TOKEN AUDIT

BELLA CIAO

Smart Contract Security Audit

Audit Report

SEP, 2021

BELLA CIAO

Table of contents

_Project Introduction	2
Auditing Methodologies applied:	3
Static Analysis	4
Auditing Tools	4
Tokenomics	5
Issues Checking	6
Severity Issue Categories	8
Issues Found	9
Automated Testing	13
Functional View	17
Inheritance Chart	18
Audit Findings Results	19

Project Introduction

Earn ETHEREUM as you HODL $BellaCiao token.Inspired by the Movie Series Money Heist / La Casa De Papel.
When the buyback function is turned on, tokens are bought back from the market, resulting in an immediate effect on the price.

Name	BELLA CIAO
Total Supply	ONE Billion
Type	BSC Token
Website	https://www.bellaciaotoken.com
Platform	Binance Smart Chain
Deployed Contract	0xD764add9fE609A3698C8314e28f279c5aF7f9a06

Token Audit Team performed a security audit for BELLA CIAO smart contracts during the period of Sep 16, 2021 to Sep 17, 2021.

The code for the audit was taken from following the official link:

https://bscscan.com/address/0xd764add9fe609a3698c8314e28f279c5af7f9a06#code

Auditing Methodologies applied:

- In this audit, we can review the code listed below.
- The overall quality of code.
- Whether the implementation of BEP 20 standards.
- Whether the code is secure.
- Gas Optimization
- Code is safe from reentrancy and other vulnerabilities

Manual Audit

- Manually analyzing the source code line-by-line in an attempt to identify security vulnerabilities.
- Gas Consumption and optimization
- Assessing the overall project structure, complexity & quality.
- Checking whether all the libraries used in the code of the latest version.

Automated Audit

- Projects can be Automated using these tools with Slither, Manticore, Sol Graph others.
- Performing Unit testing.
- Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.

➔ 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.

Static Analysis

Static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step a series of automated tools are used to test security of smart contracts.

Auditing Tools

Language: Solidity

Platform and tools: Slither, Manti-Core, VScode, Solhint, Solc-select, Solidity-coverage

Audit Aim: The focus of the audit was to verify whether the smart contract is secure, resilient, and working according to the standard specs. The audit activity can be categories into three types

- Security
- Sound Architecture
- Code Correctness and Quality

Tokenomics:

The Tokenomics were as follows (in number of tokens):

One Billion Total Supply

30% Pancake Swap Supply

Issues Checking

We have scanned this smart contract code for commonly known and more specific Vulnerabilities that are below listed:

SN	Issue Description	Status
1	Re-entrancy	Verified
2	Compiler errors	Verified
3	Timestamp Dependence	Verified
4	Unsafe external calls	Verified
5	Gas Limit and Loops	Verified
6	DoS with Block Gas Limit	Verified
7	Private user data leaks	Verified
8	Code clones, functionality duplication	Verified
9	Style guide violation	Verified
10	Costly Loop	Verified
11	Balance equality	Verified
12	Unchecked math	Verified

13	Integer overflow/underflow	Verified
14	Cross-function Race Condition	Verified
15	Fallback function security	Verified
16	Data Consistency	Verified
17	Balance equality	Verified
18	ERC20 API violation	Verified
19	Deployment Consistency	Verified
20	Arithmetic accuracy	Verified
21	Transaction-Ordering Dependence (TOD) / Front Running	Verified
22	Address hardcoded	Verified
23	Scoping and Declarations	Verified
24	Implicit visibility level	Verified
25	Call Depth Attack (deprecated)	Verified

Severity Issue Categories

Every issue in this report was assigned a severity level from the following:

Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	The issue affects the ability of the contract to compile or operate in a significant way.
Medium	Issues on this level could potentially bring problems and should eventually be fixed.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a significant impact on execution.
Informational	The issue has no impact on the contract's ability to operate.

Issues Found

- Critical severity issues - No Critical severity issues found.
- High severity issues - No Critical severity issues found.
- Medium severity issues - No medium severity issues found.
- Low severity issues - 1 low severity issues were found.
- Informational - 3 Informational severity issues were found.

