RugFreeCoins Audit

CZ Terminator Token Smart Contract Security Audit

July 2nd ,2023

Overview

✅ No mint function found, the owner cannot mint tokens after initial deployment.

✅ The owner can’t set a max transaction limit

✅ The owner can’t pause trading.

✅ The owner can't blacklist wallets.

✅ The owner can’t set a max wallet limit

✅ The owner can’t claim the contract’s balance of its own token.

✅ The owner can’t set fees over 20%.

Contents

Overview ii

Audit details 1

Disclaimer 2

Background 3

Target market and the concept 5

Potential to grow with score points 6

Total Points 6

Contract details 7

Contract code function details 8

Contract description table 10

Security issue checking status 16

Owner privileges 18

Audit conclusion 23

Audit details

Audited project
CZ Terminator Token

Contract Address
0x0558666BA9c0aaEDA820Ab36F305A8d9EeB7F7c6

Client contact
CZ Terminator Token Team

Blockchain
Binance smart chain

Project website
http://czterminator.com

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 the CZ Terminator Team to perform an audit of the smart contract.

https://bscscan.com/token/0x0558666BA9c0aaEDA820Ab36F305A8d9EeB7F7c6

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, and long-term sustainability, and as a guide to improving the smart contract's security posture by remediating the identified issues.

Tokenomics

6% tax when buying & selling

6% of trade goes to the marketing wallet in BNB

Target market and the concept

Target market

- Anyone who’s interested in the Crypto space with long-term investment plans.
- Anyone who’s ready to earn a passive income by holding tokens.
- Anyone who’s interested in trading tokens.
- Anyone who’s ready to staking and receive rewards.
- Anyone who’s interested in taking part in the CZ Terminator ecosystem.
- Anyone who’s interested in taking part in the future plans of CZ Terminator Token.
- Anyone who’s interested in making financial transactions with any other party using CZ Terminator Token as the currency.

Potential to grow with score points

1.	Project efficiency	9/10
2.	Project uniqueness	9/10
3	Information quality	9/10
4	Service quality	9/10
5	System quality	8/10
6	Impact on the community	8/10
7	Impact on the business	9/10
8	Preparing for the future	8/10
9	Smart contract security	10/10
10	Smart contract functionality assessment	10/10
Total Points		**8.9/10**

Contract details

Token contract details for 2nd of July 2023

Contract name	CZ TERMINATOR
Contract address	0xa55781b30b7d37F1B5C7CC52eb86FB8d1F842122
Token supply	420,690,000,000,000
Token ticker	CZWINSEC
Decimals	9
Token holders	1
Transaction count	1
Contract deployer address	0x91B6cCEe7FA9d6DD68180Cc4a97C1F3a4460c90a
Contract's current owner address	0x91b6ccee7fa9d6dd68180cc4a97c1f3a4460c90a
Marketing wallet	0x91b6ccee7fa9d6dd68180cc4a97c1f3a4460c90a

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	pass
		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 & centralisation	Access control of owners	pass
		Business logics	pass
		Business implementation	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

The 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 their visibility and mutability.

Contract	Type	Bases		
L	Function Name	Visibility	Mutability	Modifiers
IBEP20	**Interface**			
L	totalSupply	External ❗		NO ❗
L	balanceOf	External ❗		NO ❗
L	transfer	External ❗	🛑	NO ❗
L	allowance	External ❗		NO ❗
L	approve	External ❗	🛑	NO ❗
L	transferFrom	External ❗	🛑	NO ❗
Context	**Implementation**			
L	_msgSender	Internal 🔒		
L	_msgData	Internal 🔒		
Ownable	Implementation	Context		

L		Public ❗	🛑	NO ❗
L	owner	Public ❗		NO ❗
L	renounceOwnership	Public ❗	🛑	onlyOwner
L	transferOwnership	Public ❗	🛑	onlyOwner
L	_setOwner	Private 🔐	🛑	
IFactory	**Interface**			
L	createPair	External ❗	🛑	NO ❗
InterfaceLP	**Interface**			
L	sync	External ❗	🛑	NO ❗
IRouter	**Interface**			
L	factory	External ❗		NO ❗
L	WETH	External ❗		NO ❗
L	addLiquidityETH	External ❗	💵	NO ❗
L	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
Address	**Library**			
L	sendValue	Internal 🔒	🛑	

Cz_ TERMINATOR	**Implementation**	**Context, IBEP20, 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	allowance	Public ❗		NO ❗
L	approve	Public ❗	🛑	NO ❗
L	transferFrom	Public ❗	🛑	NO ❗
L	increaseAllowance	Public ❗	🛑	NO ❗
L	decreaseAllowance	Public ❗	🛑	NO ❗
L	transfer	Public ❗	🛑	NO ❗
L	isExcludedFromReward	Public ❗		NO ❗
L	reflectionFromToken	Public ❗		NO ❗
L	updatedeadline	External ❗	🛑	onlyOwner
L	tokenFromReflection	Public ❗		NO ❗
L	excludeFromReward	Public ❗	🛑	onlyOwner
L	includeInReward	External ❗	🛑	onlyOwner

