotes(address)
782d6fe1 => getPriorVotes(address,uint256)
a28a42b3 => _delegate(address,address)
955f9fd8 => _moveDelegates(address,address,uint256)
ee59e77f => _writeCheckpoint(address,uint32,uint256,uint256)
869d1f83 => safe32(uint256,string)
3408e470 => getChainId()
b6b55f25 => deposit(uint256)
853828b6 => withdrawAll()
4641257d => harvest()
704b6c02 => setAdmin(address)
f0f44260 => setTreasury(address)
70897b23 => setPerformanceFee(uint256)
b6ac642a => setWithdrawFee(uint256)

```



1efac1b8 => setWithdrawFeePeriod(uint256)



```

db2e21bc => emergencyWithdraw()
def68a9c => inCaseTokensGetStuck(address)
8456cb59 => pause()
3f4ba83a => unpause()
9d72596b => calculateHarvestCakeRewards()
58ebceb6 => calculateTotalPendingCakeRewards()
77c7b8fc => getPricePerFullShare()
2e1a7d4d => withdraw(uint256)
48a0d754 => available()
722713f7 => balanceOf()
6f48813d => _earn()
7d48441f => _isContract(address)
574f2ba3 => allPairsLength()
c9c65396 => createPair(address,address)
f46901ed => setFeeTo(address)
692eb56f => setExchangeFee(address,uint256)
5f4153f7 => getExchangeFee(address)
b3cba4a2 => setFeeShare(uint256)
6eb2d031 => setFeeAdmin(address)
0902f1ac => getReserves()
26e6cdde => _safeTransfer(address,address,uint256)
485cc955 => initialize(address,address)
7dc0a1fa => _update(uint256,uint256,uint112,uint112)
f65d5f86 => _mintFee(uint112,uint112)
6a627842 => mint(address)
89afcb44 => burn(address)
022c0d9f => swap(uint256,uint256,address,bytes)
bc25cf77 => skim(address)
fff6cae9 => sync()
6d7746bc => _addLiquidity(address,address,uint256,uint256,uint256,uint256)
e8e33700 => addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256)
f305d719 => addLiquidityETH(address,uint256,uint256,uint256,address,uint256)
baa2abde => removeLiquidity(address,address,uint256,uint256,uint256,address,uint256)
02751cec => removeLiquidityETH(address,uint256,uint256,uint256,address,uint256)
2195995c =>
removeLiquidityWithPermit(address,address,uint256,uint256,uint256,address,uint256,bool,uint8,byte
s3 2,bytes32)
ded9382a =>
removeLiquidityETHWithPermit(address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes32,byt es32)
af2979eb =>
removeLiquidityETHSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,uint256) 5b0d5984
=>
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,u
int256,bool,uint8,bytes32,bytes32)
f5901d4d = _swap(uint256[],address[],address)
>
38ed1739 = swapExactTokensForTokens(uint256,uint256,address[],address,uint256)
>

```





```
8803dbee = swapTokensForExactTokens(uint256,uint256,address[],address,uint256)
>
7ff36ab5 = swapExactETHForTokens(uint256,address[],address,uint256)
>
4a25d94a = swapTokensForExactETH(uint256,uint256,address[],address,uint256)
>
```