High	Medium	Low	Informational
0	0	1	3

Low level severity issues

1. Missing zero address validation

Severity: Low

Description

When updating the marketing address, it should be checked for zero address. Otherwise, tokens/ETH sent to the zero address may be burnt forever.

```
ftrace | funcSig
function setFeeReceivers(address _autoLiquidityReceiver, address _marketingFeeReceiver) external authorized {
    autoLiquidityReceiver = _autoLiquidityReceiver;
    marketingFeeReceiver = _marketingFeeReceiver;
}
```

Suggestion:

Use a require statement to check for zero address when updating the marketing address

Informational level severity issues

1. Solidity naming conventions

Severity: Informational

Description:

In the contract, many function names were found to be starting with capital letters. Functions other than constructors should use mixedCase

```
Parameter DividendDistributor.setDistributionCriteria(uint256,uint256)._minPeriod (bellaciao.sol#253) is not in mixedCase
Parameter DividendDistributor.setDistributionCriteria(uint256,uint256)._minDistribution (bellaciao.sol#253) is not in mixedCase
Variable DividendDistributor._token (bellaciao.sol#205) is not in mixedCase
Variable DividendDistributor.BETH (bellaciao.sol#213) is not in mixedCase
Variable DividendDistributor.WBNB (bellaciao.sol#214) is not in mixedCase
Parameter BellaCiao.tradingStatus(bool)._status (bellaciao.sol#578) is not in mixedCase
Parameter BellaCiao.cooldownEnabled(bool,uint8)._status (bellaciao.sol#583) is not in mixedCase
Parameter BellaCiao.cooldownEnabled(bool,uint8)._interval (bellaciao.sol#583) is not in mixedCase
Parameter BellaCiao.setFees(uint256,uint256,uint256,uint256)._liquidityFee (bellaciao.sol#663) is not in mixedCase
Parameter BellaCiao.setFees(uint256,uint256,uint256,uint256)._reflectionFee (bellaciao.sol#663) is not in mixedCase
Parameter BellaCiao.setFees(uint256,uint256,uint256,uint256)._marketingFee (bellaciao.sol#663) is not in mixedCase
Parameter BellaCiao.setFees(uint256,uint256,uint256,uint256)._feeDenominator (bellaciao.sol#663) is not in mixedCase
Parameter BellaCiao.setFeeReceivers(address,address)._autoLiquidityReceiver (bellaciao.sol#672) is not in mixedCase
Parameter BellaCiao.setFeeReceivers(address,address)._marketingFeeReceiver (bellaciao.sol#672) is not in mixedCase
Parameter BellaCiao.setSwapBackSettings(bool,uint256)._enabled (bellaciao.sol#677) is not in mixedCase
Parameter BellaCiao.setSwapBackSettings(bool,uint256)._amount (bellaciao.sol#677) is not in mixedCase
Parameter BellaCiao.setTargetLiquidity(uint256,uint256)._target (bellaciao.sol#682) is not in mixedCase
Parameter BellaCiao.setTargetLiquidity(uint256,uint256)._denominator (bellaciao.sol#682) is not in mixedCase
Parameter BellaCiao.setDistributionCriteria(uint256,uint256)._minPeriod (bellaciao.sol#687) is not in mixedCase
Parameter BellaCiao.setDistributionCriteria(uint256,uint256)._minDistribution (bellaciao.sol#687) is not in mixedCase
Variable BellaCiao.BETH (bellaciao.sol#372) is not in mixedCase
Variable BellaCiao.WBNB (bellaciao.sol#373) is not in mixedCase
Variable BellaCiao.DEAD (bellaciao.sol#374) is not in mixedCase
Variable BellaCiao.ZERO (bellaciao.sol#375) is not in mixedCase
Constant BellaCiao._name (bellaciao.sol#377) is not in UPPER_CASE_WITH_UNDERSCORES
Constant BellaCiao._symbol (bellaciao.sol#378) is not in UPPER_CASE_WITH_UNDERSCORES
Constant BellaCiao._decimals (bellaciao.sol#379) is not in UPPER_CASE_WITH_UNDERSCORES
Variable BellaCiao._totalSupply (bellaciao.sol#381) is not in mixedCase
Variable BellaCiao._maxTxAmount (bellaciao.sol#382) is not in mixedCase
Variable BellaCiao._maxWalletToken (bellaciao.sol#385) is not in mixedCase
Variable BellaCiao._balances (bellaciao.sol#387) is not in mixedCase
Variable BellaCiao._allowances (bellaciao.sol#388) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

Suggestion:

Follow the Solidity: https://docs.soliditylang.org/en/v0.4.25/style-guide.html#naming-conventions

2. Incorrect versions of solidity

Severity: Informational

```
Pragma version^0.7.4 (bellaciao.sol#17) allows old versions
solc-0.7.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

Suggestion:

We recommend u can use latest version of solidity.

.

3. Public function that could be declared external

Severity: Informational

Description:

The following public functions that are never called by the contract

should be declared external to save gas

```
authorize(address) should be declared external:
        - Auth.authorize(address) (bellaciao.sol#106-108)
unauthorize(address) should be declared external:
        - Auth.unauthorize(address) (bellaciao.sol#113-115)
transferOwnership(address) should be declared external:
        - Auth.transferOwnership(address) (bellaciao.sol#134-138)
tradingStatus(bool) should be declared external:
        - BellaCiao.tradingStatus(bool) (bellaciao.sol#578-580)
cooldownEnabled(bool,uint8) should be declared external:
        - BellaCiao.cooldownEnabled(bool,uint8) (bellaciao.sol#583-586)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

Suggestion:

Use the external attribute for functions never called from the contract.

4. Outdated Openzeppelin lib version

Severity: Informational

Description:

The project uses old version OpenZeppelin lib.

Suggestion:

Consider updating the lib to get up-to-date improvements and patches

Automated Testing

We have to use Automated testing Slither. It is an Automated Analysis Tool in Smart Contract.