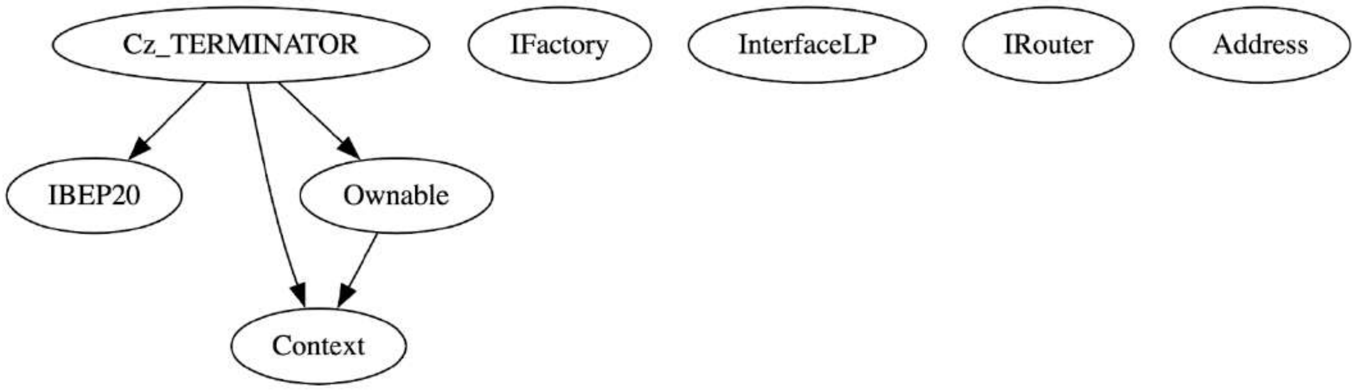
L	excludeFromFee	Public ❗	🛑	onlyOwner
L	isExcludedFromFee	Public ❗		NO ❗
L	_reflectRfi	Private 🔐	🛑	
L	_takeLiquidity	Private 🔐	🛑	
L	_takeMarketing	Private 🔐	🛑	
L	_takeBurn	Private 🔐	🛑	
L	_takeDev	Private 🔐	🛑	
L	_getValues	Private 🔐		
L	_getTValues	Private 🔐		
L	_getRValues1	Private 🔐		
L	_getRValues2	Private 🔐		
L	_getRate	Private 🔐		
L	_getCurrentSupply	Private 🔐		
L	_approve	Private 🔐	🛑	
L	_transfer	Private 🔐	🛑	
L	disableBurns	External ❗	🛑	onlyOwner

L	LpBurn	External ❗	🛑	onlyOwner
L	burn	External ❗	🛑	NO ❗
L	_burn	Internal 🔒	🛑	
L	_tokenTransfer	Private 🔐	🛑	
L	swapAndLiquify	Private 🔐	🛑	lockTheSwap
L	addLiquidity	Private 🔐	🛑	
L	swapTokensForBNB	Private 🔐	🛑	
L	setFees	External ❗	🛑	onlyOwner
L	updateMarketingWallet	External ❗	🛑	onlyOwner
L	updateDevWallet	External ❗	🛑	onlyOwner
L	updateSwapTokensAtAmount	External ❗	🛑	onlyOwner
L	updateSwapEnabled	External ❗	🛑	onlyOwner
L	rescueBNB	External ❗	🛑	onlyOwner
L	rescueAnyBEP20Tokens	Public ❗	🛑	onlyOwner
L		External ❗	💵	NO ❗

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Inheritance Hierarchy

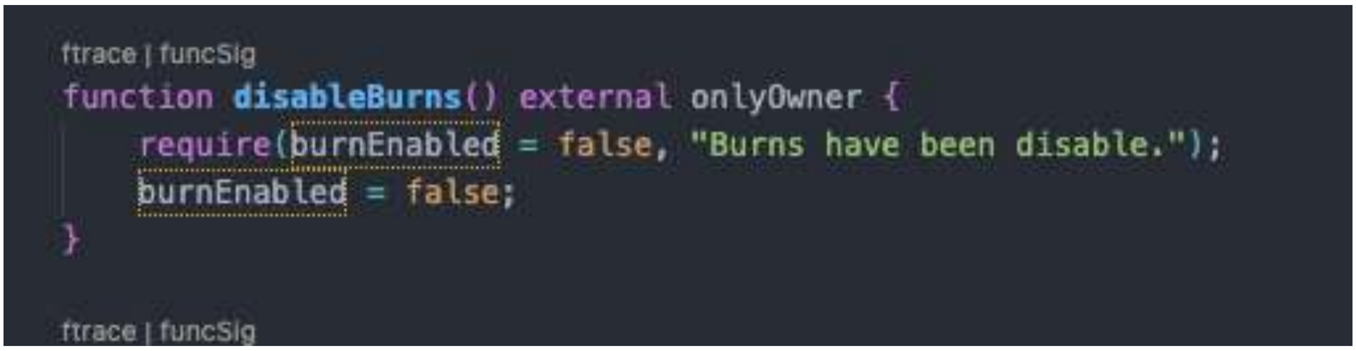


Security issue checking status

- **High severity issues**

It is not possible to disable burn permanently. In the "disableBurns" function, it checks if "burnEnabled" is false instead of true. Since "burnEnabled" is set to false by default, this function can never be called.

