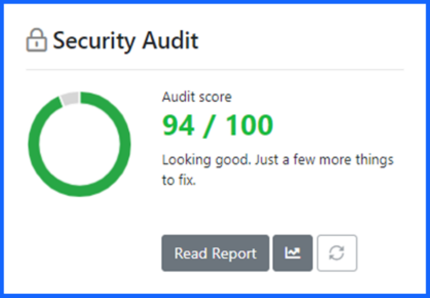
SkyNET.

SECURITY AUDIT REPORT FOR

ElonCola

COLA.sol



Security Audit

Audit score

94 / 100

Looking good. Just a few more things to fix.

CERTIFIED

Confidential

SMART CONTRACT SECURITY AUDIT

OF ElonCola (COLA.sol)

Audit Introduction [1]

Auditing Firm Skynet Audits Network

Audit Architecture Pro Audit

Language Solidity

Client Firm ElonCola

Website https://www.eloncola.com

Telegram https://t.me/ElonColaOfficial

Twitter https://twitter.com/ElonColaToken

Contract https://bscscan.com/address/0x681b76c338055d0590E48FBB972A345D32692331#code

Report Date April 29, 2022

About ElonCola

Bringing together a community of like-minded individuals from all over the globe with a common goal, earning Bitcoin passively. Our vision is quite simple, bitcoin is the future currency of the world. Why not start earning it years in advance by trading an asset and investing in a team you can trust? Our experienced team of fintech professionals have aggregated a long-term plan to not only grow but potentially dominate the bitcoin reflection market.

[1] SkyNET Audits. All right reserved.

Audit Summary

Skynet 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:

- ❖ ElonCola's solidity source code has **LOW RISK SEVERITY**
- ❖ ElonCola's smart contract has an **ACTIVE OWNERSHIP**
- ❖ Important owner privileges – **SET FEES**
- ❖ ElonCola's smart contract owner has multiple "Write Contract" privileges. Centralization risk correlated to the active owner is **LOW**

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:

0x681b76c338055d0590E48FBB972A345D32692331⛓

⛓ Blockchain: **Binance Smart Chain**

Table Of Contents

Audit Information

Audit Scope..5

Echelon Audit Standard

Audit Methodology

....6 Risk

Classification..8

Smart Contract Risk Assessment

Static

Analysis...9

Software Analysis

....14 Manual

Analysis...18 SWC

Attacks..22

Risk Status & Radar Chart..

24

Audit Summary

Auditor’s Verdict.. 25

Legal Advisory

Important Disclaimer...

Network..27

Audit Scope

Skynet was consulted by ElonCola 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:

- COLA.sol

Solidity Source Code On Blockchain (Verified Contract Source Code)

https://bscscan.com/token/0x681b76c338055d0590E48FBB972A345D32692331

Contract Name: ElonCola

Compiler Version: v0.6.12

Optimization Enabled: Yes with 200 runs

Audit Methodology

The scope of this report is to audit the smart contract source code of ElonCola. Skynet 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

- Re-entrancy
- Unhandled Exceptions
- Transaction Order Dependency
- Integer Overflow

Smart Contract Vulnerabilities

❖ Incorrect Inheritance Order

❖ Typographical Errors

❖ Requirement Violation

❖ Ownership Takeover

❖ Gas Limit and Loops

❖ Deployment Consistency

❖ Repository Consistency

Source Code Review

❖ Data Consistency

❖ Token Supply Manipulation

❖ Access Control and Authorization ❖

Operations Trail and Event Generation ❖

Assets Manipulation

Functional Assessment

❖ Liquidity Access

❖ Unrestricted Action

Skynet’s Echelon Pro Audit

The aim of Skynet's “Echelon” pro 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 the specifications, sources, and instructions provided to SkiNET 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. [4]

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, clarity, 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

[4] SkyNET Audits. All right reserved.

