



# SOLIDProof

*Bring trust into your projects*

**Blockchain Security | Smart Contract Audits | KYC  
Development | Marketing**

MADE IN GERMANY

## COIF

# Audit

**Security Assessment**  
**20. June, 2023**

**For**

© IF COIF

**Community Investment Fund**



**SolidProof\_io**



**@solidproof\_io**

Disclaimer	3
Description	5
Project Engagement	5
Logo	5
Contract Link	5
Methodology	8
Used Code from other Frameworks/Smart Contracts (direct imports)	9
Tested Contract Files	10
Source Lines	11
Risk Level	11
Capabilities	12
Inheritance Graph	13
CallGraph	14
Scope of Work/Verify Claims	15
Modifiers and public functions	24
Source Units in Scope	26
Critical issues	27
High issues	27
Medium issues	27
Low issues	28
Informational issues	28
Audit Comments	28
Alleviation:	28
SWC Attacks	30

# Disclaimer

SolidProof.io reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team. SolidProof.io do not cover testing or auditing the integration with external contract or services (such as Unicrypt, Uniswap, PancakeSwap etc’...)

**SolidProof.io Audits do not provide any warranty or guarantee regarding the absolute bug- free nature of the technology analyzed, nor do they provide any indication of the technology proprietors. SolidProof Audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.**

SolidProof.io Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology. Blockchain technology and cryptographic assets present a high level of ongoing risk. SolidProof’s position is that each company and individual are responsible for their own due diligence and continuous security. SolidProof in no way claims any guarantee of security or functionality of the technology we agree to analyze.

Version	Date	Description
1.0	19. June 2023	<ul style="list-style-type: none"><li>• Layout project</li><li>• Automated- /Manual-Security Testing</li><li>• Summary</li></ul>
1.1	20. June 2023	<ul style="list-style-type: none"><li>• Reaudit</li></ul>

**Note** - This Audit report comprises a security analysis of the **COIF** smart contracts. This analysis did not include functional testing (or unit testing) of the contract’s logic.

## **Network**

Binance Smart Chain

## **Website**

[www.coif.capital](http://www.coif.capital)

## **Telegram**

<https://t.me/coifcapital>

## **Twitter**

<https://twitter.com/coifcapital>

## **Medium**

<https://medium.com/@coif.capital>

## **Youtube**

<https://www.youtube.com/@Coif.Capital>

## Description

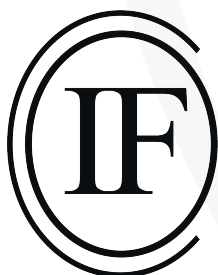
COIF.CAPITAL is a crypto project that combines the idea of a traditional investment fund, such as an equity fund, with cryptocurrencies and adds many additional features.

The investors do not invest directly in the fund, as is usual with conventional investment funds, but buy the project's own cryptocurrency COIF<sup>1</sup> and thereby automatically acquire the right to the shares of the fund's assets, regardless of the time of purchase. At the same time, they acquire the right to co-decide how the fund assets are managed. Among other things, they decide which share of the fund assets is paid out to the community<sup>2</sup> and when.

## Project Engagement

During the Date of 15 June 2023, **COIF Team** engaged Solidproof.io to audit smart contracts that they created. The engagement was technical and focused on identifying security flaws in the design and implementation of the contracts. They provided Solidproof.io with access to their code repository and whitepaper.

## Logo



## Contract Link

### v1.0

- [https://github.com/coifcapitalaudit/contract\\_private/commit/a19ca418640e97a138a324417722fd239566177a](https://github.com/coifcapitalaudit/contract_private/commit/a19ca418640e97a138a324417722fd239566177a)

### v1.1

- [https://github.com/coifcapitalaudit/contract/blob/main/coif\\_contract\\_mainchain\\_audit\\_v03.sol](https://github.com/coifcapitalaudit/contract/blob/main/coif_contract_mainchain_audit_v03.sol)
- Commit: 4c46d9c8a286a2028705c45877843543aa8427ae

**Note for Investors:** We only Audited a dividend-paying token contract for **COIF**(contract CommunityInvestmentFundContract and contract

DividendDistributor). However, If the project has other contracts (for example, a Presale contract, staking contract etc), and they were not provided to us in the audit scope, then we cannot comment on its security, and we are not responsible for it in any way.



# Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
<b>Critical</b>	9 - 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
<b>High</b>	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon as possible.
<b>Medium</b>	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
<b>Low</b>	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
<b>Informational</b>	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

# Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

## Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
  - i) Review of the specifications, sources, and instructions provided to SolidProof to make sure we understand the size, scope, and functionality of the smart contract.
  - ii) Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
  - iii) Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to SolidProof describe.
2. Testing and automated analysis that includes the following:
  - i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
  - ii) 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.



## Used Code from other Frameworks/Smart Contracts (direct imports)

Imported packages:

Dependency / Import Path	Count
@openzeppelin/contracts/access/Ownable.sol	1
@openzeppelin/contracts/token/ERC20/ERC20.sol	1
@openzeppelin/contracts/token/ERC20/IERC20.sol	1
@openzeppelin/contracts/utils/math/SafeMath.sol	1
@uniswap/v2-core/contracts/interfaces/IUniswapV2Factory.sol	1
@uniswap/v2-core/contracts/interfaces/IUniswapV2Pair.sol	1
@uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router01.sol	1
@uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router02.sol	1

## Tested Contract Files

This audit covered the following files listed below with a SHA-1 Hash.

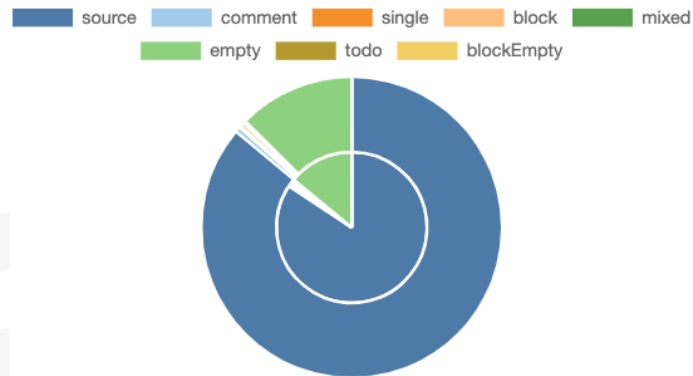
*A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.*

### v1.1

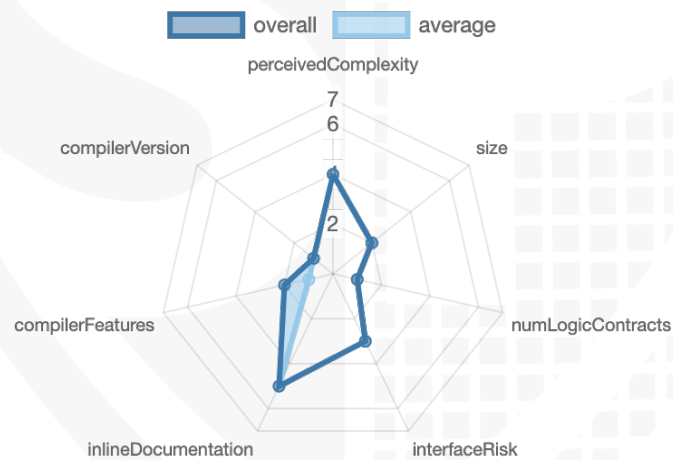
File Name	SHA-1 Hash
contracts/ coif_contract_mainchain_audit_ v03.sol	0f944fb7b31ed71dc6d3ade877d574 3fa7694c25

# Metrics

## Source Lines v1.0



## Risk Level v1.0



# Capabilities

## Components

 Contracts	 Libraries	 Interfaces	 Abstract
2	0	1	0

### Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.