INFORMED & FIXED



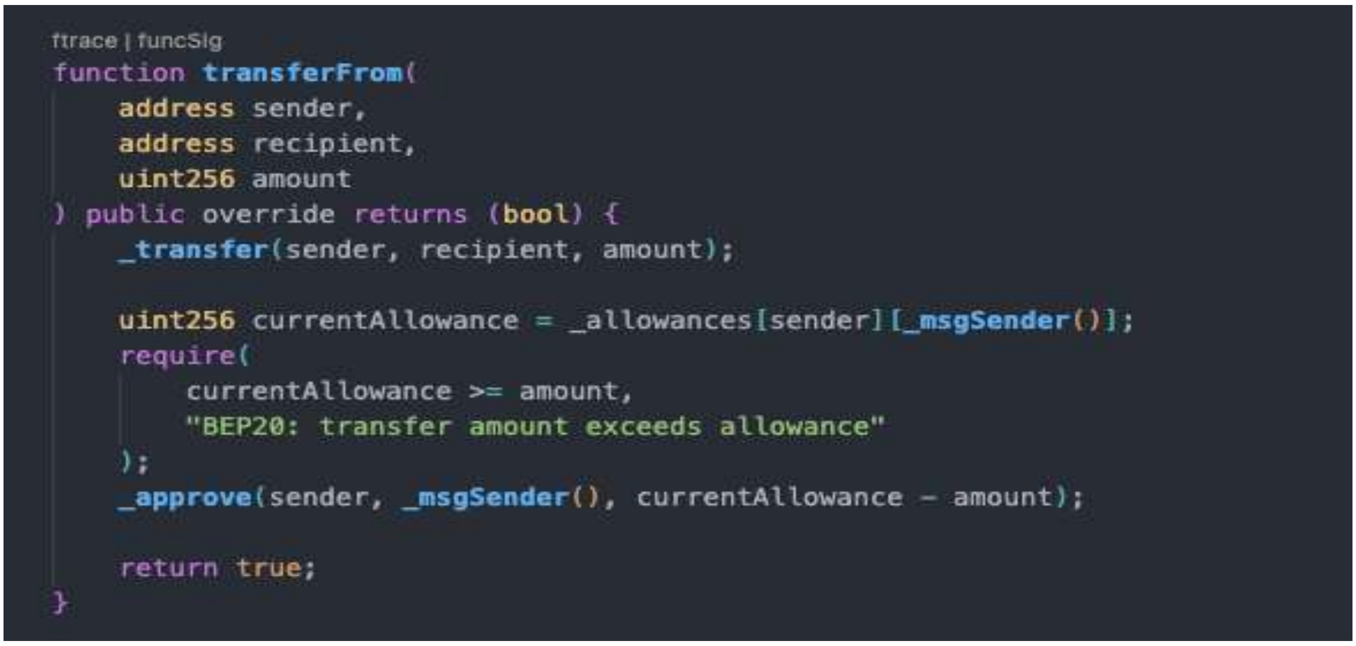
```
ftrace | funcSig
function disableBurns() external onlyOwner {
    require(burnEnabled = false, "Burns have been disable.");
    burnEnabled = false;
}

ftrace | funcSig
```

- **Medium severity issues**
  No medium severity issues found

- **Low severity issues**

In the "transferFrom" function, checking the allowance after transferring tokens can potentially cause a Reentrancy attack.

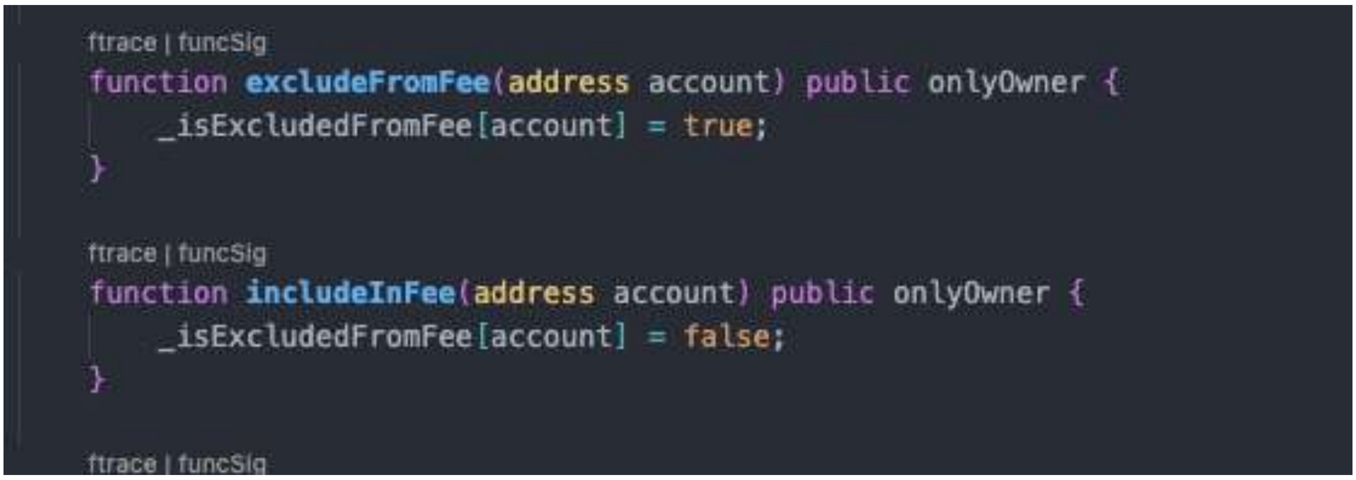
INFORMED & FIXED



```
ftrace | funcSig
function transferFrom(
    address sender,
    address recipient,
    uint256 amount
) public override returns (bool) {
    _transfer(sender, recipient, amount);

    uint256 currentAllowance = _allowances[sender][_msgSender()];
    require(
        currentAllowance >= amount,
        "BEP20: transfer amount exceeds allowance"
    );
    _approve(sender, _msgSender(), currentAllowance - amount);

    return true;
}
```

Combining the inclusion and exclusion of wallets in the fee function can save gas.