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

This level of vulnerabilities could be exploited easily and can lead to asset loss,

! High

important to fix, they carry an elevated risk of

smart contract manipulation, which can lead to

high-risk severity

! Medium

This level of vulnerabilities should be fixed, as

they carry an inherent risk of future exploits, and

! Low

hacks which may or may not impact the smart

contract execution. This level of vulnerability can

be ignored. They are code style violations and

! Informational

informational statements in the code. They may

not affect the smart contract execution

data loss, asset, or data manipulation. They

should be fixed right away. This level of

vulnerabilities are hard to exploit but very

Static Analysis

Symbol Meaning

□ � Function can be modified

□ � Function is payable

□ � Function is locked [5]

[5] SkyNET Audits. All right reserved.

□ � Function can be accessed

□ ❗ Important functionality

COLA	Implementation	ERC20, Ownable		
└	<Constructor>	Public □ ❗	□�	ERC20
└	<Receive Ether>	External ❗ □	□�	NO□ ❗
└	updateDividendTracker	Public □ ❗	�□	onlyOwner		└
updateUniswapV2Router	Public □ ❗	�□	onlyOwner		└	excludeFromFees
Public ❗ □	□�	onlyOwner		└	excludeMultipleAccountsFromFees	Public ❗ □
� □| onlyOwner		└	setMarketingWallet	External □ ❗	□�	onlyOwner		└
setBitcoinRewardsFee	External ❗ □	�□	onlyOwner		└	setLiquiditFee	External
□ ❗	� □| onlyOwner		└	setMarketingFee	External ❗ □| □�	onlyOwner		└
setAutomatedMarketMakerPair	Public ❗ □| □�	onlyOwner		└
_setAutomatedMarketMakerPair	Private � □	□�			└	updateGasForProcessing
Public ❗ □	□�	onlyOwner		└	updateClaimWait	External ❗ □	□�
onlyOwner		└	getClaimWait	External ❗ □| |NO ❗ □
└	getTotalDividendsDistributed	External □ ❗	|NO ❗ □|	└
isExcludedFromFees	Public □ ❗		NO ❗ □
└	withdrawableDividendOf	Public ❗ □		NO□ ❗
└	dividendTokenBalanceOf	Public ❗ □		NO□ ❗
└	excludeFromDividends	External ❗ □	□�	onlyOwner		└
getAccountDividendsInfo	External □ ❗	|NO□ ❗		└
getAccountDividendsInfoAtIndex	External □ ❗	|NO□ ❗		└
processDividendTracker	External ❗ □	�□	NO□ ❗		└	claim	External
❗ □| �□ |NO ❗ □|
└	getLastProcessedIndex	External □ ❗	|NO ❗ □|	└
getNumberOfDividendTokenHolders	External □ ❗	|NO ❗ □|	└	_transfer
Internal □�	� □	
└	swapAndSendToFee	Private � □	□�	
└	swapAndLiquify	Private � □	□�	
└	swapTokensForEth	Private � □	□�	

└	swapTokensForBitcoin	Private □�	□�	
└	addLiquidity	Private □�	□�	
└	swapAndSendDividends	Private □�	�□	
COLADividendTracker	Implementation	Ownable, DividendPayingToken |||	└	<Constructor>
Public □ ❗	□�	DividendPayingToken
└	_transfer	Internal □�	�□	
└	withdrawDividend	Public □ ❗	�□	NO ❗ □
└	excludeFromDividends	External ❗ □	□�	onlyOwner
└	updateClaimWait	External ❗ □	□�	onlyOwner
└	getLastProcessedIndex	External □ ❗		NO ❗ □
└	getNumberOfTokenHolders	External □ ❗		NO□ ❗