 Public	 Payable
67	1

External	Internal	Private	Pure	View
24	70	6	0	14

### StateVariables

Total	 Public
85	64

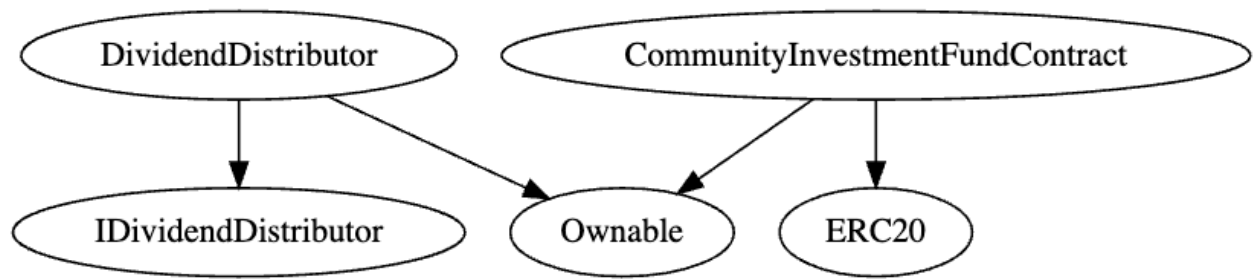
### Capabilities

Solidity Versions observed	 Experimental Features	 Can Receive Funds	 Uses Assembly	 Has Destroyable Contracts
 ^0.8.17		 yes		

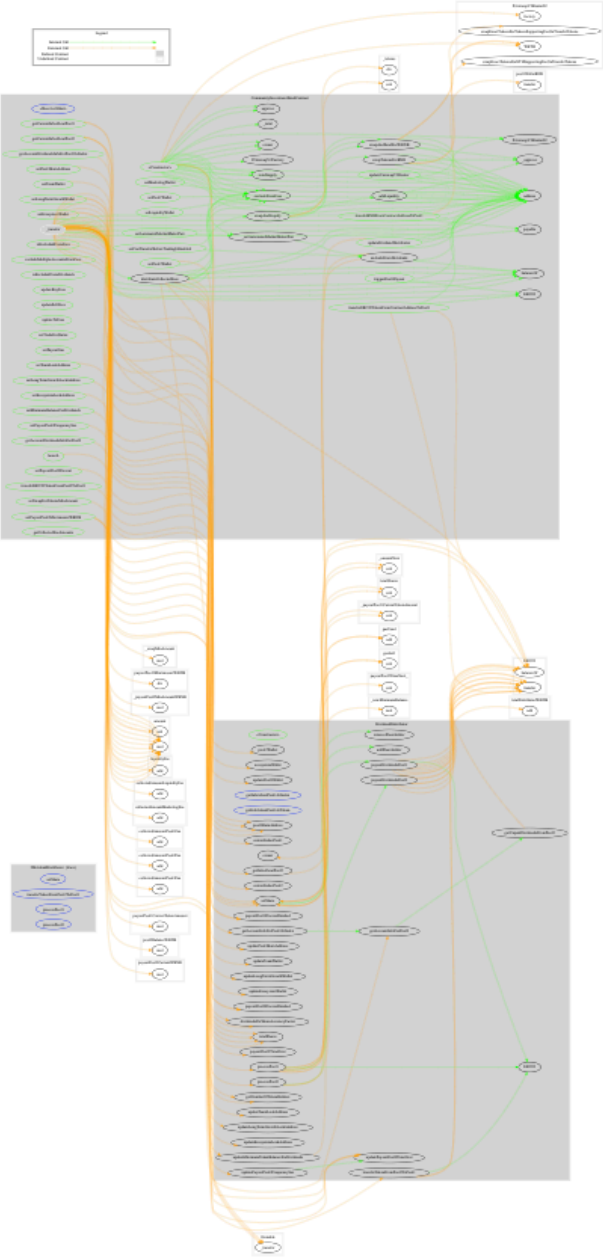
 Transfers ETH	 Low-Level Calls	 DelegateCall	 Uses Hash Functions	 ECRRecover	 New/Create/Create2
 yes					 yes →  NewContract:DividendDistributor

 TryCatch	 Unchecked
 yes	

## Inheritance Graph v1.0



CallGraph  
v1.0



## Scope of Work/Verify Claims

The above token Team provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract (usual the same name as team appended with .sol).

