



# **RugFreeCoins Audit**



**Tres Leches Cake Audit**  
**Smart Contract Security Audit**  
**October 09, 2021**



# Contents

Audit details	1
Disclaimer	2
Background	3
About the project	4
Target market and the concept	6
Potential to grow with score points	10
Total Points	10
Contract details	11
Token distribution	12
Contract code function details	13
Contract description table	14
Security issue checking status	22
Owner privileges	24
Audit conclusion	26

# Audit details



## **Audited project**

Tres Leches Cake Token



## **Contract Address**

0xf6aba0d60a55e77a8924e75382a45802acafc09f



## **Client contact**

Tres Leches Cake Team



## **Blockchain**

Binance smart chain



## **Project website**

<https://tresleches.finance/>

# Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

**DISCLAIMER:** By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Rugfreecoins and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (Rugfreecoins) owe no duty of care towards you or any other person, nor does Rugfreecoins make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and Rugfreecoins hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, Rugfreecoins hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Rugfreecoins, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report. 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.

# Background

Rugfreecoins was commissioned by Tres Leches Cake to perform an audit of the smart contract.

**<https://bscscan.com/token/0xf6aba0d60a55e77a8924e75382a45802acafc09f>**

The focus of this audit is to verify that the smart contract is secure, resilient and working according to the specifications.

The information in this report should be used to understand the risk exposure of the smart contract, project feasibility, long term sustainability and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

# About the project

Tres Leches Cake is a token built on the Binance Smart Chain. Each transaction, purchase and sale incur a 12% fee. The main goal of this token is to create a community effort to assist the feature students in having a fantastic education via scholarship donations. The token is driven by the community, and the options are endless.

## Features

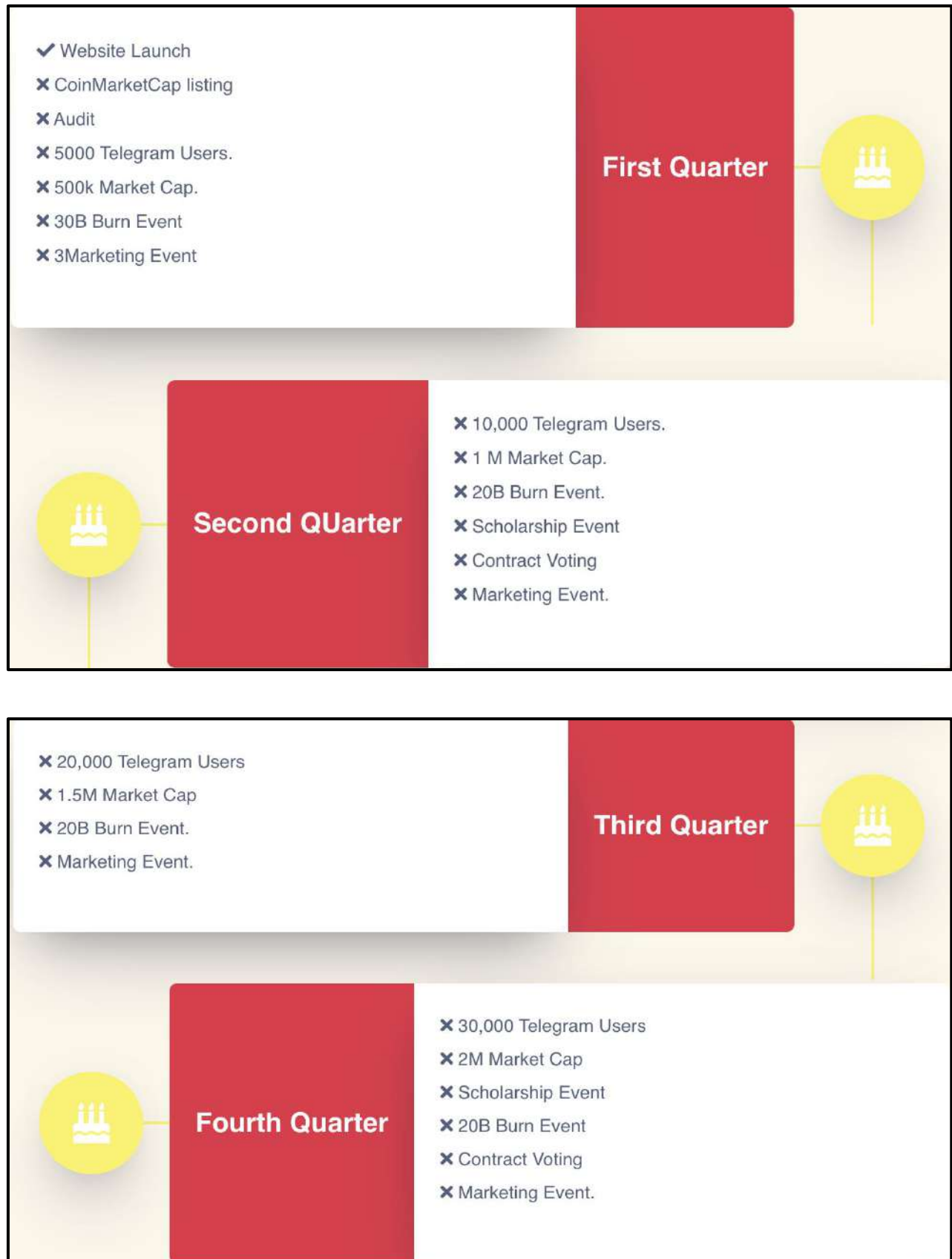
- ❖ The automatic Tres Leches Cake token rewards will be distributed among every holder proportional to how many tokens each individual hold in values of 5% when buying and selling.
- ❖ The additional component included under the sustainability section is a liquidity fee of 5%, which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity. This is a key element for decentralized exchanges like Pancakeswap.
- ❖ The scholarship fee of 2% when buying and selling for charity is what allows Tres Leches Cake to allocate funds for the good cause. This will empower the Tres Leches Cake community in the long run and motivate more people to join in!