[6]| └ | getAccount | Public ❗ □| |NO□ ❗ |

| └ | getAccountAtIndex | Public ❗ □| |NO□ ❗ |

| └ | canAutoClaim | Private □� | | |

| └ | setBalance | External ❗ □| �□ | onlyOwner |

| └ | process | Public ❗ □| � □|NO ❗ □|

| └ | processAccount | Public ❗ □| �□ | onlyOwner |

||||||

| **Context** | Implementation | |||

| └ | _msgSender | Internal � □| | |

| └ | _msgData | Internal � □| | |

||||||

| **DividendPayingToken** | Implementation | ERC20, Ownable, DividendPayingTokenInterface, DividendPayingTokenOptionalInterface |||

| └ | <Constructor> | Public □ ❗ | □� | ERC20 |

| └ | distributeBitcoinDividends | Public □ ❗ | �□ | onlyOwner |

| └ | withdrawDividend | Public □ ❗ | �□ |NO ❗ □|

| └ | _withdrawDividendOfUser | Internal □� | □� | |

| └ | dividendOf | Public ❗ □| |NO□ ❗ |

| └ | withdrawableDividendOf | Public ❗ □| |NO□ ❗ |

| └ | withdrawnDividendOf | Public ❗ □| |NO ❗ □|

| └ | accumulativeDividendOf | Public ❗ □| |NO□ ❗ |

| └ | _transfer | Internal □� | � □| |

| └ | _mint | Internal � □| � □| |

| └ | _burn | Internal � □| �□ | |

| └ | _setBalance | Internal □� | □� | |

||||||

| **DividendPayingTokenInterface** | Interface | |||

| └ | dividendOf | External ❗ □| |NO□ ❗ |

| └ | withdrawDividend | External □ ❗ | □� |NO ❗ □|

||||||

| **DividendPayingTokenOptionalInterface** | Interface | |||

| └ | withdrawableDividendOf | External ❗ □| |NO□ ❗ |

| └ | withdrawnDividendOf | External □ ❗ | |NO ❗ □|

| └ | accumulativeDividendOf | 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 □� | �□ ||| └ | _mint | Internal

[6] SkyNET Audits. All right reserved.

```
� □| �□ | |
| └ | _burn | Internal � □| � □| |
| └ | _approve | Internal � □| □� | | | └ |
_beforeTokenTransfer | 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□ ❗ | ||||||
| **IterableMapping** | Library | ||| | └ | get |
Public ❗ □| |NO□ ❗ |
| └ | getIndexOfKey | Public □ ❗ | |NO□ ❗ | | └ |
getKeyAtIndex | Public □ ❗ | |NO□ ❗ | | └ | size |
Public □ ❗ | |NO ❗ □|
| └ | set | Public ❗ □| □� |NO□ ❗ |
| └ | remove | Public □ ❗ | □� |NO ❗ □|
||||||
| **IUniswapV2Factory** | Interface | ||| | └ | feeTo |
External ❗ □| |NO ❗ □|
| └ | feeToSetter | External □ ❗ | |NO□ ❗ | | └ | getPair |
External □ ❗ | |NO ❗ □| | └ | allPairs | External ❗ □|
|NO□ ❗ | | └ | allPairsLength | External □ ❗ | |NO ❗ □| |
└ | createPair | External ❗ □| � □|NO□ ❗ | | └ |
setFeeTo | External ❗ □| □� |NO□ ❗ | | └ |
setFeeToSetter | External □ ❗ | �□ |NO ❗ □| ||||||
| **IUniswapV2Pair** | Interface | ||| | └ | name |
External □ ❗ | |NO ❗ □|
| └ | symbol | External □ ❗ | |NO□ ❗ |

| └ | decimals | External ❗ □| |NO□ ❗ |
| └ | totalSupply | External □ ❗ | |NO□ ❗ |
| └ | balanceOf | External □ ❗ | |NO ❗ □|
| └ | allowance | External □ ❗ | |NO ❗ □|
| └ | approve | External □ ❗ | �□ |NO ❗ □|
| └ | transfer | External ❗ □| □� |NO□ ❗ |
| └ | transferFrom | External ❗ □| �□ |NO ❗ □|
| └ | DOMAIN_SEPARATOR | External □ ❗ | |NO ❗ □|
| └ | PERMIT_TYPEHASH | External ❗ □| |NO□ ❗ |
| └ | nonces | External □ ❗ | |NO□ ❗ |
| └ | permit | External □ ❗ | □� |NO□ ❗ |
| └ | MINIMUM_LIQUIDITY | External ❗ □| |NO ❗ □|
| └ | factory | External □ ❗ | |NO ❗ □|
```