INFORMED & FIXED

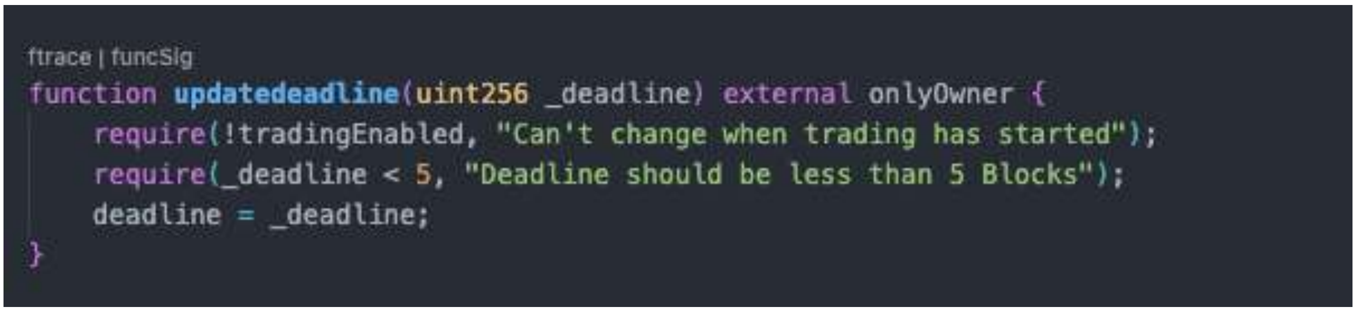


```
ftrace | funcSig
function excludeFromFee(address account) public onlyOwner {
    _isExcludedFromFee[account] = true;
}

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

ftrace | funcSig
```

- ❖ **Centralization Risk**
  No centralization issues found

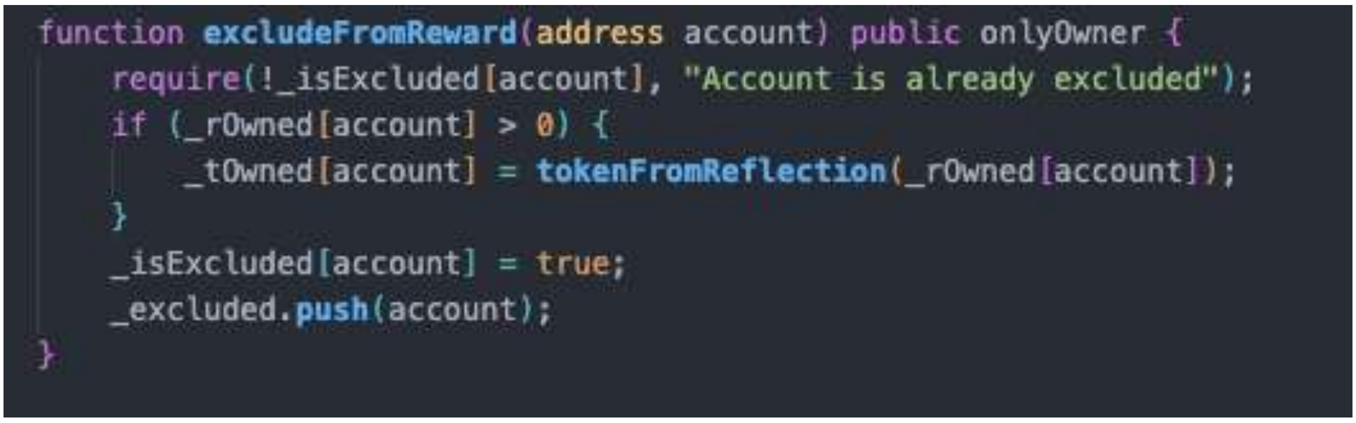
Owner privileges

❖ The owner can update the number of dead blocks before enabling trading a maximum of up to 4 blocks (if someone buys or sells within a dead block they will have more taxes)



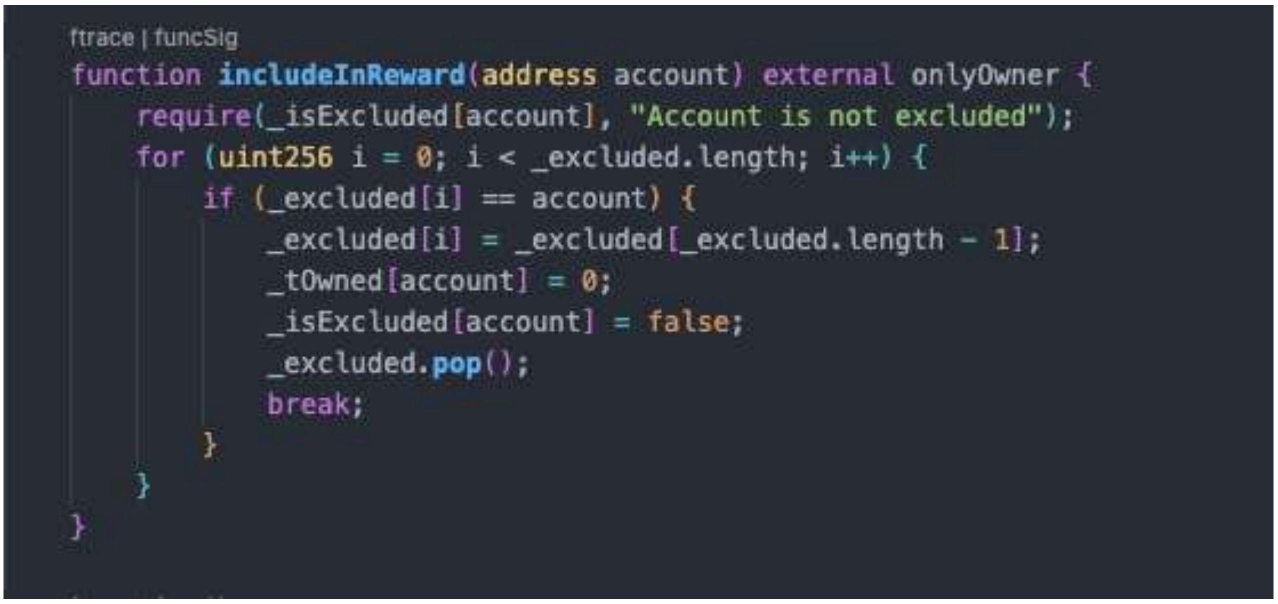
```
ftrace | funcSig
function updatedeadline(uint256 _deadline) external onlyOwner {
    require(!tradingEnabled, "Can't change when trading has started");
    require(_deadline < 5, "Deadline should be less than 5 Blocks");
    deadline = _deadline;
}
```