## Tokenomics

### 12% fee when buying and selling

- ❖ 5% of trade goes to holders' pockets in Tres Leches Cake tokens.
- ❖ 5% of every trade goes to the liquidity pool.
- ❖ 2% of trade goes to the scholarship wallet in tokens.

# Roadmap



# Target market and the concept

## Target market

- ❖ Anyone who's interested in Crypto space with long term investment plans.
- ❖ Anyone who's ready to earn a passive income in Tres Leches Cake tokens by holding tokens.
- ❖ Anyone who's interested in trading tokens.
- ❖ Anyone who's interested in taking part with decision making of the project.
- ❖ Anyone who's interested in collecting NFTs or trading NFTs.
- ❖ Anyone who's interested in taking part with the future plans of the Tres Leches Cake token.
- ❖ Anyone who's interested in supporting a good cause. (Scholarship feature)
- ❖ Anyone who's interested in making financial transactions with any other party using Tres Leches Cake token as the currency.

## Core concept

### The token reward system

5% of each transaction when selling gets sent amongst all holders in tokens. The holders will be eligible to receive tokens, whenever a transaction occurs, and rewards are proportional to how many tokens each individual holds.

### Sustainable mechanism

**The liquidity fee of 5% when buying and selling**, which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity.

### Good cause

**2% Scholarship fee** per transaction will be sent in tokens to a separate wallet.



## SCHOLARSHIP REQUIREMENTS

1. GPA 3.0 or Higher, if this is different in your country, feel free to explain why and what is your current academic achievement.
2. Proof that you deserve to be the Winner of the scholarship; show the community why and how you plan to change the future by having a college education.
3. Current Sophomore or Junior.
4. A 650-word essay, the Topic will be chosen by the community and announced when the application process is ready to start. Check Roadmap Section.
5. Must be a Tres Leches Token Holder and always hold at least 1,000 tokens.
6. Must show proof of enrollment into a higher education institution.
7. Candidates must be willing to join a telegram channel to answer three questions from the community leads during the nomination process.

We will open the nomination process on the website by submitting a form that will be made available. The community will choose Top 3 candidates to go into a final Interview process. During this interview, the community leads will ask the candidates three main questions. The community then will vote to choose their community winner and award the scholarship.

# WINNER ANNOUNCEMENT



The Winner will be announced on our social media channels. After the Winner has been chosen, the Winner will have 30 days (about four and a half weeks) to present the evidence required for the award.

The award will be paid in BNB to the Winner, and the transaction will be made available. Why not an official check? This is the cryptocurrency world. We want to encourage all the nominees to use these methods to pay for their college education.

What is the maximum a winner can expect to win from the scholarship?