| └ | token0 | External □ ❗ | |NO□ ❗ |

| └ | token1 | External □ ❗ | |NO□ ❗ |

| └ | getReserves | External □ ❗ | |NO□ ❗ |

| └ | price0CumulativeLast | External ❗ □| |NO□ ❗ |

| └ | price1CumulativeLast | External ❗ □| |NO□ ❗ |

| └ | kLast | External ❗ □| |NO ❗ □|

| └ | mint | External □ ❗ | □� |NO ❗ □|

| └ | burn | External □ ❗ | □� |NO ❗ □|

| └ | swap | External □ ❗ | □� |NO ❗ □|

| └ | skim | External □ ❗ | □� |NO ❗ □|

| └ | sync | External □ ❗ | □� |NO ❗ □|

| └ | initialize | External ❗ □| �□ |NO□ ❗ |

||||||

| **IUniswapV2Router01** | Interface | |||

| └ | factory | External □ ❗ | |NO ❗ □|

| └ | WETH | External □ ❗ | |NO ❗ □|

| └ | addLiquidity | External ❗ □| �□ |NO ❗ □|

| └ | addLiquidityETH | External ❗ □| □� |NO□ ❗ |

| └ | removeLiquidity | External ❗ □| □� |NO□ ❗ |

| └ | removeLiquidityETH | External □ ❗ | □� |NO□ ❗ |

| └ | removeLiquidityWithPermit | External ❗ □| □� |NO□ ❗ |

| └ | removeLiquidityETHWithPermit | External □ ❗ | □� |NO ❗ □|

| └ | swapExactTokensForTokens | External ❗ □| �□ |NO ❗ □|

| └ | swapTokensForExactTokens | External ❗ □| �□ |NO ❗ □|

| └ | swapExactETHForTokens | External □ ❗ | � □|NO ❗ □|

| └ | swapTokensForExactETH | External □ ❗ | �□ |NO ❗ □|

| └ | swapExactTokensForETH | External □ ❗ | �□ |NO ❗ □|

| └ | swapETHForExactTokens | External □ ❗ | � □|NO ❗ □|

| └ | quote | External ❗ □| |NO ❗ □|

| └ | getAmountOut | External ❗ □| |NO ❗ □|

| └ | getAmountIn | External □ ❗ | |NO□ ❗ |

| └ | getAmountsOut | External ❗ □| |NO□ ❗ |

| └ | getAmountsIn | External ❗ □| |NO ❗ □|

||||||

| **IUniswapV2Router02** | Interface | IUniswapV2Router01 |||

| └ | removeLiquidityETHSupportingFeeOnTransferTokens | External □ ❗ | □� |NO□ ❗ || └ |
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens | External □ ❗ | �□ |NO ❗ □|

| └ | swapExactTokensForTokensSupportingFeeOnTransferTokens | External ❗ □| �□ |NO ❗ □|| └ |
swapExactETHForTokensSupportingFeeOnTransferTokens | External □ ❗ | □� |NO ❗ □|| └ |
swapExactTokensForETHSupportingFeeOnTransferTokens | External □ ❗ | �□ |NO ❗ □| ||||||

| **Ownable** | Implementation | Context |||

| └ | <Constructor> | Public □ ❗ | □� |NO□ ❗ |

| └ | owner | Public ❗ □| |NO□ ❗ |

| └ | renounceOwnership | Public ❗ □| □� | onlyOwner |

| └ | transferOwnership | Public ❗ □| □� | onlyOwner |

||||||

| **SafeMath** | Library | |||

```
| └ | add | Internal � □| | |
| └ | sub | Internal � □| | |
| └ | sub | Internal � □| | |
| └ | mul | Internal � □| | |
| └ | div | Internal � □| | |
| └ | div | Internal � □| | |
| └ | mod | Internal � □| | |
| └ | mod | Internal � □| | |
||||||
| **SafeMathInt** | Library | |||
| └ | mul | Internal � □| | |
| └ | div | Internal � □| | |
| └ | sub | Internal � □| | |
| └ | add | Internal � □| | |
| └ | abs | Internal � □| | |
| └ | toUint256Safe | Internal � □| | |
||||||
| **SafeMathUint** | Library | |||
| └ | toInt256Safe | Internal � □| | |
```