❖ The owner can exclude wallets from rewards



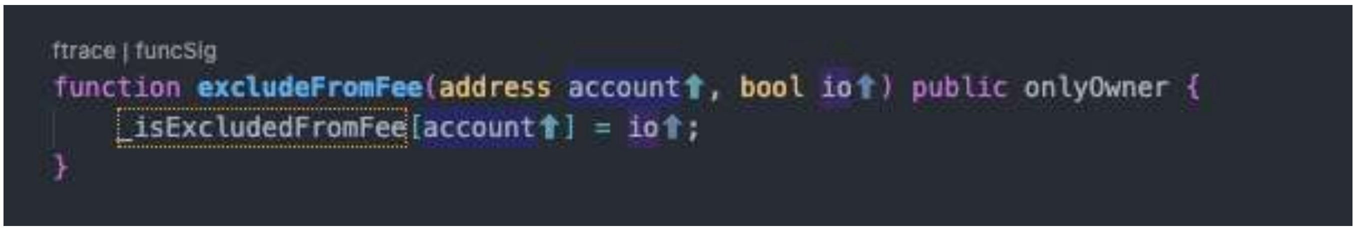
```
function excludeFromReward(address account) public onlyOwner {
    require(!_isExcluded[account], "Account is already excluded");
    if (_rOwned[account] > 0) {
        _tOwned[account] = tokenFromReflection(_rOwned[account]);
    }
    _isExcluded[account] = true;
    _excluded.push(account);
}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

❖ The owner can include wallets from rewards.



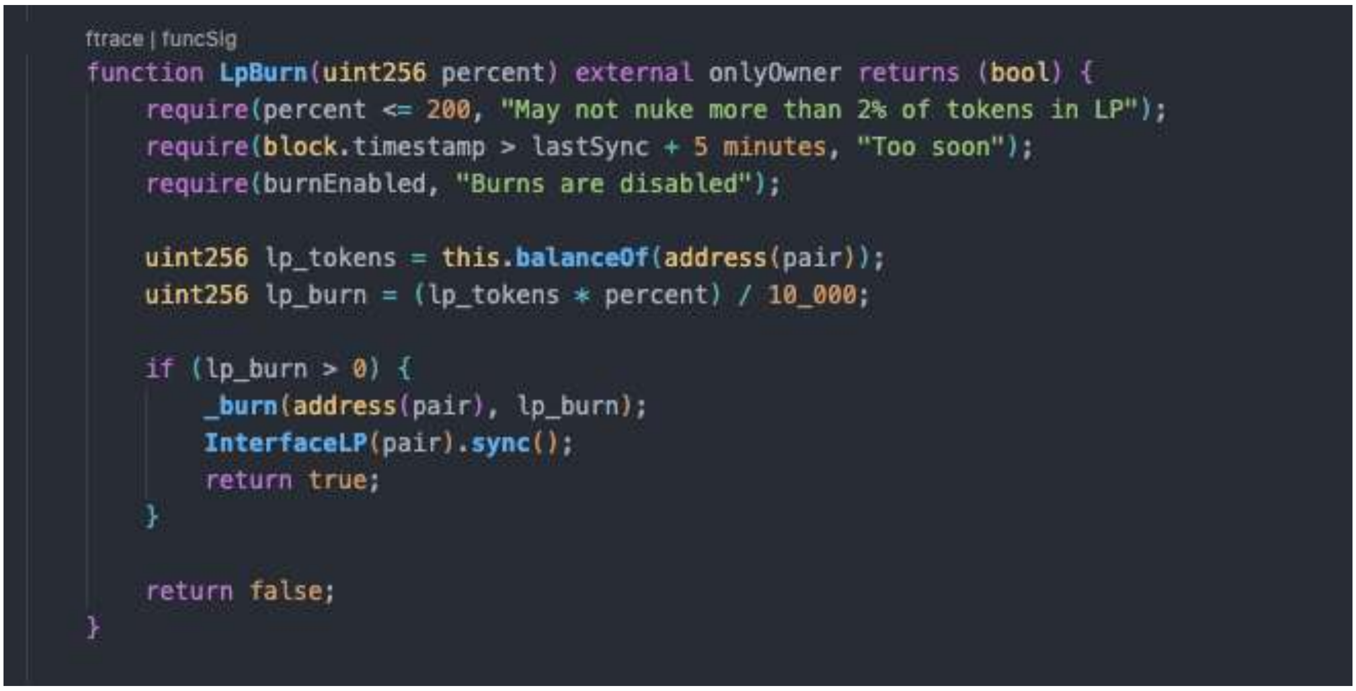
```
ftrace | funcSig
function includeInReward(address account) external onlyOwner {
    require(_isExcluded[account], "Account is not excluded");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _tOwned[account] = 0;