These will all depend on the cryptocurrency market and the token market cap. The money will be taken out of the scholarship wallet and converted into BNB. We should see at least 1,000 in the scholarship wallet or an equivalent to 10 Billion tokens, and if the value of 10 Billion tokens is more, then we will award that total value to the Winner.

What happens if the market cap is less than the desired amount? Then we as a community should contribute to make the first Winner of the scholarship a success. If everything fails, then the community leads will meet with the Chef and decide how to proceed. But rest assured that we will do our best to meet our goals.

### COMMUNITIES

Building a community is essential to any project. We want to create the best environment for all.

We will look for Community Leaders that can help us build the following vital communities.

- English
- Spanish
- Chinese
- Portuguese

If you are interested in being a community leader, please reach out to the Chef to get your kitchen set up.



### WALLET

Every coin needs a wallet. Building an Android wallet is something we have in mind. However, we feel that Trust Wallet gives us the best options and stability to transfer coins between each holder. We will keep using Trust Wallet until the community vote and decide on next steps for the token.

### NFT

What Is a Non-Fungible Token (NFT)?

Non-fungible tokens or NFTs are cryptographic assets on blockchain with unique identification codes and metadata that distinguish them from each other. Unlike cryptocurrencies, they cannot be traded or exchanged at equivalency. This differs from fungible tokens like cryptocurrencies, which are identical to each other and, therefore, can be used as a medium for commercial transactions.

Non-Fungible Token Definition: Understanding NFTs.

<https://www.investopedia.com/non-fungible-tokens-nft-5115211>

The goal will be to create a version of the Tres Leches Cake from each country into an NFT, so if you are passionate about this, let us know, and we can start to launch International Tasty NFTs for all.

# Potential to grow with score points

1.	Project efficiency	9/10
2.	Project uniqueness	8/10
3	Information quality	10/10
4	Service quality	9/10
5	System quality	8/10
6	Impact on the community	10/10
7	Impact on the business	9/10
8	Preparing for the future	9/10
Total Points		<b>9/10</b>

# Contract details

## Token contract details for 09<sup>th</sup> October 2021

<b>Contract name</b>	Tres Leches Cake
<b>Contract address</b>	0xf6aba0d60a55e77a8924e75382a45802acafc09f
<b>Token supply</b>	1,000,000,000,000
<b>Token ticker</b>	3Leches
<b>Decimals</b>	9
<b>Token holders</b>	4
<b>Transaction count</b>	11
<b>Dev wallet address</b>	0x558b624de1d61379e0a131c7a9c6f6d9dcc14abe
<b>Contract deployer address</b>	0x4eeb7a903cC5E16F5Cf99e3Eb2dE94A51794FAE9
<b>Contract's current owner address</b>	0xf4c8d34bf1dc879eaf0267f038e838898e914d9f