We will verify the following claims:

1. Is contract an upgradeable
2. Correct implementation of Token standard
3. Deployer cannot mint any new tokens
4. Deployer cannot burn or lock user funds
5. Deployer cannot pause the contract
6. Deployer cannot set fees
7. Deployer cannot blacklist/antisnipe addresses
8. Overall checkup (Smart Contract Security)

## Is contract an upgradeable

Name	
Is contract an upgradeable?	No





## Correct implementation of Token standard

ERC20				
Function	Description	Exist	Tested	Verified
TotalSupply	Provides information about the total token supply	✓	✓	✓
BalanceOf	Provides account balance of the owner's account	✓	✓	✓
Transfer	Executes transfers of a specified number of tokens to a specified address	✓	✓	✓
TransferFrom	Executes transfers of a specified number of tokens from a specified address	✓	✓	✓
Approve	Allow a spender to withdraw a set number of tokens from a specified account	✓	✓	✓
Allowance	Returns a set number of tokens from a spender to the owner	✓	✓	✓

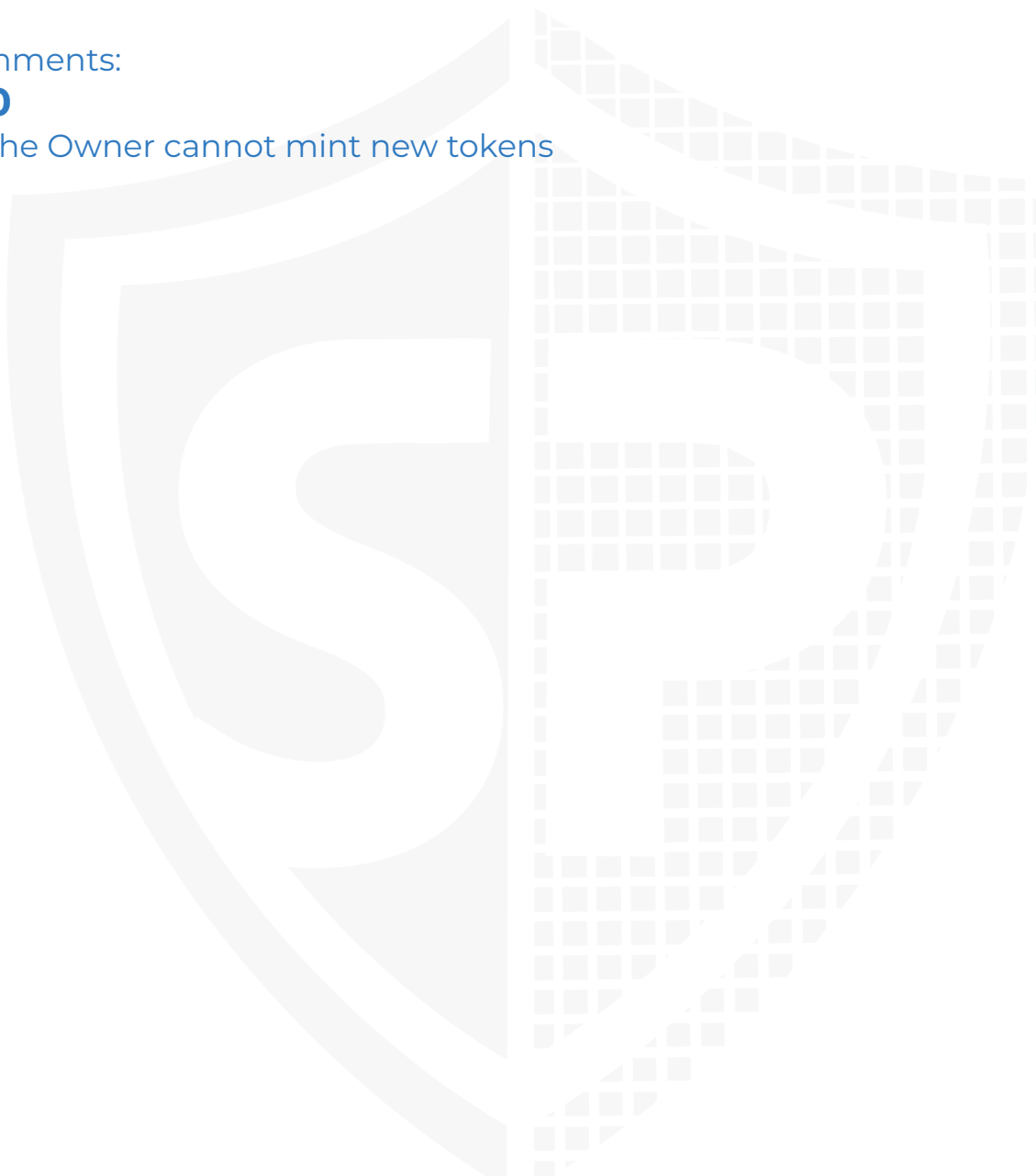
## Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint	✓	✓	✓
Max / Total Supply	100_000_000		