            _isExcluded[account] = false;
            _excluded.pop();
            break;
        }
    }
}
```

- The owner can include/exclude wallets from fees



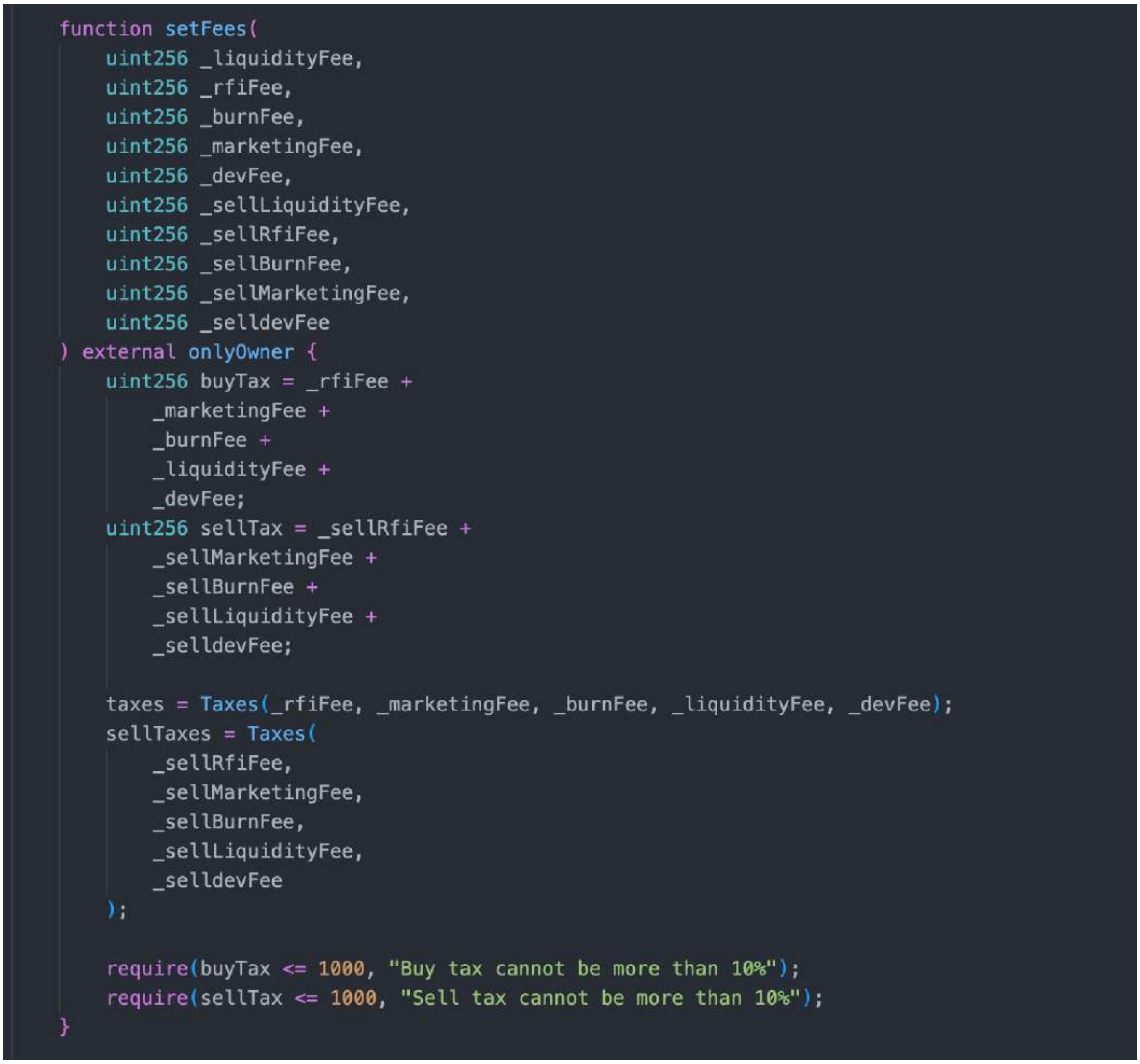
```
ftrace | funcSig
function excludeFromFee(address account, bool io) public onlyOwner {
    _isExcludedFromFee[account] = io;
}
```

- The owner can burn tokens from the lp in every 5 mins maximum of 2%



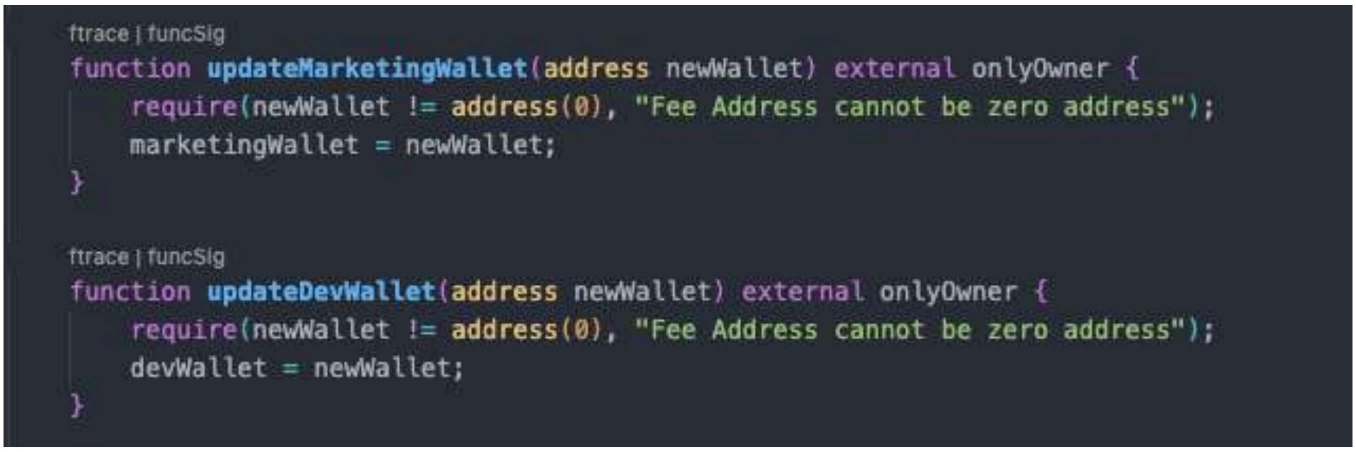
```
ftrace | funcSig
function LpBurn(uint256 percent) external onlyOwner returns (bool) {
    require(percent <= 200, "May not nuke more than 2% of tokens in LP");
    require(block.timestamp > lastSync + 5 minutes, "Too soon");
    require(burnEnabled, "Burns are disabled");