```
BellaCiao.swapBack() (bellaciao.sol#590-634) sends eth to arbitrary user
        Dangerous calls:
        - (tmpSuccess) = address(marketingFeeReceiver).call{gas: 30000,value: amountBNBMarketing}() (bellaciao.sol#618)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations

Reentrancy in BellaCiao._transferFrom(address,address,uint256) (bellaciao.sol#490-538):
        External calls:
        - swapBack() (bellaciao.sol#517)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(amountToSwap,0,path,address(this),block.timestamp) (bellaciao.sol#601-607)
                - distributor.deposit{value: amountBNBReflection}() (bellaciao.sol#617)
                - (tmpSuccess) = address(marketingFeeReceiver).call{gas: 30000,value: amountBNBMarketing}() (bellaciao.sol#618)
                - router.addLiquidityETH{value: amountBNBLiquidity}(address(this),amountToLiquify,0,0,autoLiquidityReceiver,block.timestamp) (bellaciao.sol#624-631)
        External calls sending eth:
        - swapBack() (bellaciao.sol#517)
                - distributor.deposit{value: amountBNBReflection}() (bellaciao.sol#617)
                - (tmpSuccess) = address(marketingFeeReceiver).call{gas: 30000,value: amountBNBMarketing}() (bellaciao.sol#618)
                - router.addLiquidityETH{value: amountBNBLiquidity}(address(this),amountToLiquify,0,0,autoLiquidityReceiver,block.timestamp) (bellaciao.sol#624-631)
        State variables written after the call(s):
        - _balances[sender] = _balances[sender].sub(amount,Insufficient Balance) (bellaciao.sol#520)
        - _balances[recipient] = _balances[recipient].add(amountReceived) (bellaciao.sol#523)
        - amountReceived = takeFee(sender,amount) (bellaciao.sol#522)
                - _balances[address(this)] = _balances[address(this)].add(feeAmount) (bellaciao.sol#558)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities

DividendDistributor.distributeDividend(address) (bellaciao.sol#325-336) ignores return value by BETH.transfer(shareholder,amount) (bellaciao.sol#331)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
```

```
Reentrancy in DividendDistributor.distributeDividend(address) (bellaciao.sol#325-336):
        External calls:
        - BETH.transfer(shareholder,amount) (bellaciao.sol#331)
        State variables written after the call(s):
        - shares[shareholder].totalRealised = shares[shareholder].totalRealised.add(amount) (bellaciao.sol#333)
        - shares[shareholder].totalExcluded = getCumulativeDividends(shares[shareholder].amount) (bellaciao.sol#334)
Reentrancy in DividendDistributor.process(uint256) (bellaciao.sol#294-318):
        External calls:
        - distributeDividend(shareholders[currentIndex]) (bellaciao.sol#310)
                - BETH.transfer(shareholder,amount) (bellaciao.sol#331)
        State variables written after the call(s):
        - currentIndex ++ (bellaciao.sol#315)
Reentrancy in DividendDistributor.setShare(address,uint256) (bellaciao.sol#258-272):
        External calls:
        - distributeDividend(shareholder) (bellaciao.sol#260)
                - BETH.transfer(shareholder,amount) (bellaciao.sol#331)
        State variables written after the call(s):
        - shares[shareholder].amount = amount (bellaciao.sol#270)
        - shares[shareholder].totalExcluded = getCumulativeDividends(shares[shareholder].amount) (bellaciao.sol#271)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

BellaCiao.swapBack() (bellaciao.sol#590-634) ignores return value by router.addLiquidityETH{value: amountBNBLiquidity}(address(this),amountToLiquify,0,0,autoLiquidityReceive
r,block.timestamp) (bellaciao.sol#624-631)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

BellaCiao.swapBack().tmpSuccess (bellaciao.sol#618) is written in both
        (tmpSuccess) = address(marketingFeeReceiver).call{gas: 30000,value: amountBNBMarketing}() (bellaciao.sol#618)
        tmpSuccess = false (bellaciao.sol#621)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#write-after-write
```