Comments:

**v1.0**

- The Owner cannot mint new tokens



## Deployer cannot burn or lock user funds

Name	Exist	Tested	Status
Deployer cannot lock	—	—	—
Deployer cannot burn	—	—	—



## Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause	—	—	—



## Deployer cannot set fees

Name	Exist	Tested	Status
Deployer cannot set fees over 25%	✓	✓	✓
Deployer cannot set fees to nearly 100% or to 100%	✓	✓	✓

Comments:

### v1.0

- There are total 5 fee components and their combined maximum fees can be 25%

## Deployer can blacklist/antisnipe addresses

Name	Exist	Tested	Status
Deployer cannot blacklist/antisnipe addresses	—	—	—



## Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

### Legend

Attribute	Symbol
Verified / Checked	✓
Partly Verified	⚠
Unverified / Not checked	✗
Not available	—

# Modifiers and public functions

## v1.0

<ul style="list-style-type: none"> <li>updateDividendDistributor</li> <li>onlyOwner</li> <li>setLiquidityWallet</li> <li>onlyOwner</li> <li>setMarketingWallet</li> <li>onlyOwner</li> <li>setPool1Wallet</li> <li>onlyOwner</li> <li>setPool3Wallet</li> <li>onlyOwner</li> <li>setPool3BurnAddress</li> <li>onlyOwner</li> <li>setTeamWallet</li> <li>onlyOwner</li> <li>setLongTermGrowthWallet</li> <li>onlyOwner</li> <li>setEcosystemWallet</li> <li>onlyOwner</li> <li>setTeamLockAddress</li> <li>onlyOwner</li> <li>setLongTermGrowthLockAddress</li> <li>onlyOwner</li> <li>setEcosystemLockAddress</li> <li>onlyOwner</li> <li>setSwapFeeTokensMinAmount</li> <li>onlyOwner</li> <li>updateUniswapV2Router</li> <li>onlyOwner</li> <li>excludeFromFees</li> <li>onlyOwner</li> <li>excludeMultipleAccountsFromFees</li> <li>onlyOwner</li> <li>setAutomatedMarketMakerPair</li> <li>onlyOwner</li> <li>excludeFromDividends</li> <li>onlyOwner</li> <li>updateBuyFees</li> <li>onlyOwner</li> <li>updateSellFees</li> <li>onlyOwner</li> <li>updateTxFees</li> <li>onlyOwner</li> <li>setTradeFeeStatus</li> <li>onlyOwner</li> <li>setPayoutGas</li> <li>onlyOwner</li> </ul>	<ul style="list-style-type: none"> <li>setPayoutPool2Percent</li> <li>onlyOwner</li> <li>setPayoutPool2MinAmountWBNB</li> <li>onlyOwner</li> <li>setMinimumBalanceForDividends</li> <li>onlyOwner</li> <li>setPayoutPool2FrequencySec</li> <li>onlyOwner</li> <li>triggerPool1Payout</li> <li>onlyOwner</li> <li>launch</li> <li>onlyOwner</li> <li>setCanTransferBeforeTradingIsEnabled</li> <li>onlyOwner</li> <li>transferERC20TokenFromPool2ToPool1</li> <li>onlyOwner</li> <li>transferERC20TokenFromContractAddressToPool1</li> <li>onlyOwner</li> <li>transferBNBFromContractAddressToPool1</li> <li>onlyOwner</li> </ul>	<ul style="list-style-type: none"> <li>setShare</li> <li>onlyOwner</li> <li>transferTokenFromPool2ToPool1</li> <li>onlyOwner</li> <li>processPool1</li> <li>onlyOwner</li> <li>processPool2</li> <li>onlyOwner</li> <li>updatePayoutPool2FrequencySec</li> <li>onlyOwner</li> <li>updatePayoutPool2TimeNext</li> <li>onlyOwner</li> <li>updateMinimumTokenBalanceForDividends</li> <li>onlyOwner</li> <li>updatePool3Wallet</li> <li>onlyOwner</li> <li>updatePool3BurnAddress</li> <li>onlyOwner</li> <li>updateTeamWallet</li> <li>onlyOwner</li> <li>updateLongTermGrowthWallet</li> <li>onlyOwner</li> <li>updateEcosystemWallet</li> <li>onlyOwner</li> <li>updateTeamLockAddress</li> <li>onlyOwner</li> <li>updateLongTermGrowthLockAddress</li> <li>onlyOwner</li> <li>updateEcosystemLockAddress</li> <li>onlyOwner</li> </ul>
---	--	---