    uint256 lp_tokens = this.balanceOf(address(pair));
    uint256 lp_burn = (lp_tokens * percent) / 10_000;

    if (lp_burn > 0) {
        _burn(address(pair), lp_burn);
        InterfaceLP(pair).sync();
        return true;
    }

    return false;
}
```

❖ The owner can change all buy and fees each maximum of up to 10%



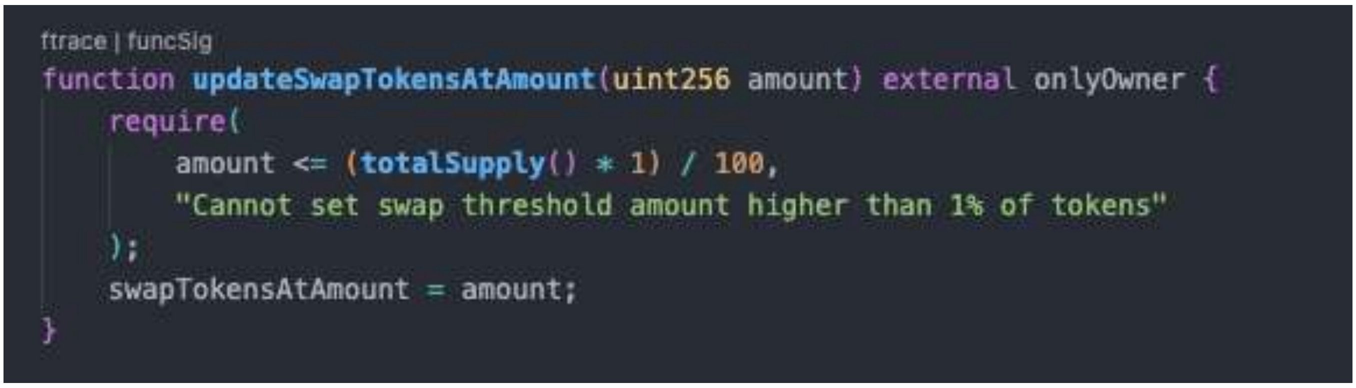
```
function setFees(
    uint256 _liquidityFee,
    uint256 _rfiFee,
    uint256 _burnFee,
    uint256 _marketingFee,
    uint256 _devFee,
    uint256 _sellLiquidityFee,
    uint256 _sellRfiFee,
    uint256 _sellBurnFee,
    uint256 _sellMarketingFee,
    uint256 _selldevFee
) external onlyOwner {
    uint256 buyTax = _rfiFee +
        _marketingFee +
        _burnFee +
        _liquidityFee +
        _devFee;
    uint256 sellTax = _sellRfiFee +
        _sellMarketingFee +
        _sellBurnFee +
        _sellLiquidityFee +
        _selldevFee;

    taxes = Taxes(_rfiFee, _marketingFee, _burnFee, _liquidityFee, _devFee);
    sellTaxes = Taxes(
        _sellRfiFee,
        _sellMarketingFee,
        _sellBurnFee,
        _sellLiquidityFee,
        _selldevFee
    );

    require(buyTax <= 1000, "Buy tax cannot be more than 10%");
    require(sellTax <= 1000, "Sell tax cannot be more than 10%");
}
```

❖ The owner can change the dev and marketing wallet



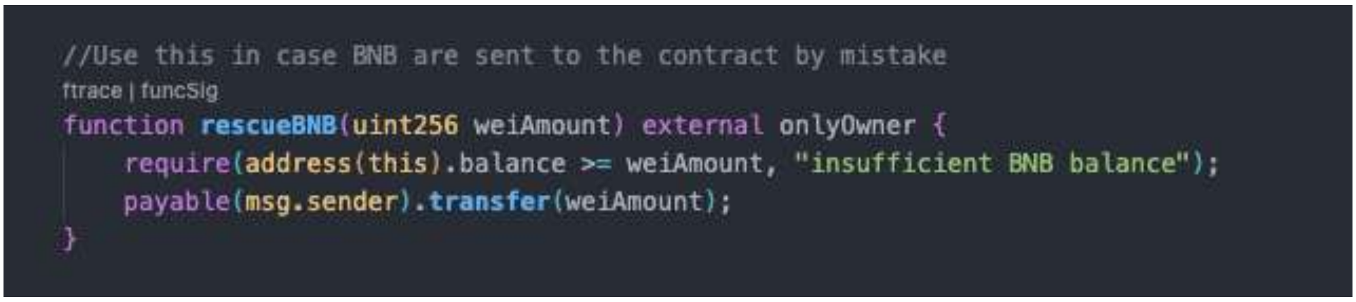
```
ftrace | funcSig
function updateMarketingWallet(address newWallet) external onlyOwner {
    require(newWallet != address(0), "Fee Address cannot be zero address");
    marketingWallet = newWallet;
}