```
DividendDistributor.setDistributionCriteria(uint256,uint256) (bellaciao.sol#253-256) should emit an event for:
        - minPeriod = _minPeriod (bellaciao.sol#254)
        - minDistribution = _minDistribution (bellaciao.sol#255)
BellaCiao.setTxLimit(uint256) (bellaciao.sol#637-639) should emit an event for:
        - _maxTxAmount = amount (bellaciao.sol#638)
BellaCiao.setFees(uint256,uint256,uint256,uint256) (bellaciao.sol#663-670) should emit an event for:
        - liquidityFee = _liquidityFee (bellaciao.sol#664)
        - reflectionFee = _reflectionFee (bellaciao.sol#665)
        - marketingFee = _marketingFee (bellaciao.sol#666)
        - totalFee = _liquidityFee.add(_reflectionFee).add(_marketingFee) (bellaciao.sol#667)
        - feeDenominator = _feeDenominator (bellaciao.sol#668)
BellaCiao.setSwapBackSettings(bool,uint256) (bellaciao.sol#677-680) should emit an event for:
        - swapThreshold = _amount (bellaciao.sol#679)
BellaCiao.setTargetLiquidity(uint256,uint256) (bellaciao.sol#682-685) should emit an event for:
        - targetLiquidity = _target (bellaciao.sol#683)
        - targetLiquidityDenominator = _denominator (bellaciao.sol#684)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

Auth.transferOwnership(address).adr (bellaciao.sol#134) lacks a zero-check on :
                - owner = adr (bellaciao.sol#135)
BellaCiao.setFeeReceivers(address,address)._autoLiquidityReceiver (bellaciao.sol#672) lacks a zero-check on :
                - autoLiquidityReceiver = _autoLiquidityReceiver (bellaciao.sol#673)
BellaCiao.setFeeReceivers(address,address)._marketingFeeReceiver (bellaciao.sol#672) lacks a zero-check on :
                - marketingFeeReceiver = _marketingFeeReceiver (bellaciao.sol#674)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

BellaCiao.airdrop(address,address[],uint256[]) (bellaciao.sol#713-736) has external calls inside a loop: distributor.setShare(addresses[i_scope_0],_balances[addresses[i_sco
pe_0]]) (bellaciao.sol#728)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
```