## Ownership Privileges

- ❖ The owner can update/change the following addresses -
  - Dividend distributor
  - Liquidity, Marketing, Pool1, and Pool3 wallets
  - Pool3 burn address and Team wallet
  - Long-Term Growth Wallet, Ecosystem Wallet address.
  - lock addresses: Team, Long Term Growth and Ecosystem
  - Uniswap V2 router address and AMM pair address.
- ❖ The owner can update/change the following values -
  - Buy, Sell, and Transaction fees, but cannot set it to more than 5% for each of the 5 fee components.
  - Enable/Disable transaction fees
  - Payout Gas
  - The payout from Pool2 percentage can be set up to 100%
  - Minimum balance for dividends to any arbitrary value
  - Pool2 frequency per second to any arbitrary value
  - The minimum amount of tokens for distribution of collected fees to liquidity, marketing and pool 1,2,3 wallets
- ❖ Other Privileges -
  - Include/Exclude accounts from fees
  - Manually trigger Pool1 payout
  - Enable Trading
  - Allow accounts to trade before the trading is enabled
  - Transfer any type of tokens to the pool wallet from the contract's balance of the token contract and poolDistributor

## Source Units in Scope

### v1.0

File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score
contracts/coifcontractmainchainauditv02.sol	2	1	1160	1004	857	7	729
Totals	2	1	1160	1004	857	7	729

### Legend

Attribute	Description
Lines	total lines of the source unit
nLines	normalised lines of the source unit (e.g. normalises functions spanning multiple lines)
nSLOC	normalised source lines of code (only source-code lines; no comments, no blank lines)
Comment Lines	lines containing single or block comments
Complexity Score	a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

# Audit Results

## Critical issues

**No critical issues**

## High issues

**No high issues**

## Medium issues

### Medium Issues Acknowledged

Issue	File	Type	Line	Description	Status
#1	Main	Liquidity will be credited to an EOA	722	The liquidity of the contract is automatically added to the DevWallet address which is not recommended because, in an extreme scenario, this can be used to drain liquidity from the contract.	ACK
#2	Main	Owner needs to Enable Trading	595	The trading needs to be enabled by the owner in order for regular users to transfer tokens. On the contrary, the owner can include addresses to a list and those addresses will be able to trade tokens. This functionality can be exploited in the following way, For example, there is a presale and the wallets used for the presale can be included in the list by the owner. All the tokens obtained can be consolidated into a final wallet address and facilitate trading and selling of the acquired tokens, then the last wallet address can be included in the list.	ACK