Software Analysis

Function Signatures

39509351 => increaseAllowance(address,uint256) 43509138
=> div(int256,int256)
88bdd9be => updateDividendTracker(address)
65b8dbc0 => updateUniswapV2Router(address)
c0246668 => excludeFromFees(address,bool)
c492f046 => excludeMultipleAccountsFromFees(address[],bool) 5d098b38 =>
setMarketingWallet(address)
ce2fea33 => setBitcoinRewardsFee(uint256)
adefd90c => setLiquiditFee(uint256)
625e764c => setMarketingFee(uint256)
9a7a23d6 => setAutomatedMarketMakerPair(address,bool) a7f7b36f =>
_setAutomatedMarketMakerPair(address,bool) 871c128d =>
updateGasForProcessing(uint256)
e98030c7 => updateClaimWait(uint256)
a26579ad => getClaimWait()
30bb4cff => getTotalDividendsDistributed()
4fbee193 => isExcludedFromFees(address)
a8b9d240 => withdrawableDividendOf(address)
6843cd84 => dividendTokenBalanceOf(address)
31e79db0 => excludeFromDividends(address)
ad56c13c => getAccountDividendsInfo(address) f27fd254 =>
getAccountDividendsInfoAtIndex(uint256) 700bb191 =>
processDividendTracker(uint256)
4e71d92d => claim()
e7841ec0 => getLastProcessedIndex()
64b0f653 => getNumberOfDividendTokenHolders() 30e0789e

```
=> _transfer(address,address,uint256) a210621e =>
swapAndSendToFee(uint256)
173865ad => swapAndLiquify(uint256)
b28805f4 => swapTokensForEth(uint256)
4dd807ee => swapTokensForBitcoin(uint256)
9cd441da => addLiquidity(uint256,uint256)
818c19dc => swapAndSendDividends(uint256)
6a474002 => withdrawDividend()
09bbedde => getNumberOfTokenHolders()
fbcbc0f1 => getAccount(address)
5183d6fd => getAccountAtIndex(uint256)
77fdb837 => canAutoClaim(uint256)
e30443bc => setBalance(address,uint256)
ffb2c479 => process(uint256)
bc4c4b37 => processAccount(address,bool)
119df25f => _msgSender()
8b49d47e => _msgData()
edd6bf87 => distributeBitcoinDividends(uint256) 373de4aa
=> _withdrawDividendOfUser(address)


91b89fba => dividendOf(address)
aafd847a => withdrawnDividendOf(address)
27ce0147 => accumulativeDividendOf(address)
4e6ec247 => _mint(address,uint256)
6161eb18 => _burn(address,uint256)
ab86e0a6 => _setBalance(address,uint256)
06fdde03 => name()
95d89b41 => symbol()
313ce567 => decimals()
18160ddd => totalSupply()
70a08231 => balanceOf(address)
a9059cbb => transfer(address,uint256)
dd62ed3e => allowance(address,address)
095ea7b3 => approve(address,uint256)
23b872dd => transferFrom(address,address,uint256)
a457c2d7 => decreaseAllowance(address,uint256)
104e81ff => _approve(address,address,uint256)
cad3be83 => _beforeTokenTransfer(address,address,uint256) 268d8e2e =>
get(Map,address)
b45dad3d => getIndexOfKey(Map,address)
7596720f => getKeyAtIndex(Map,uint256)
b1b533f3 => size(Map)
6b06f325 => set(Map,address,uint256)
0eac8729 => remove(Map,address)
017e7e58 => feeTo()
094b7415 => feeToSetter()
e6a43905 => getPair(address,address)
1e3dd18b => allPairs(uint256)
574f2ba3 => allPairsLength()
c9c65396 => createPair(address,address)
f46901ed => setFeeTo(address)
a2e74af6 => setFeeToSetter(address)
3644e515 => DOMAIN_SEPARATOR()
```