```
        - isTxLimitExempt[msg.sender] = true (bellaciao.sol#454)
        - marketingFeeReceiver = msg.sender (bellaciao.sol#447)
Reentrancy in DividendDistributor.deposit() (bellaciao.sol#274-292):
        External calls:
        - router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: msg.value}(0,path,address(this),block.timestamp) (bellaciao.sol#281-286)
        State variables written after the call(s):
        - dividendsPerShare = dividendsPerShare.add(dividendsPerShareAccuracyFactor.mul(amount).div(totalShares)) (bellaciao.sol#291)
        - totalDividends = totalDividends.add(amount) (bellaciao.sol#290)
Reentrancy in DividendDistributor.distributeDividend(address) (bellaciao.sol#325-336):
        External calls:
        - BETH.transfer(shareholder,amount) (bellaciao.sol#331)
        State variables written after the call(s):
        - shareholderClaims[shareholder] = block.timestamp (bellaciao.sol#332)
Reentrancy in DividendDistributor.setShare(address,uint256) (bellaciao.sol#258-272):
        External calls:
        - distributeDividend(shareholder) (bellaciao.sol#260)
                - BETH.transfer(shareholder,amount) (bellaciao.sol#331)
        State variables written after the call(s):
        - addShareholder(shareholder) (bellaciao.sol#264)
                - shareholderIndexes[shareholder] = shareholders.length (bellaciao.sol#358)
        - removeShareholder(shareholder) (bellaciao.sol#266)
                - shareholderIndexes[shareholders[shareholders.length - 1]] = shareholderIndexes[shareholder] (bellaciao.sol#364)
        - addShareholder(shareholder) (bellaciao.sol#264)
                - shareholders.push(shareholder) (bellaciao.sol#359)
        - removeShareholder(shareholder) (bellaciao.sol#266)
                - shareholders[shareholderIndexes[shareholder]] = shareholders[shareholders.length - 1] (bellaciao.sol#363)
                - shareholders.pop() (bellaciao.sol#365)
        - totalShares = totalShares.sub(shares[shareholder].amount).add(amount) (bellaciao.sol#269)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

```
                - amountReceived = takeFee(sender,amount) (bellaciao.sol#522)
Reentrancy in BellaCiao.constructor() (bellaciao.sol#426-451):
        External calls:
        - pair = IDEXFactory(router.factory()).createPair(WBNB,address(this)) (bellaciao.sol#428)
        Event emitted after the call(s):
        - Transfer(address(0),msg.sender,_totalSupply) (bellaciao.sol#450)
Reentrancy in BellaCiao.swapBack() (bellaciao.sol#590-634):
        External calls:
        - router.swapExactTokensForETHSupportingFeeOnTransferTokens(amountToSwap,0,path,address(this),block.timestamp) (bellaciao.sol#601-607)
        - distributor.deposit{value: amountBNBReflection}() (bellaciao.sol#617)
        - (tmpSuccess) = address(marketingFeeReceiver).call{gas: 30000,value: amountBNBMarketing}() (bellaciao.sol#618)
        - router.addLiquidityETH{value: amountBNBLiquidity}(address(this),amountToLiquify,0,0,autoLiquidityReceiver,block.timestamp) (bellaciao.sol#624-631)
        External calls sending eth:
        - distributor.deposit{value: amountBNBReflection}() (bellaciao.sol#617)
        - (tmpSuccess) = address(marketingFeeReceiver).call{gas: 30000,value: amountBNBMarketing}() (bellaciao.sol#618)
        - router.addLiquidityETH{value: amountBNBLiquidity}(address(this),amountToLiquify,0,0,autoLiquidityReceiver,block.timestamp) (bellaciao.sol#624-631)
        Event emitted after the call(s):
        - AutoLiquify(amountBNBLiquidity,amountToLiquify) (bellaciao.sol#632)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

DividendDistributor.shouldDistribute(address) (bellaciao.sol#320-323) uses timestamp for comparisons
        Dangerous comparisons:
        - shareholderClaims[shareholder] + minPeriod < block.timestamp && getUnpaidEarnings(shareholder) > minDistribution (bellaciao.sol#321-322)
BellaCiao._transferFrom(address,address,uint256) (bellaciao.sol#490-538) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(cooldownTimer[recipient] < block.timestamp,Please wait for 1min between two buys) (bellaciao.sol#508)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

DividendDistributor.process(uint256) (bellaciao.sol#294-318) has costly operations inside a loop:
        - currentIndex = 0 (bellaciao.sol#306)
DividendDistributor.process(uint256) (bellaciao.sol#294-318) has costly operations inside a loop:
        - currentIndex ++ (bellaciao.sol#315)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop
```

```
BellaCiao._maxTxAmount (bellaciao.sol#382) is set pre-construction with a non-constant function or state variable:
        - _totalSupply * 1 / 100
BellaCiao._maxWalletToken (bellaciao.sol#385) is set pre-construction with a non-constant function or state variable:
        - (_totalSupply * 2) / 100
BellaCiao.swapThreshold (bellaciao.sol#422) is set pre-construction with a non-constant function or state variable:
        - _totalSupply * 10 / 10000
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state

Pragma version^0.7.4 (bellaciao.sol#17) allows old versions
solc-0.7.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in BellaCiao.swapBack() (bellaciao.sol#590-634):
        - (tmpSuccess) = address(marketingFeeReceiver).call{gas: 30000,value: amountBNBMarketing}() (bellaciao.sol#618)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