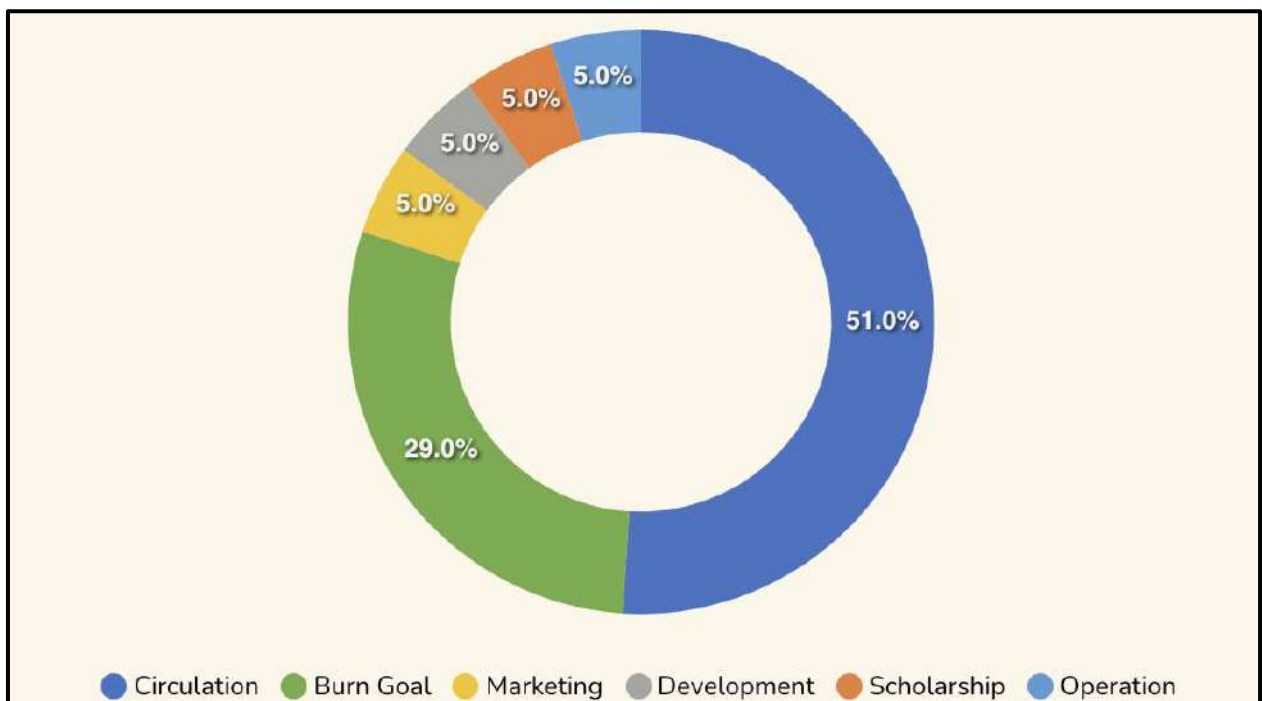


# Token distribution

## Tokens are distributed as follows:

Total of 1 trillion tokens, with 51% of the tokens being in circulation, and the other 49% will be distributed as follows.

- ❖ Tres Leches Cake Team will add 200B to the Operations Wallet, and this will be divided into the following areas.
  - 50B for Marketing.
  - 50B for Operations.
  - 50B for Development.
  - 50B for the Scholarship Wallet.
- ❖ 290B of the remaining will be burned monthly.
  - 30B burned the first month of the token.
  - 20B will be burned every until ran out of tokens to burn.





























# Contract code function details









No	Category	Item	Result
1	Coding conventions	BRC20 Token standards	pass
		compile errors	pass
		Compiler version security	pass
		visibility specifiers	pass
		Gas consumption	low issue
		SafeMath features	pass
		Fallback usage	pass
		tx.origin usage	pass
		deprecated items	pass
		Redundant code	pass
		Overriding variables	pass
2	Function call audit	Authorization of function call	pass
		Low level function (call/delegate call) security	pass
		Returned value security	pass
		Selfdestruct function security	pass
3	Business security	Access control of owners	low issue
		Business logics	pass
		Business implementations	pass
4	Integer overflow/underflow		pass
5	Reentrancy		pass
6	Exceptional reachable state		pass
7	Transaction ordering dependence		pass
8	Block properties dependence		pass
9	Pseudo random number generator (PRNG)		pass
10	DoS (Denial of Service)		pass
11	Token vesting implementation		pass
12	Fake deposit		pass
13	Event security		pass

# Contract description table












Below table represents the summary of the contracts and methods in the token contract. We scanned the whole contract and listed down all the Interfaces, functions and implementations with its visibility and mutability.












Contract	Type	Bases		
L	Function Name	Visibility	Mutability	Modifiers
<b>IERC20</b>	<b>Interface</b>			
L	totalSupply	External 		NO 
L	balanceOf	External 		NO 
L	transfer	External 		NO 
L	allowance	External 		NO 
L	approve	External 		NO 
L	transferFrom	External 		NO 
<b>SafeMath</b>	<b>Library</b>			
L	tryAdd	Internal 		
L	trySub	Internal 		
L	tryMul	Internal 		
L	tryDiv	Internal 		
L	tryMod	Internal 		
L	add	Internal 		
L	sub	Internal 		
L	mul	Internal 		
L	div	Internal 		
L	mod	Internal 		
L	sub	Internal 		










L	div	Internal 🔒		
L	mod	Internal 🔒		
<b>Context</b>	<b>Implementation</b>			
L	_msgSender	Internal 🔒		
L	_msgData	Internal 🔒		
<b>Address</b>	<b>Library</b>			
L	isContract	Internal 🔒		
L	sendValue	Internal 🔒	⬛	
L	functionCall	Internal 🔒	⬛	
L	functionCall	Internal 🔒	⬛	
L	functionCallWithValue	Internal 🔒	⬛	
L	functionCallWithValue	Internal 🔒	⬛	
L	functionStaticCall	Internal 🔒		
L	functionStaticCall	Internal 🔒		
L	functionDelegateCall	Internal 🔒	⬛	
L	functionDelegateCall	Internal 🔒	⬛	
L	_verifyCallResult	Private 🔒🔑		
<b>Ownable</b>	<b>Implementation</b>	<b>Context</b>		
L	owner	Public 🔓		NO🔓
L	renounceOwnership	Public 🔓	⬛	onlyOwner
L	transferOwnership	Public 🔓	⬛	onlyOwner





