30adf81f => PERMIT_TYPEHASH()
7ecebe00 => nonces(address)
d505accf => permit(address,address,uint256,uint256,uint8,bytes32,bytes32) ba9a7a56 =>
MINIMUM_LIQUIDITY()
c45a0155 => factory()
0dfe1681 => token0()
d21220a7 => token1()
0902f1ac => getReserves()
5909c0d5 => price0CumulativeLast()
5a3d5493 => price1CumulativeLast()
7464fc3d => kLast()
6a627842 => mint(address)
89afcb44 => burn(address)
022c0d9f => swap(uint256,uint256,address,bytes)
bc25cf77 => skim(address)
fff6cae9 => sync()
485cc955 => initialize(address,address)

ad5c4648 => WETH()
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,bytes3 2,bytes32)
ded9382a =>
removeLiquidityETHWithPermit(address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes32,byt es32)
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)
ad615dec => quote(uint256,uint256,uint256)
054d50d4 => getAmountOut(uint256,uint256,uint256)
85f8c259 => getAmountIn(uint256,uint256,uint256)
d06ca61f => getAmountsOut(uint256,address[])
1f00ca74 => getAmountsIn(uint256,address[])
af2979eb =>
removeLiquidityETHSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,uint256) 5b0d5984 =>
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,u
int256,bool,uint8,bytes32,bytes32)
5c11d795 =>
swapExactTokensForTokensSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256) b6f9de95 =>
swapExactETHForTokensSupportingFeeOnTransferTokens(uint256,address[],address,uint256) 791ac947 =>
swapExactTokensForETHSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256) 8da5cb5b =>
owner()
715018a6 => renounceOwnership()
f2fde38b => transferOwnership(address)
771602f7 => add(uint256,uint256)
b67d77c5 => sub(uint256,uint256)
e31bdc0a => sub(uint256,uint256,string)
c8a4ac9c => mul(uint256,uint256)
a391c15b => div(uint256,uint256)

```
b745d336 => div(uint256,uint256,string)
f43f523a => mod(uint256,uint256)
71af23e8 => mod(uint256,uint256,string)
bbe93d91 => mul(int256,int256)
adefc37b => sub(int256,int256)
a5f3c23b => add(int256,int256)
1b5ac4b5 => abs(int256)
744f7c7d => toUint256Safe(int256)
e823b9bf => toInt256Safe(uint256)
```

Manual Analysis

Function Description Tested Verdict provides information about the total token

Total Supply Balance Of Transfer

executes transfers of a specified number of

tokens to specified addresses **Passed**

Approve

allow a spender to withdraw a set number of

tokens from a specified accounts **Passed**

Allowance

returns a set number of tokens from a spender to

supply Yes **Passed**

the owner Yes **Passed** is an action in which the

provides account balance of the owner's

project buys back its

account Yes **Passed**

Buy Back

tokens from the existing

holders usually at a market

price

executes transfers of a specified number of

NA NA

executes the creation of a specified number of

Burn Mint

tokens and adds it to the total supply NA

circulating token supply adjusts (increases or

tokens to a burn address NA

Rebase

decreases) automatically

according to a token's price

fluctuations

stops specified wallets from interacting with the

NA NA

smart-contract function modules NA

stops or locks all function modules of the smart

Blacklist Lock

contract NA

Function Description	Tested	Verdict
executes transfers of a specified dividend token		

[7] SkyNET Audits. All right reserved.