Parameter DividendDistributor.setDistributionCriteria(uint256,uint256)._minPeriod (bellaciao.sol#253) is not in mixedCase
Parameter DividendDistributor.setDistributionCriteria(uint256,uint256)._minDistribution (bellaciao.sol#253) is not in mixedCase
Variable DividendDistributor._token (bellaciao.sol#205) is not in mixedCase
Variable DividendDistributor.BETH (bellaciao.sol#213) is not in mixedCase
Variable DividendDistributor.WBNB (bellaciao.sol#214) is not in mixedCase
Parameter BellaCiao.tradingStatus(bool)._status (bellaciao.sol#578) is not in mixedCase
Parameter BellaCiao.cooldownEnabled(bool,uint8)._status (bellaciao.sol#583) is not in mixedCase
Parameter BellaCiao.cooldownEnabled(bool,uint8)._interval (bellaciao.sol#583) is not in mixedCase
Parameter BellaCiao.setFees(uint256,uint256,uint256,uint256)._liquidityFee (bellaciao.sol#663) is not in mixedCase
Parameter BellaCiao.setFees(uint256,uint256,uint256,uint256)._reflectionFee (bellaciao.sol#663) is not in mixedCase
Parameter BellaCiao.setFees(uint256,uint256,uint256,uint256)._marketingFee (bellaciao.sol#663) is not in mixedCase
Parameter BellaCiao.setFees(uint256,uint256,uint256,uint256)._feeDenominator (bellaciao.sol#663) is not in mixedCase
Parameter BellaCiao.setFeeReceivers(address,address)._autoLiquidityReceiver (bellaciao.sol#672) is not in mixedCase
Parameter BellaCiao.setFeeReceivers(address,address)._marketingFeeReceiver (bellaciao.sol#672) is not in mixedCase
Parameter BellaCiao.setSwapBackSettings(bool,uint256)._enabled (bellaciao.sol#677) is not in mixedCase
Parameter BellaCiao.setSwapBackSettings(bool,uint256)._amount (bellaciao.sol#677) is not in mixedCase
Parameter BellaCiao.setTargetLiquidity(uint256,uint256)._target (bellaciao.sol#682) is not in mixedCase
Parameter BellaCiao.setTargetLiquidity(uint256,uint256)._denominator (bellaciao.sol#682) is not in mixedCase
Parameter BellaCiao.setDistributionCriteria(uint256,uint256)._minPeriod (bellaciao.sol#687) is not in mixedCase
Parameter BellaCiao.setDistributionCriteria(uint256,uint256)._minDistribution (bellaciao.sol#687) is not in mixedCase
Variable BellaCiao.BETH (bellaciao.sol#372) is not in mixedCase
Variable BellaCiao.WBNB (bellaciao.sol#373) is not in mixedCase
Variable BellaCiao.DEAD (bellaciao.sol#374) is not in mixedCase
Variable BellaCiao.ZERO (bellaciao.sol#375) is not in mixedCase
Constant BellaCiao._name (bellaciao.sol#377) is not in UPPER_CASE_WITH_UNDERSCORES
Constant BellaCiao._symbol (bellaciao.sol#378) is not in UPPER_CASE_WITH_UNDERSCORES
Constant BellaCiao._decimals (bellaciao.sol#379) is not in UPPER_CASE_WITH_UNDERSCORES
Variable BellaCiao._totalSupply (bellaciao.sol#381) is not in mixedCase
Variable BellaCiao._maxTxAmount (bellaciao.sol#382) is not in mixedCase
Variable BellaCiao._maxWalletToken (bellaciao.sol#385) is not in mixedCase
Variable BellaCiao._balances (bellaciao.sol#387) is not in mixedCase
Variable BellaCiao._allowances (bellaciao.sol#388) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
Variable IDEXRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (bellaciao.sol#154) is too similar to IDEXRouter.addLiquidi
ty(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (bellaciao.sol#155)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

BellaCiao.slitherConstructorVariables() (bellaciao.sol#369-740) uses literals with too many digits:
        - DEAD = 0x000000000000000000000000000000000000dEaD (bellaciao.sol#374)
BellaCiao.slitherConstructorVariables() (bellaciao.sol#369-740) uses literals with too many digits:
        - ZERO = 0x0000000000000000000000000000000000000000 (bellaciao.sol#375)
BellaCiao.slitherConstructorVariables() (bellaciao.sol#369-740) uses literals with too many digits:
        - distributorGas = 500000 (bellaciao.sol#414)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

BellaCiao.BETH (bellaciao.sol#372) is never used in BellaCiao (bellaciao.sol#369-740)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

BellaCiao.BETH (bellaciao.sol#372) should be constant
BellaCiao.DEAD (bellaciao.sol#374) should be constant
BellaCiao.WBNB (bellaciao.sol#373) should be constant
BellaCiao.ZERO (bellaciao.sol#375) should be constant
BellaCiao._totalSupply (bellaciao.sol#381) should be constant
BellaCiao.launchedAt (bellaciao.sol#410) should be constant
DividendDistributor.WBNB (bellaciao.sol#214) should be constant
DividendDistributor.dividendsPerShareAccuracyFactor (bellaciao.sol#227) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

authorize(address) should be declared external:
        - Auth.authorize(address) (bellaciao.sol#106-108)
unauthorize(address) should be declared external:
        - Auth.unauthorize(address) (bellaciao.sol#113-115)
transferOwnership(address) should be declared external:
        - Auth.transferOwnership(address) (bellaciao.sol#134-138)
tradingStatus(bool) should be declared external:
        - BellaCiao.tradingStatus(bool) (bellaciao.sol#578-580)
cooldownEnabled(bool,uint8) should be declared external:
        - BellaCiao.cooldownEnabled(bool,uint8) (bellaciao.sol#583-586)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
./bellaciao.sol analyzed (8 contracts with 75 detectors), 86 result(s) found
token@token-VirtualBox:~/Desktop/audit$
```