L	lock	Public ¶		onlyOwner
L	unlock	Public ¶		NO¶
<b>IUniswapV2Factory</b>	<b>Interface</b>			
L	feeTo	External ¶		NO¶
L	feeToSetter	External ¶		NO¶
L	getPair	External ¶		NO¶
L	allPairs	External ¶		NO¶
L	allPairsLength	External ¶		NO¶
L	createPair	External ¶		NO¶
L	setFeeTo	External ¶		NO¶
L	setFeeToSetter	External ¶		NO¶
<b>IUniswapV2Pair</b>	<b>Interface</b>			
L	name	External ¶		NO¶
L	symbol	External ¶		NO¶
L	decimals	External ¶		NO¶
L	totalSupply	External ¶		NO¶
L	balanceOf	External ¶		NO¶
L	allowance	External ¶		NO¶
L	approve	External ¶		NO¶
L	transfer	External ¶		NO¶
L	transferFrom	External ¶		NO¶
L	DOMAIN_SEPARATOR	External ¶		NO¶
L	PERMIT_TYPEHASH	External ¶		NO¶

































L	nonces	External ¶		NO¶
L	permit	External ¶		NO¶
L	MINIMUM_LIQUIDITY	External ¶		NO¶
L	factory	External ¶		NO¶
L	token0	External ¶		NO¶
L	token1	External ¶		NO¶
L	getReserves	External ¶		NO¶
L	price0CumulativeLast	External ¶		NO¶
L	price1CumulativeLast	External ¶		NO¶
L	kLast	External ¶		NO¶
L	mint	External ¶		NO¶
L	burn	External ¶		NO¶
L	swap	External ¶		NO¶
L	skim	External ¶		NO¶
L	sync	External ¶		NO¶
L	initialize	External ¶		NO¶
<b>IUniswapV2Router01</b>	<b>Interface</b>			
L	factory	External ¶		NO¶
L	WETH	External ¶		NO¶
L	addLiquidity	External ¶		NO¶
L	addLiquidityETH	External ¶		NO¶
L	removeLiquidity	External ¶		NO¶
L	removeLiquidityETH	External ¶		NO¶



L	removeLiquidityWithPermit	External ¶		NO ¶
L	removeLiquidityETHWithPermit	External ¶		NO ¶
L	swapExactTokensForTokens	External ¶		NO ¶
L	swapTokensForExactTokens	External ¶		NO ¶
L	swapExactETHForTokens	External ¶		NO ¶
L	swapTokensForExactETH	External ¶		NO ¶
L	swapExactTokensForETH	External ¶		NO ¶
L	swapETHForExactTokens	External ¶		NO ¶
L	quote	External ¶		NO ¶
L	getAmountOut	External ¶		NO ¶
L	getAmountIn	External ¶		NO ¶
L	getAmountsOut	External ¶		NO ¶
L	getAmountsIn	External ¶		NO ¶
<b>IUniswapV2Router02</b>	<b>Interface</b>	<b>IUniswapV2Router01</b>		
L	removeLiquidityETHSupportingFeeOnTransferTokens	External ¶		NO ¶
L	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ¶		NO ¶
L	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ¶		NO ¶

L	swapExactETHFor TokensSupportingF eeOnTransferToke ns	External ¶		NO¶
L	swapExactTokensF orETHSupportingF eeOnTransferToke ns	External ¶		NO¶
CoinToken	Implementation	Context, IERC20, Ownable		
L		Public ¶		NO¶
L	name	Public ¶		NO¶
L	symbol	Public ¶		NO¶
L	decimals	Public ¶		NO¶
L	totalSupply	Public ¶		NO¶
L	balanceOf	Public ¶		NO¶
L	transfer	Public ¶		NO¶
L	allowance	Public ¶		NO¶
L	approve	Public ¶		NO¶
L	transferFrom	Public ¶		NO¶
L	increaseAllowance	Public ¶		NO¶
L	decreaseAllowance	Public ¶		NO¶
L	isExcludedFromRe ward	Public ¶		NO¶
L	totalFees	Public ¶		NO¶
L	deliver	Public ¶		NO¶
L	reflectionFromToke n	Public ¶		NO¶
L	tokenFromReflectio n	Public ¶		NO¶