Dividend

Airdrop

Max Transaction Max Wallet

Cooldown Timer Anti Bot

Anti Snipe

Transfer Ownership

Renounce

Ownership

to a specified address Yes **Passed**

executes transfers of a specified number of

tokens to a specified address NA NA

a non-whitelisted wallet can only transfer a

specified number of tokens NA NA

a non-whitelisted wallet can only hold a

specified number of tokens NA NA

functionality to limit the number of transactions

that a wallet can make within 24-hours NA NA

stops some or all bot wallets from interacting

with the smart contract NA NA

prevents bots from making transaction at

“addLiquidity” block NA NA

executes transfer of contract ownership to a

specified wallet Yes **Passed**

executes transfer of contract ownership to a

dead address Yes **Passed**

Best Practices ✅ ☐

- ❖ Owner cannot stop or pause the smart contract.
- ❖ Owner cannot lock or burn the user assets.
- ❖ Owner cannot mint tokens after initial contract creation/deployment.
- ❖ The smart contract utilizes “SafeMath” function to avoid common smart contract vulnerabilities.

```
string private _name = "ElonCola";
library SafeMath {
function add(uint256 a, uint256 b) internal pure returns (uint256) {
uint256 c = a + b;
require(c >= a, "SafeMath: addition overflow");
function sub(uint256 a, uint256 b) internal pure returns (uint256) {
return sub(a, b, "SafeMath: subtraction overflow");
uint256 c = a * b;
require(c / a == b, "SafeMath: multiplication overflow");
```

```
return c;
function div(uint256 a, uint256 b) internal pure returns (uint256) {
return div(a, b, "SafeMath: division by zero");
function mod(uint256 a, uint256 b) internal pure returns (uint256) {
return mod(a, b, "SafeMath: modulo by zero");
```

Note □⚠

❖ Active smart contract owner: 0xB39275240B0E72892D25e17abC67e63F9c838e55 ❖ ***Be aware that active smart contract owner privileges constitute an elevated impact to smart contract safety and security.***

❖ Smart contract owner can ***change transaction fees***. This function module can be used to impose extraordinary transaction fees. No arbitrary limit set.

```
function setBitcoinRewardsFee(uint256 value) external onlyOwner{
BitcoinRewardsFee = value;

function setLiquiditFee(uint256 value) external onlyOwner{
liquidityFee = value;

function setMarketingFee(uint256 value) external onlyOwner{
MarketingFee = value;
```

❖ The smart contract has a ***low severity issue*** which may or may not create any functional vulnerability.

```
{
        "resource": " /COLA.sol",
        "owner": "_generated_diagnostic_collection_name_#0",
        "severity": 8, (! Low Severity)
        " Expected token Semicolon got 'Identifier'",
        "source": "solc",
} [8]
```

[8] SkyNET Audits. All right reserved.

SWC Attacks

SWC ID Description Verdict SWC-101 Integer Overflow and Underflow **Passed SWC-102** Outdated Compiler Version**! Low 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 2 low severity issues identified

❖ Please Review Report

! Informational 1 informational severity issue identified ❖

Active Ownership

Verified 54 functions and instances verified and checked

Score 94 out of 100.

Auditor’s Verdict

Skynet 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.

❖ ElonCola’s smart contract source code has **LOW RISK SEVERITY**

❖ ElonCola’s smart contract has an **ACTIVE OWNERSHIP**

❖ ElonCola’s smart contract owner has multiple “Write Contract” privileges. Centralization risk correlated to the active owner is **LOW**

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 the 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

Skynet 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 Skynet's prior written consent.**

Skynet 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, Skynet 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 Skynet Network

Skynet Network provides intelligent blockchain solutions. Skynet is developing an ecosystem that is seamless and responsive. Some of our services: Blockchain Security, Token Launchpad, NFT Marketplace, etc. **Skynet'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.**

Skynet 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. **Skynet provides manual, static, and automatic smart contract analysis, to ensure that project is checked against known attacks and potential vulnerabilities.**

[9] SkyNET Audits. All right reserved.