Sol-hint Tool:

A linter for Solidity that provides both Security and Style Guide validations.

Coding style issues influence code readability and, in some cases, may lead to bugs in future. Smart Contracts have a naming convention, indentation and code layout issues. It's recommended to use Solidity Style Guide to fix all the issues. Consider following the Solidity guidelines on formatting the code and commenting for all the files. It can improve the overall code quality and readability

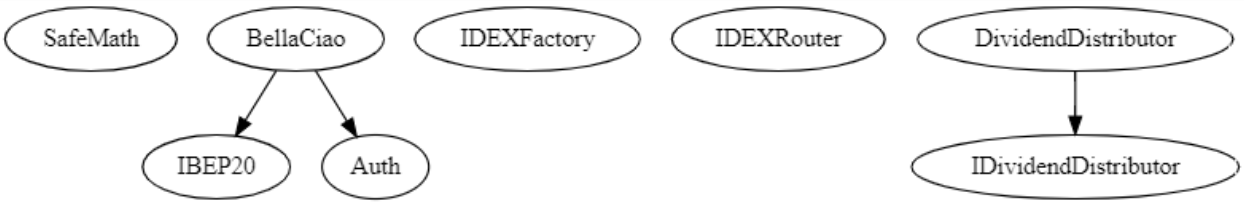
```
bellaciaotoken.sol
    7:2  error  Line length must be no more than 120 but current length is 133  max-line-length
  461:2  error  Line length must be no more than 120 but current length is 137  max-line-length
  498:2  error  Line length must be no more than 120 but current length is 194  max-line-length
  500:2  error  Line length must be no more than 120 but current length is 128  max-line-length
  663:2  error  Line length must be no more than 120 but current length is 137  max-line-length

✖ 5 problems (5 errors, 0 warnings)
```

Functional View

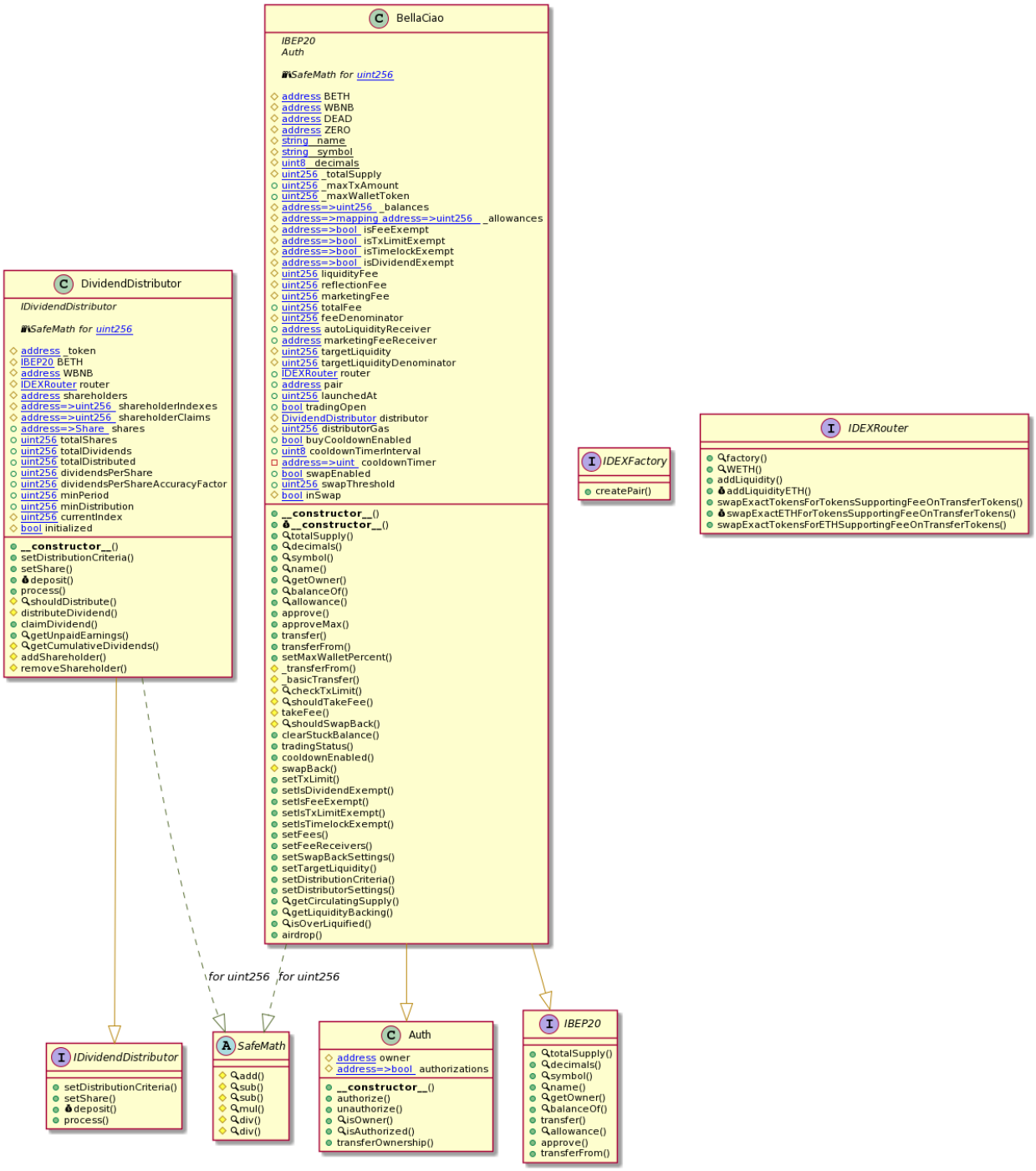
Function	Return Type	Test Result
name	Public	Verified
symbol	Public	Verified
decimals	Public	Verified
totalSupply	Public	Verified
balanceOf	Public	Verified
transfer	Public	Verified
allowance	Public	Verified
approve	Public	Verified
TransferFrom	Public	Verified
increaseAllowance	Public	Verified
decreaseAllowance	Public	Verified
isExcludedFromReward	Public	Verified
totalFees	Public	Verified
deliver	Public	Verified
reflectionFromToken	Public	Verified