L	excludeFromReward	Public 		onlyOwner
L	includeInReward	External 		onlyOwner
L	_transferBothExcluded	Private 		
L	excludeFromFee	Public 		onlyOwner
L	includeInFee	Public 		onlyOwner
L	setTaxFeePercent	External 		onlyOwner
L	setDevFeePercent	External 		onlyOwner
L	setLiquidityFeePercent	External 		onlyOwner
L	setMaxTxPercent	Public 		onlyOwner
L	setDevWalletAddress	Public 		onlyOwner
L	setSwapAndLiquifyEnabled	Public 		onlyOwner
L		External 		NO 
L	_reflectFee	Private 		
L	_getValues	Private 		
L	_getTValues	Private 		
L	_getRValues	Private 		
L	_getRate	Private 		
L	_getCurrentSupply	Private 		
L	_takeLiquidity	Private 		
L	_takeDev	Private 		
L	calculateTaxFee	Private		

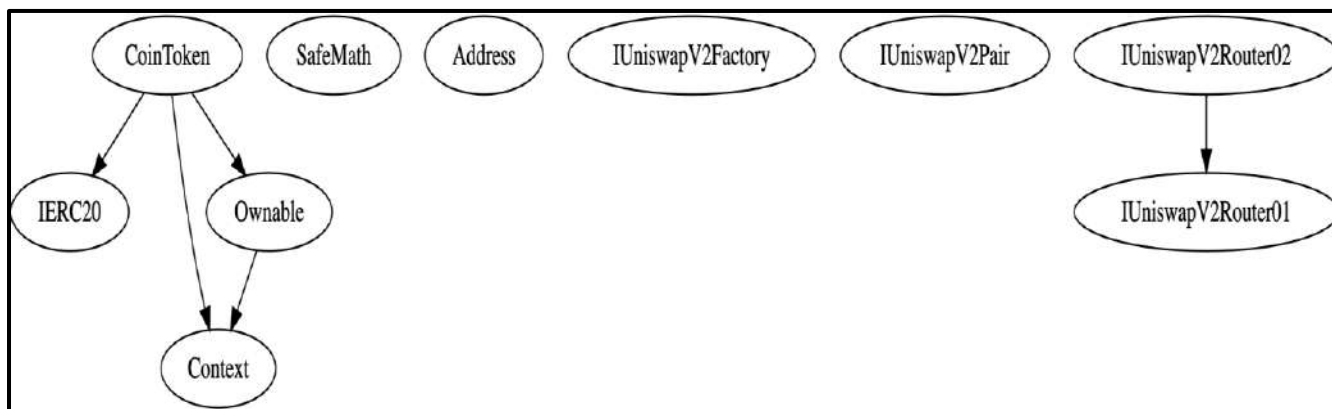
L	calculateDevFee	Private 		
L	calculateLiquidityFee	Private 		
L	removeAllFee	Private 		
L	restoreAllFee	Private 		
L	isExcludedFromFee	Public 		NO 
L	_approve	Private 		
L	_transfer	Private 		
L	swapAndLiquify	Private 		lockTheSwap
L	swapTokensForEth	Private 		
L	addLiquidity	Private 		
L	_tokenTransfer	Private 		
L	_transferStandard	Private 		
L	_transferToExcluded	Private 		
L	_transferFromExcluded	Private 		
L	setRouterAddress	External 		onlyOwner
L	setNumTokensSellToAddToLiquidity	External 		onlyOwner

### Legend

Symbol	Meaning
	Function can modify state
	Function is payable



## Inheritance Hierarchy



## Security issue checking status

### ❖ High severity issues

No medium severity issues found.

### ❖ Medium severity issues

No medium severity issues found.