## Low issues

Issue	File	Type	Line	Description	Status
#1	Main	A floating pragma is set	—	The current pragma Solidity directive is „^0.8.17”.	Fixed
#2	Main	Missing Zero Address Validation (missing-zero-check)	168-229	Check that the address is not zero	Fixed
#3	Main	Missing Events Arithmetic	233, 472	Emit an event for critical parameter changes	Fixed
#4	Main	State variable visibility is not set	49, 58, 753, 764, 765	It is best practice to set the visibility of state variables explicitly	Fixed

## Informational issues

Issue	File	Type	Line	Description	Status
#1	Main	Unused return values	714	Ensure that all the return values of the function calls are used and handle both success and failure cases if needed by the business logic	Fixed
#2	Main	Error message is missing	277	Provide an error message for require statement	Fixed
#3	Main	NatSpec documentation missing	—	If you started to comment your code, also comment all other functions, variables etc.	ACK

## Audit Comments

We recommend you use the particular form of comments (NatSpec Format, Follow the link for more information <https://docs.soliditylang.org/en/latest/natspec-format.html>) for your contracts to provide rich documentation for functions, return variables and more. This helps investors to make clear what that variable, functions etc., do.

## Alleviation:

The COIF team have the following comments about the acknowledgement of the medium issues:

**Issue#1** - *“Initial liquidity (10,000,000 COIF + BNB) will be locked for 2 years directly after the public sale.*

*The newly added liquidity will also be locked at regular intervals. All locks of liquidity can be checked publicly. Locks will be renewed shortly after expiry. At any time, there will always be sufficient locked liquidity available, so that a complete drain would not even be theoretically possible. We are planning to use for locks:*

*<https://www.pinksale.finance/pinklock/create?chain=BSC>*

**Issue#2** - *"The procedure will be as follows:*

*The deployer of the contract can distribute the tokens at the beginning according to the allocation table. Only the Deployer and public and private sale contracts get the rights to transfer tokens. After the public sale, liquidity is added according to the allocation table and trading is activated immediately.*

*The reason for "Owner needs to Enable Trading" is to avoid no one adding liquidity at the unfavourable rate and thus makes it inconvenient to add initial liquidity at the planned rate.*

*Without liquidity, trading will not be possible. And as soon as liquidity is added, trading is also activated immediately. This will be communicated accordingly and can be checked on the blockchain."*

## **20. June 2023:**

- There is still an owner (The owner still has not renounced ownership)
- Read the whole report and modifiers section for more information

## SWC Attacks

ID	Title	Relationships	Status
<a href="#">SW C-1 36</a>	Unencrypted Private Data On-Chain	<a href="#">CWE-767: Access to Critical Private Variable via Public Method</a>	PASSED
<a href="#">SW C-1 35</a>	Code With No Effects	<a href="#">CWE-1164: Irrelevant Code</a>	PASSED
<a href="#">SW C-1 34</a>	Message call with hardcoded gas amount	<a href="#">CWE-655: Improper Initialization</a>	PASSED
<a href="#">SW C-1 33</a>	Hash Collisions With Multiple Variable Length Arguments	<a href="#">CWE-294: Authentication Bypass by Capture-replay</a>	PASSED
<a href="#">SW C-1 32</a>	Unexpected Ether balance	<a href="#">CWE-667: Improper Locking</a>	PASSED
<a href="#">SW C-1 31</a>	Presence of unused variables	<a href="#">CWE-1164: Irrelevant Code</a>	PASSED
<a href="#">SW C-1 30</a>	Right-To-Left-Override control character (U+202E)	<a href="#">CWE-451: User Interface (UI) Misrepresentation of Critical Information</a>	PASSED
<a href="#">SW C-1 29</a>	Typographical Error	<a href="#">CWE-480: Use of Incorrect Operator</a>	PASSED
<a href="#">SW C-1 28</a>	DoS With Block Gas Limit	<a href="#">CWE-400: Uncontrolled Resource Consumption</a>	PASSED