```

18cbafe5 = swapExactTokensForETH(uint256,uint256,address[],address,uint256)
>
fb3bdb41 = swapETHForExactTokens(uint256,address[],address,uint256)
>
d1c474e3 => _swapSupportingFeeOnTransferTokens(address[],address)
5c11d795 =>
swapExactTokensForTokensSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
b6f9de95 => swapExactETHForTokensSupportingFeeOnTransferTokens(uint256,address[],address,uint256)
791ac947 =>
swapExactTokensForETHSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
ad615dec => quote(uint256,uint256,uint256)
054d50d4 => getAmountOut(uint256,uint256,uint256)
85f8c259 => getAmountIn(uint256,uint256,uint256)
d06ca61f => getAmountsOut(uint256,address[])
1f00ca74 => getAmountsIn(uint256,address[])
544caa56 => sortTokens(address,address)
6d91c0e2 => pairFor(address,address,address)
cef496b1 => getAmountOut(address,address,uint256,uint256,uint256)
f1dfeabe => getAmountIn(address,address,uint256,uint256,uint256)
bb7b9c76 => getAmountsOut(address,uint256,address[])
192128b2 => getAmountsIn(address,uint256,address[])
efde4e64 => allPoolsLength()
eb8bfb05 => deployPool(IBEP20,IBEP20,uint256,uint256,uint256,uint256,address) 48cf8750
=> initialize(IBEP20,IBEP20,uint256,uint256,uint256,uint256,address) 3279beab =>
emergencyRewardWithdraw(uint256)
3f138d4b => recoverWrongTokens(address,uint256)
80dc0672 => stopReward()
a0b40905 => updatePoolLimitPerUser(bool,uint256)
9513997f => updateStartAndEndBlocks(uint256,uint256)
f40f0f52 => pendingReward(address)
2061e1e5 => _updatePool()
8e356d7a => _getMultiplier(uint256,uint256)

```



## Smart Contract – Manual Analysis

❖ **Active smart contract owner privileges constitute an elevated impact to smart contracts' safety and security.**

❖ **CX1's smart contract MMFPair.sol has a low severity issue which may not create any functional vulnerability.**

Expected identifier, got 'Payable'

"severity": NULL

❖ **CX1 uses multiple imports from Yam Finance for governance. These codes have not been audited due to being out of scope.**

Smart Contract  
Security Audit



# Smart Contract – SWC Attacks

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

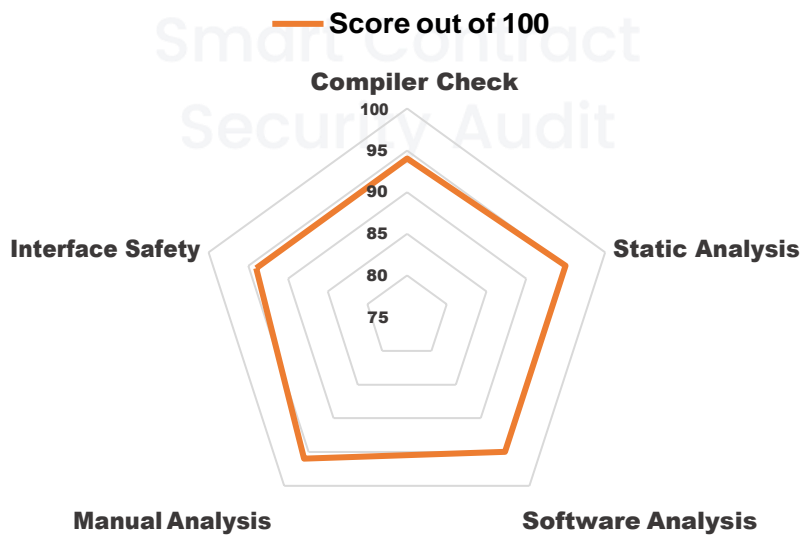


SWC-119	<b>Shadowing State Variables</b>	Passed
SWC-120	<b>Weak Sources of Randomness from Chain Attributes</b>	Passed
SWC-121	<b>Missing Protection against Signature Replay Attacks</b>	Passed
SWC-122	<b>Lack of Proper Signature Verification</b>	Passed
SWC-123	<b>Requirement Violation</b>	Passed
SWC-124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



# Smart Contract - Risk Status & Radar Chart

Risk Severity	Status
<b>! Critical</b>	<b>None critical severity issues identified</b>
<b>! High</b>	<b>None high severity issues identified</b>
<b>! Medium</b>	<b>None medium severity issues identified</b>
<b>! Low</b>	<b>2 low severity issues identified</b>
<b>Verified</b>	<b>54 functions and instances verified and checked</b>
<b>Safety Score</b>	<b>94 out of 100</b>



## 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.**

**CX1's smart contract source code has **LOW RISK SEVERITY**.**

**CX1 has **PASSED** the smart contract audit.**

InterFi

Smart Contract  
Security Audit

### 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, e.g., InterFi.**



## 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/interfiaudit> .....**

Smart Contract  
Security Audit





{{gINTERFINETWORK

RELENTLESSLY SECURING THE PUBLIC BLOCKCHAIN | MADE IN CANADA 🇨🇦