### ❖ Low severity issues

- In the includeInReward, getCurrentSupply, and removeSniperList functions, if they use a long wallet list there can be an OUT\_OF\_GAS issue, better to use a small array list at once.

```
ftrace | funcSig
function includeInReward(address account↑) external onlyOwner {
    require(!_isExcluded[account↑], "Account is already included");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account↑) {
            _excluded[i] = _excluded[_excluded.length - 1];
            tOwned[account↑] = 0;
            _isExcluded[account↑] = false;
            _excluded.pop();
            break;
        }
    }
}
```

```
ftrace | funcSig
function _getCurrentSupply() private view returns (uint256, uint256) {
    uint256 rSupply = _rTotal;
    uint256 tSupply = _tTotal;
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (
            _rOwned[_excluded[i]] > rSupply ||
            _tOwned[_excluded[i]] > tSupply
        ) return (_rTotal, _tTotal);
        rSupply = rSupply.sub(_rOwned[_excluded[i]]);
        tSupply = tSupply.sub(_tOwned[_excluded[i]]);
    }
    if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
    return (rSupply, tSupply);
}
```

- The addLiquidity function calls the uniswapV2Router.addLiquidityETH function with the address specified as owner() for acquiring the generated LP tokens from the Tres Leches Cake Token-BNB pool. As a result, over time the \_owner address will accumulate a significant portion of LP tokens. If the \_owner is an EOA(Externally Owned Account), mishandling of its private key can have devastating consequences to the project as a whole.

```
ftrace | funcSig
function addLiquidity(uint256 tokenAmount↑, uint256 ethAmount↑) private {
    _approve(address(this), address(uniswapV2Router), tokenAmount↑);
    uniswapV2Router.addLiquidityETH{value: ethAmount↑}(
        address(this),
        tokenAmount↑,
        0, // slippage is unavoidable
        0, // slippage is unavoidable
        owner(),
        block.timestamp
    );
}
```

# Owner privileges

- ❖ The owner can exclude and include wallets from rewards.

```
ftrace | funcSig
function excludeFromReward(address account↑) public onlyOwner {
    require(!_isExcluded[account↑], "Account is already excluded");
    if (_rOwned[account↑] > 0) {
        _tOwned[account↑] = tokenFromReflection(_rOwned[account↑]);
    }
    _isExcluded[account↑] = true;
    _excluded.push(account↑);
}

ftrace | funcSig
function includeInReward(address account↑) external onlyOwner {
    require(_isExcluded[account↑], "Account is already included");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account↑) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _tOwned[account↑] = 0;
            _isExcluded[account↑] = false;
            _excluded.pop();
            break;
        }
    }
}
```

- ❖ The owner can exclude and include wallets from the fees.

```
ftrace | funcSig
function excludeFromFee(address account↑) public onlyOwner {
    _isExcludedFromFee[account↑] = true;
}

ftrace | funcSig
function includeInFee(address account↑) public onlyOwner {
    _isExcludedFromFee[account↑] = false;
}
```

- ❖ The owner can change the swap point.

```
ftrace | funcSig
function setNumTokensSellToAddToLiquidity(uint256 amountToUpdate↑)
    external
    onlyOwner
{
    numTokensSellToAddToLiquidity = amountToUpdate↑;
}
```

- ❖ The owner can change all fees.

```
ftrace | funcSig
function setTaxFeePercent(uint256 taxFee↑) external onlyOwner {
    _taxFee = taxFee↑;
}

ftrace | funcSig
function setDevFeePercent(uint256 devFee↑) external onlyOwner {
    _devFee = devFee↑;
}

ftrace | funcSig
function setLiquidityFeePercent(uint256 liquidityFee↑) external onlyOwner {
    _liquidityFee = liquidityFee↑;
}

ftrace | funcSig
```

- ❖ The owner can change the dev wallet address.

```
ftrace | funcSig
function setDevWalletAddress(address _addr↑) public onlyOwner {
    _devWalletAddress = _addr↑;
}

}
```

- ❖ The owner can enable and disable swap and liquify.

```
ftrace | funcSig
function setSwapAndLiquifyEnabled(bool _enabled↑) public onlyOwner {
    swapAndLiquifyEnabled = _enabled↑;
    emit SwapAndLiquifyEnabledUpdated(_enabled↑);
}

}
```

- ❖ The owner can change the pancake swap router address.

```
ftrace | funcSig
function setRouterAddress(address newRouter↑) external onlyOwner {
    IUniswapV2Router02 _uniswapV2Router = IUniswapV2Router02(newRouter↑);
    _uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
        .createPair(address(this), _uniswapV2Router.WETH());
    _uniswapV2Router = _uniswapV2Router;
}

}
```

# Audit conclusion

While conducting the audit of the Tres Leches Cake smart contract, it was observed that there is nothing alarming with the code and it only contains low severity issues.