ftrace | funcSig
function updateDevWallet(address newWallet) external onlyOwner {
    require(newWallet != address(0), "Fee Address cannot be zero address");
    devWallet = newWallet;
}
```

❖ The owner can change the swap token amount maximum of up to 1%



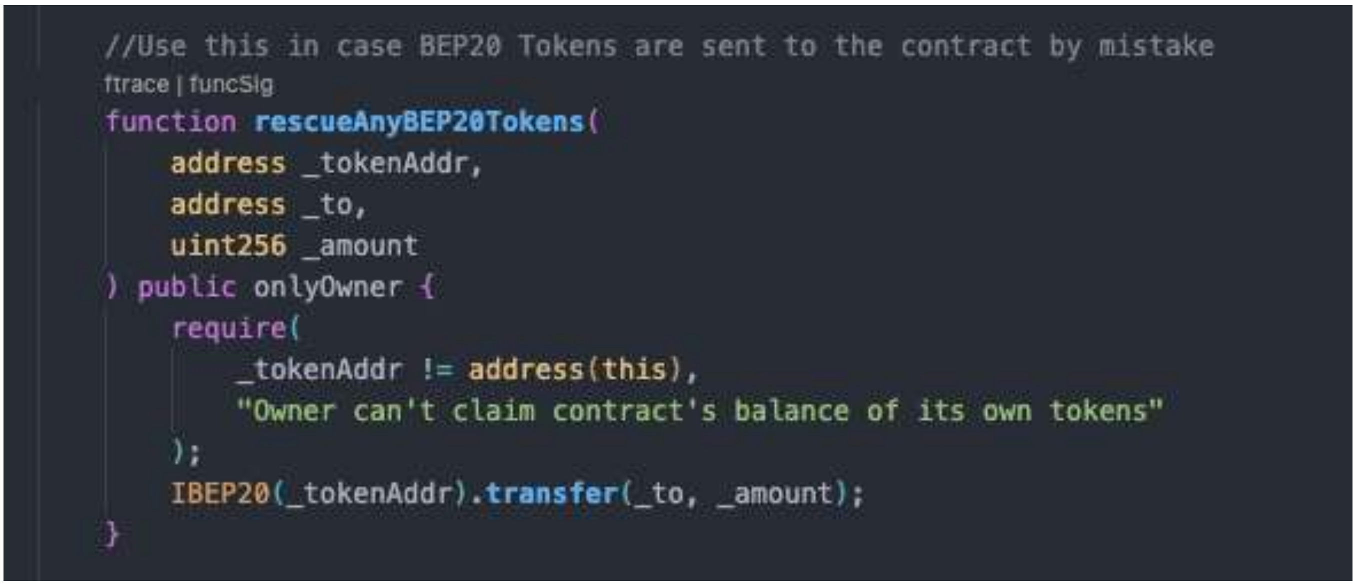
```
ftrace | funcSig
function updateSwapTokensAtAmount(uint256 amount) external onlyOwner {
    require(
        amount <= (totalSupply() * 1) / 100,
        "Cannot set swap threshold amount higher than 1% of tokens"
    );
    swapTokensAtAmount = amount;
}
```

❖ The owner can take BNB from the contract



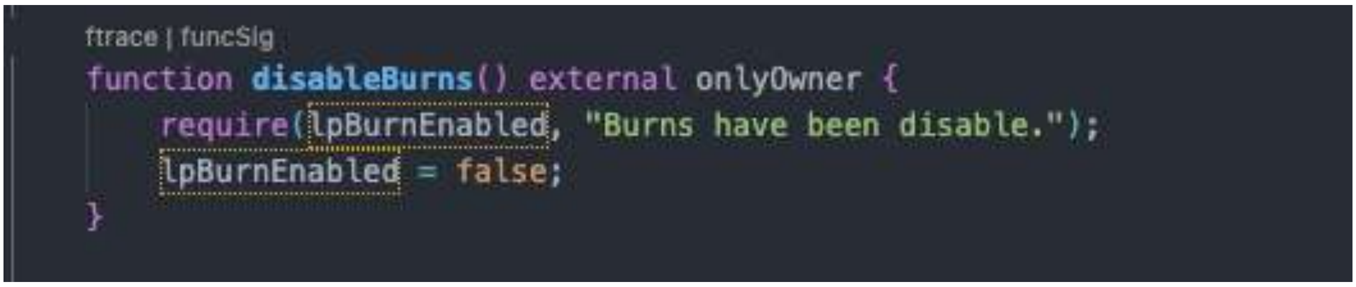
```
//Use this in case BNB are sent to the contract by mistake
ftrace | funcSig
function rescueBNB(uint256 weiAmount) external onlyOwner {
    require(address(this).balance >= weiAmount, "insufficient BNB balance");
    payable(msg.sender).transfer(weiAmount);
}
```

- The owner can take any bep20 token from the contract (can not get native tokens)



```
//Use this in case BEP20 Tokens are sent to the contract by mistake
ftrace | funcSig
function rescueAnyBEP20Tokens(
    address _tokenAddr,
    address _to,
    uint256 _amount
) public onlyOwner {
    require(
        _tokenAddr != address(this),
        "Owner can't claim contract's balance of its own tokens"
    );
    IBEP20(_tokenAddr).transfer(_to, _amount);
}
```

- The owner can disable burn, but once disabled can not enable it again



```
ftrace | funcSig
function disableBurns() external onlyOwner {
    require(lpBurnEnabled, "Burns have been disable.");
    lpBurnEnabled = false;
}
```

Audit conclusion

RugFreeCoins team has performed in-depth testings, line-by-line manual code review, and automated audit of the smart contract. The smart contract was analyzed mainly for common smart contract vulnerabilities, exploits, manipulations, and hacks. According to the smart contract audit.

Smart contract functional Status: **PASS**

Number of risk issues: **0**

Solidity code functional issue level: **PASS**

Number of owner privileges: **11**

Centralization risk correlated to the active owner: **HIGH**

Smart contract active ownership: **ACTIVE**