<a href="#">SW C-1 27</a>	Arbitrary Jump with Function Type Variable	<a href="#">CWE-695: Use of Low-Level Functionality</a>	<b>PASSED</b>
<a href="#">SW C-1 25</a>	Incorrect Inheritance Order	<a href="#">CWE-696: Incorrect Behavior Order</a>	<b>PASSED</b>
<a href="#">SW C-1 24</a>	Write to Arbitrary Storage Location	<a href="#">CWE-123: Write-what-where Condition</a>	<b>PASSED</b>
<a href="#">SW C-1 23</a>	Requirement Violation	<a href="#">CWE-573: Improper Following of Specification by Caller</a>	<b>PASSED</b>
<a href="#">SW C-1 22</a>	Lack of Proper Signature Verification	<a href="#">CWE-345: Insufficient Verification of Data Authenticity</a>	<b>PASSED</b>
<a href="#">SW C-1 21</a>	Missing Protection against Signature Replay Attacks	<a href="#">CWE-347: Improper Verification of Cryptographic Signature</a>	<b>PASSED</b>
<a href="#">SW C-1 20</a>	Weak Sources of Randomness from Chain Attributes	<a href="#">CWE-330: Use of Insufficiently Random Values</a>	<b>PASSED</b>
<a href="#">SW C-11 9</a>	Shadowing State Variables	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	<b>PASSED</b>
<a href="#">SW C-11 8</a>	Incorrect Constructor Name	<a href="#">CWE-665: Improper Initialization</a>	<b>PASSED</b>
<a href="#">SW C-11 7</a>	Signature Malleability	<a href="#">CWE-347: Improper Verification of Cryptographic Signature</a>	<b>PASSED</b>

<a href="#">SW C-11 6</a>	Timestamp Dependence	<a href="#">CWE-829: Inclusion of Functionality from Untrusted Control Sphere</a>	<b>PASSED</b>
<a href="#">SW C-11 5</a>	Authorization through tx.origin	<a href="#">CWE-477: Use of Obsolete Function</a>	<b>PASSED</b>
<a href="#">SW C-11 4</a>	Transaction Order Dependence	<a href="#">CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')</a>	<b>PASSED</b>
<a href="#">SW C-11 3</a>	DoS with Failed Call	<a href="#">CWE-703: Improper Check or Handling of Exceptional Conditions</a>	<b>PASSED</b>
<a href="#">SW C-11 2</a>	Delegatecall to Untrusted Callee	<a href="#">CWE-829: Inclusion of Functionality from Untrusted Control Sphere</a>	<b>PASSED</b>
<a href="#">SW C-11 1</a>	Use of Deprecated Solidity Functions	<a href="#">CWE-477: Use of Obsolete Function</a>	<b>PASSED</b>
<a href="#">SW C-11 0</a>	Assert Violation	<a href="#">CWE-670: Always-Incorrect Control Flow Implementation</a>	<b>PASSED</b>
<a href="#">SW C-1 09</a>	Uninitialized Storage Pointer	<a href="#">CWE-824: Access of Uninitialized Pointer</a>	<b>PASSED</b>
<a href="#">SW C-1 08</a>	State Variable Default Visibility	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	<b>PASSED</b>
<a href="#">SW C-1 07</a>	Reentrancy	<a href="#">CWE-841: Improper Enforcement of Behavioral Workflow</a>	<b>PASSED</b>
<a href="#">SW C-1 06</a>	Unprotected SELFDESTRUCT Instruction	<a href="#">CWE-284: Improper Access Control</a>	<b>PASSED</b>



<a href="#">SW</a> <a href="#">C-1</a> <a href="#">05</a>	Unprotected Ether Withdrawal	<a href="#">CWE-284: Improper Access Control</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">04</a>	Unchecked Call Return Value	<a href="#">CWE-252: Unchecked Return Value</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">03</a>	Floating Pragma	<a href="#">CWE-664: Improper Control of a Resource Through its Lifetime</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">02</a>	Outdated Compiler Version	<a href="#">CWE-937: Using Components with Known Vulnerabilities</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">01</a>	Integer Overflow and Underflow	<a href="#">CWE-682: Incorrect Calculation</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">00</a>	Function Default Visibility	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	<b>PASSED</b>

*Solid  
Proofed*

**Blockchain Security | Smart Contract Audits | KYC  
Development | Marketing**

  
MADE IN GERMANY