tokenFromReflection	Public	Verified
excludeFromReward	Public	Verified
includeInReward	Public	Verified
transferBothExcluded	Public	Verified

Inheritance Chart



Uml Chart


BellaCiao
IBEP20
Auth
SafeMath for uint256
address BETH
address WBNB
address DEAD
address ZERO
string _name
string _symbol
uint8 _decimals
uint256 _totalSupply
uint256 _maxTxAmount
uint256 _maxWalletToken
address=>uint256 _balances
address=>mapping address=>uint256 _allowances
address=>bool isFeeExempt
address=>bool isTxLimitExempt
address=>bool isTimelockExempt
address=>bool isDividendExempt
uint256 liquidityFee
uint256 reflectionFee
uint256 marketingFee
uint256 totalFee
uint256 feeDenominator
address autoLiquidityReceiver
address marketingFeeReceiver
uint256 targetLiquidity
uint256 targetLiquidityDenominator
IDEXRouter router
address pair
uint256 launchedAt
bool tradingOpen
DividendDistributor distributor
uint256 distributorGas
bool buyCooldownEnabled
uint8 cooldownTimerInterval
address=>uint _cooldownTimer
bool swapEnabled
uint256 swapThreshold
bool inSwap
__constructor__()
__constructor__()
totalSupply()
decimals()
symbol()
name()
getOwner()
balanceOf()
allowance()
approve()
approveMax()
transfer()
transferFrom()
setMaxWalletPercent()
_transferFrom()
_basicTransfer()
checkTxLimit()
shouldTakeFee()
takeFee()
shouldSwapBack()
clearStuckBalance()
tradingStatus()
cooldownEnabled()
swapBack()
setTxLimit()
setIsDividendExempt()
setIsFeeExempt()
setIsTxLimitExempt()
setIsTimelockExempt()
setFees()
setFeeReceivers()
setSwapBackSettings()
setTargetLiquidity()
setDistributionCriteria()
setDistributorSettings()
getCirculatingSupply()
getLiquidityBacking()
isOverLiquified()
airdrop()
DividendDistributor
IDividendDistributor
SafeMath for uint256
address _token
IBEP20 BETH
address WBNB
IDEXRouter router
address shareholders
address=>uint256 _shareholderIndexes
address=>uint256 _shareholderClaims
address=>Share _shares
uint256 totalShares
uint256 totalDividends
uint256 totalDistributed
uint256 dividendsPerShare
uint256 dividendsPerShareAccuracyFactor
uint256 minPeriod
uint256 minDistribution
uint256 currentIndex
bool initialized
__constructor__()
setDistributionCriteria()
setShare()
deposit()
process()
shouldDistribute()
distributeDividend()
claimDividend()
getUnpaidEarnings()
getCumulativeDividends()
addShareholder()
removeShareholder()
IDEXFactory
createPair()
IDEXRouter
factory()
WETH()
addLiquidity()
addLiquidityETH()
swapExactTokensForTokensSupportingFeeOnTransferTokens()
swapExactETHForTokensSupportingFeeOnTransferTokens()
swapExactTokensForETHSupportingFeeOnTransferTokens()
for uint256
for uint256
IDividendDistributor
setDistributionCriteria()
setShare()
deposit()
process()
SafeMath
add()
sub()
sub()
mul()
div()
div()
Auth
address owner
address=>bool authorizations
__constructor__()
authorize()
unauthorize()
isOwner()
isAuthorized()
transferOwnership()
IBEP20
totalSupply()
decimals()
symbol()
name()
getOwner()
balanceOf()
transfer()
allowance()
approve()
transferFrom()

Audit Findings Results

There were 1 Low issue and 3 Informational found during the audit. All the mentioned findings may have an effect only in case of specific conditions performed by the contract owner. None of the critical issues were resolved.

Generally, the contracts are well written and structured. The findings during the audit have some impact on contract performance or security

Disclaimer

This audit does not provide a security or correctness guarantee of audited smart contracts You agree that your access and/or use, including but not limited to any services, products, platforms, content, will be at your Own risk. Smart contract remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security.


http://tokenaudit.net/


https://t.me/TokenAuditt


https://twitter.com/AuditToken